

Przewodnik Metodyczny

dla nauczycieli informatyki szkół zawodowych
Część druga Java i JavaScript

Opracowanie teoretyczne i metodyczne
Elżbieta Perzycka, Janusz Olczak

Opracowanie scenariuszy lekcji
Aleksander Gralak – moduł Java
Tomasz Czajkowski – moduł JavaScript



Spis treści

SPIS TREŚCI	1
WSTĘP	2
METODY KSZTAŁCENIA	3
PRZYGOTOWANIE NAUCZYCIELA INFORMATYKI	6
ANALIZA PODSTAWY PROGRAMOWEJ KSZTAŁCENIA ZAWODOWEGO	10
MODUŁ SZKOLENIOWY JAVA	21
LEKCJE	33
LEKCJA 1 WSTĘP DO JAVY I ŚRODOWISKA PRACY	33
LEKCJA 2 TYPY PODSTAWOWE I INSTRUKCJE STERUJĄCE	63
LEKCJA 3 WSTĘP DO PROGRAMOWANIA OBIEKTOWEGO	77
LEKCJA 4 INTERFEJSY, KLASY ABSTRAKCYJNE I DZIEDZICZENIE	92
LEKCJA 5 POLIMORFIZM, INSTRUKCJA „FINAL”, KLASY WEWNĘTRZNE I ANONIMOWE	105
LEKCJA 6 TYP WYLICZENIOWY, JAVA DOCS.....	117
LEKCJA 7 TABLICE JEDNOWYMIAROWE I WIELOWYMIAROWE, KOLEKCJE	131
LEKCJA 8 OPERACJE WEJŚCIA – WYJŚCIA I WYJĄTKI.....	143
LEKCJA 9 GRAFICZNY INTERFEJS UŻYTKOWNIKA - SWING	159
LEKCJA 10 PRZECHWYTYWANIE ZDARZEŃ, TWORZENIE PLIKÓW WYKONYWALNYCH	178
LEKCJA 11 PROJEKT INTERDYSCYPLINARNY	189
MODUŁ SZKOLENIOWY JAVASCRIPT	223
LEKCJE	228
LEKCJA 1 WPROWADZENIE	228
LEKCJA 2 ZMIENNE, TYPY DANYCH I OPERATORY	243
LEKCJA 3 INSTRUKCJE WARUNKOWE I PĘTLE.....	257
LEKCJA 4 TABLICE	278
LEKCJA 5 OBIEKTY I FUNKCJE	293
LEKCJA 6 MODEL DOKUMENTU – DOCUMENT OBJECT MODEL	315
LEKCJA 7 ZDARZENIA	325
LEKCJA 8 OBSŁUGA BŁĘDÓW	335
LEKCJA 9 WYKORZYSTANIE JĘZYKA JAVASCRIPT	343
LEKCJA 10 BIBLIOTEKA JQUERY	366
LEKCJA 11 PROJEKT INTERDYSCYPLINARNY	383
BIBLIOGRAFIA	390

Wstęp

Szkoła, a w niej edukacja przechodzi transformację nie tylko spowodowaną legislacyjnymi zmianami, ale także alternatywnym do obecnego zrozumieniem procesu kształcenia. Z jednej strony jest to dążenie do rewitalizacji i kultywowania tradycyjnych wartości, które mieszczą się w szeroko rozumianym nurcie humanizmu, z drugiej strony podporządkowanie się w coraz większym stopniu narastającej fali technologicznej. Zmiany wynikają także z oczekiwań jakie stawia się absolwentom szkoły zawodowej, w szczególności: umiejętnościom rozwiązywania problemów i podejmowania decyzji; krytycznym i twórczym myśleniem; umiejętnościami współpracy i negocjacji; intelektualną ciekawością; wyszukiwaniem, selekcjonowaniem, porządkowaniem i ocenianiem informacji; wykorzystywaniem wiedzy w nowych sytuacjach; integrowaniem technologii z kształceniem i własnym rozwojem (Sysło, 2010).

Tradycyjny model edukacji oparty na behawiorystycznym podejściu do uczenia się ustępuje miejsca refleksji krytycznej, na której wyrósł konstruktywizm. Jego idea nie jest nowa. Odnajdziemy ją w pracach J.Piageta, L.Wygotskiego, J.Brunera czy J.Deweya. Potrzeba poznania tkwi w każdym człowieku i to ona motywuje nas do działania. Instrumentalne podejście do edukacji sprawdza się w wyjaśnianiu prostych opartych na przekazie i pamięci zadań, jednak nie wystarcza, kiedy potrzebujemy uzasadnienia i wyjaśnienia zróżnicowanych zadań w złożonych sytuacjach życia codziennego. Podejście krytyczno-refleksyjne traktuje ucznia i nauczyciela nie tylko jako układ tworzący wiedzę, ale jako aktywne, samodzielnie wybierające i przetwarzające informacje podmioty edukacji. W takim układzie nauczyciel i uczeń podejmują samodzielnie decyzje, realizują indywidualne strategie nauczania i uczenia się. Współistnienie w edukacji informatycznej alternatywnego podejścia do wprowadzania pojęć i rozwijania umiejętności informatycznych jest na dzień dzisiejszy komplementarnym podejściem do kształcenia.

Propozycją wychodzącą naprzeciw oczekiwaniom rynku pracy i efektywnej edukacji jest niniejszy Przewodnik Metodyczny. Zamieszczone w nim scenariusze lekcji odpowiadają założeniom Podstawy Programowej Kształcenia w Zawodzie. Treści przewodnika uwzględniają zintegrowane i skorelowane treści kształcenia ogólnego i zawodowego – informatycznego w tym doskonalenie kompetencji kluczowych nabytych w procesie kształcenia ogólnego, z uwzględnieniem niższych etapów edukacyjnych. W szczególności:

- celom kształcenia zawodowego, których realizacja doprowadzi do przygotowania uczniów do życia w warunkach współczesnego świata, wykonywania pracy zawodowej i aktywnego funkcjonowania na zmieniającym się technologicznym rynku pracy.

- zadaniom szkoły, warunkowanym zmianami zachodzącymi w otoczeniu gospodarczo-społecznym, na które wpływają w szczególności: idea gospodarki opartej na wiedzy, globalizacja procesów gospodarczych i społecznych, rosnący udział handlu międzynarodowym, mobilność geograficzna i zawodowa, nowe techniki i technologie, a także wzrost oczekiwań pracodawców w zakresie poziomu wiedzy i umiejętności pracowników – techników - informatyków.

Autorzy przewodnika pochyłili się tutaj nad koncepcją sfery aktualnego i najbliższego rozwoju zaproponowaną przez L.Wygotskiego. Odpowiedni poziom wiedzy ogólnej powiązanej z wiedzą zawodową - informatyczną przyczyni się do podniesienia poziomu umiejętności zawodowych

absolwentów szkół kształcących w zawodzie technik - informatyk, a tym samym zapewni im możliwość sprostania wyzwaniom zmieniającego się rynku pracy.

W przewodniku uwzględniono możliwości dostosowania treści z zakresu tworzenia aplikacji internetowych na przykładach czterech języków programowania (Java, JavaScript, SQL, PHP) do poziomu rozwojowego uczących się. Zagadnienia tematyczne zestawiono w bloki problemowe, jako propozycje zajęć lekcyjnych. Wybór lekcji na kolejne zajęcia jest warunkowany wyborem nauczyciela i może być podyktowany rozpoznaniem poziomu zrozumienia treści przez ucznia i rozpoczynać się od dowolnej lekcji. Kolejność lekcji nie jest obowiązkowa. To nauczyciel decyduje jaki problem będzie analizował z uczniami na kolejnym spotkaniu.

To za sprawą znajomości różnych języków programowania uczniowie będą tworzyć własne zasoby – edukacyjne, kulturalne, informacyjne, artystyczne, graficzne, rozrywkowe. Będą konstruować swój świat – siebie.

Metody kształcenia

Czy metody stosowane dziś w dydaktyce informatyki są skuteczne? Jak wprowadzać ucznia w świat języków programowania, aby był on dla niego przyjazny i rozpoznawalny?

W zakresie kształcenia informatycznego możemy wyróżnić dwie grupy zagadnień:

I grupa – zagadnienia ogólnokształcące, które mogą być realizowane za pomocą metod przekazu (pogadanka, pokaz, referat ucznia, dyskusja, itp.)

II grupa - zagadnienia specjalistyczne wymagające bezpośredniej pracy z komputerem, realizowane przy zastosowaniu metod aktywnych (np. dyskusja, praca w grupach) koncentrujących się na uczniu (np. samodzielne wykonywanie zadań problemowych, ćwiczeń, tworzenie, projektowanie).

Niezależnie jednak od tego jaki zakres zagadnień realizuje nauczyciel z uczniami na zajęciach informatycznych nie może zapominać o zasadach metodycznego przygotowania.

Po pierwsze bardzo ważne jest, aby proces kształcenia przebiegał na poziomie dającym uczniom poczucie uczestnictwa w wartościowych dla nich zajęć. Będzie to możliwe kiedy poziom kształcenia zostanie dostosowany do poziomu rozwoju poznawczego ucznia (aktualnych umiejętności i wcześniejszych osiągnięć), tempa przyswajania nowych informacji, postaw wobec uczenia się i oczekiwań wobec lekcji.

Po drugie język informatyki składa się z wielu trudnych terminów. Możemy spotkać się także z sytuacją ukrytego żargonu. Dla przykładu, menu w życiu codziennym oznacza zupełnie co innego niż w informatyce. Uczniowie mogą nie znać języka używanego przez nauczyciela i odwrotnie, uczniowie mogą stosować terminy odmienne niż nauczyciel, ale znaczeniowo takie same. Nauczyciele mają skłonność do budowania bardziej skomplikowanych zdań niż ich uczniowie (Pety, s.47-48). Na zajęciach informatycznych wskazane jest komunikowanie dostosowane do obu stron procesu kształcenia.

Po trzecie z różnych powodów uczniowie mogą nie być zadowoleni z lekcji. To wynika z różnych dotychczasowych doświadczeń szkolnych. Warto zastanowić się nad zmotywowaniem uczniów do uczenia się. Jak to zrobić? Należy rozpocząć od odpowiedzi na pytanie, czy byli zmotywowani, gdy przyszli na pierwsze zajęcia? Czy motywacja słabła podczas kolejnych zajęć? Ważne są rozmowy z całą klasą jak również z pojedynczymi uczniami o tym, co jest dla nich ważne i dlaczego uczą się języków programowania. Znając problemy swoich uczniów nauczyciel może odnaleźć pomysł na poprawienie ich motywacji zmienić postawy do uczenia się (Perzycka, 2008).

Informatyka dostarcza jakościowo innych warunków niż pozostałe przedmioty kształcenia. Po rozpoznaniu potrzeb i oczekiwań uczniów oraz treści podstawy programowej kształcenia ogólnego nauczyciel realizuje lekcje z uwzględnieniem reguł wprowadzania do języków programowania. Reguły odnoszą się zarówno do nauczyciela jak i do ucznia. Stymulacja najbardziej naturalnych warunków pracy – jest problem do rozwiązania. Trzeba to zrobić szybko i dobrze. Każdy pracuje samodzielnie i nie przeszkadza innym. O efekcie decyduje czas spędzony przy komputerze. Lekcje informatyki to nie czytelnia, można przygotować się w domu. W domu należy przeczytać zadanie i zastanowić się jak je wykonać i gdzie szukać potrzebnych informacji. Przy korzystaniu z instrukcji najważniejsze jest rozeznanie, gdzie znajduje się opis danej metody rozwiązania problemu, czy funkcji programu. Instrukcje czyta się jak encyklopedię. Może to zrobić uczeń lub przekazać nauczyciel. Przy pierwszym pytaniu ucznia nauczyciel pokaże jak znaleźć odpowiedź czy objaśnienie w instrukcji, przy drugim – odeśle do instrukcji, przy trzecim – może wprowadzić w błąd! Kto szybciej rozwiąże postawione problemy będzie miał szansę pogłębić zrealizowany materiał i dostanie nowe zadanie.

Po rozwiązaniu problemów następuje czas na zadania sprawdzające. Każdy dostaje nowe problemy. Nauczyciel wystawia ocenę za wykonane zadanie wcześniej informując ucznia o kryteriach oceniania. Na sprawdzianach z poszczególnych języków programowania można korzystać z dowolnych źródeł. Sprawdziany mają wykazać, czy uczniowie nauczyli się samodzielnie rozwiązywać problemy za pomocą poznanego materiału na kolejnych lekcjach. Oceniany będzie poziom i czas rozwiązania.

W początkowym okresie nauki języków programowania uczeń stoi przed bardzo trudnym zadaniem. Musi bowiem równocześnie: zrozumieć co do niego mówi nauczyciel, nauczyć się interpretować sytuację na ekranie i nauczyć się wydawać polecenia systemowi i programom (Bougon, Weick, Binkhorst, s. 606-639).

Ważnym czynnikiem mającym znaczenie w procesie kształcenia jest problem wielozadaniowości, występujący u młodzieży w toku ich uczenia się. Dlatego zarówno ilość treści jak i sposób ich przyswajania mają decydujący wpływ na rozwój poznawczy.

Podczas pracy nad przygotowaniem przewodników dla nauczycieli skoncentrowano się na takim doborze zadań, aby uwzględniały zarówno percepcję ucznia, jak i stopień trudności treści kształcenia. W ich opracowaniu przyjęto zasadę stopniowania trudności według poziomów przetwarzania informacji, które można rozumieć jako zakres i intensywność przetwarzania informacji, zgodnie z teorią F.I.M.Craika, R.S.Lockharta.

W kontekście poziomów przetwarzania informacji proponujemy stopniowanie trudności uczenia się, poprzez dobór odpowiednich zadań, ćwiczeń i problemów do rozwiązania.

Poziom pierwszy to percepcja sensoryczna. Zastosowane metody podające: wykład i ćwiczenie krok po kroku pozwalają na przekazanie treści, ale nie dają gwarancji trwałego zapamiętania. Metody podające wymagają częstego powtarzania w celu zapamiętania treści – wyuczenia się materiału. Rezultaty są podatne na wszelkiego rodzaju zakłócenia.

Poziom drugi to semantyczny odbiór przekazywanych treści. Zastosowane metody zadaniowe polegające na uzupełnieniach, dokończeniach pozwalają na trwalsze zapamiętanie treści, ale wciąż wymagają powtórzeń. Uczeń rozpoznaje znaczenie poprzez interpretacje odbieranych danych i informacji w formie treści zadań, ćwiczeń.

Poziom trzeci to aktywizowanie wiedzy już posiadanej. Zastosowanie metod problemowych pozwoli wykonać zadanie na zasadzie skojarzeń podczas reorganizacji wiedzy, którą zdobył uczeń na poprzednich poziomach przetwarzania informacji. Im wyższy poziom przetwarzania informacji tym trwalsze zapamiętywanie i głębsze zrozumienie treści, a tym samym skuteczniejsze uczenie się.

Nauczyciel może rozpocząć zajęcia od metody kursowej, polegającej na kolejnym omawianiu poszczególnych funkcji języka programowania i ćwiczeniach z ich wykorzystaniem. Takie rozwiązanie jest przydatne dla osób, które będą zawodowo wykorzystywały konkretny program. W warunkach szkoły sprawia, to wrażenie, że uczy się rzeczy zupełnie później nieprzydatnych, dlatego w dalszych zajęciach wskazane jest wiązanie zadań z codziennymi sytuacjami żywymi/zawodowymi. W większości przypadków uczeń działa sam na sam z komputerem. Po stronie nauczyciela zatem stoi wyzwanie w kierunku precyzji formułowania zadań i odpowiedzialność za ich realizację. Chodzi przede wszystkim o to by uczeń po zapoznaniu się z problemem sam potrafił go rozwiązać. Programowanie to proces twórczy i może prowadzić do innowacyjnych rozwiązań i do odkryć. W takim układzie uczeń staje się odkrywcą, a języki programowania komunikują jego odkrycia.

Przygotowanie nauczyciela informatyki

Poszukując wytycznych dotyczących przygotowania nauczyciela informatyki do prowadzenia zajęć w zawodowej szkole ponadgimnazjalnej, w dostępnej literaturze przedmiotu odnajdziemy standardy przygotowania nauczyciela informatyki, które zostały opisane w dokumencie z roku 2003, zatytułowanym Standardy przygotowania nauczycieli w zakresie technologii informacyjnej i informatyki przygotowanym przez Radę ds. Edukacji Informatycznej i Medialnej działającej przy Ministerstwie Edukacji Narodowej, pod kierunkiem Profesora Macieja M.Systry.

Nauczyciel informatyki zdefiniowany został jako nauczyciel przygotowany do prowadzenia zajęć z przedmiotu informatyka w zakresie rozszerzonym w szkole ponadgimnazjalnej, które mogą kończyć się egzaminem maturalnym. Jego kompetencje są rozszerzeniem kompetencji nauczyciela technologii informacyjnej o zagadnienia z informatyki, jako dziedziny naukowej.

Szczegółowe, postulowane kompetencje nauczyciela informatyki zostały opisane w czterech blokach zagadnieniowych. Są one następujące:

1. Profesjonalne przygotowanie w zakresie informatyki

Nauczyciel wzbogaca kompetencje wymienione w punkcie 1, aby osiąść wiedzę, umiejętności i doświadczenie, odpowiednie do zakresu i poziomu wiedzy informatycznej, niezbędnej do nauczania informatyki, jako dyscypliny naukowej na poziomie szkoły ponadgimnazjalnej.

W szczególności:

1.1. Zna podstawy informatyki w zakresie: historii, struktury dziedziny informatyka i jej matematycznych podstaw (elementy teorii mnogości i logiki) oraz powiązań z innymi dziedzinami nauki.

1.2. Zna zaawansowane metody wykorzystywania oraz programowania narzędzi informatycznych takich, jak: edytor tekstu, arkusz kalkulacyjny, system zarządzania bazą danych, wybrane programy edukacyjne.

1.3. Ma praktyczną znajomość przynajmniej dwóch języków programowania wyższego poziomu, w tym języka algorytmicznego i języka do tworzenia prezentacji w sieci, w zakresie projektowania programów oraz testowania i weryfikowania ich poprawności.

1.4. Zna podstawowe metody algorytmicznego rozwiązywania problemów: techniki algorytmiczne, podstawowe algorytmy, algorytmy rozwiązywania klasycznych problemów, własności algorytmów, oraz podstawowe metody analizy algorytmów i ich własności: poprawności, skończoności, złożoności obliczeniowej i efektywności praktycznej.

1.5. Zna podstawowe struktury danych, stosuje je w powiązaniu z wybranymi algorytmami oraz w realizacji algorytmów w wybranym języku programowania.

1.6. Zna budowę typowej lokalnej sieci komputerowej oraz sieci Internet, w tym zasady funkcjonowania architektury klient-serwer.

1.7. Ma osobiste doświadczenie, nabyte w pracowni komputerowej, w zakresie:

- pracy w środowisku różnych systemów operacyjnych i systemów sieciowych;
- administrowania siecią komputerową na potrzeby swoich zajęć;
- praktycznej znajomości zagadnień wymienionych w punktach 5.1.2 - 5.1.6;
- realizacji zespołowych projektów programistycznych.

2. Warsztat pracy nauczyciela informatyki: sprzęt i oprogramowanie

Nauczyciel wzbogaca kompetencje wymienione w punkcie 2 o znajomość kryteriów doboru, sposobów instalowania i utrzymywania w sprawności sprzętu i oprogramowania, niezbędnego do prowadzenia zajęć z informatyki w szkole ponadgimnazjalnej, znajdującego się na wyposażeniu szkolnej pracowni komputerowej.

2.1. Dostosowuje konfigurację sprzętu i oprogramowania do potrzeb swoich zajęć z informatyki. Planuje dalszy ich rozwój zaplecza technicznego swoich zajęć. Radzi sobie z prostymi awariami w pracowni komputerowej.

2.2. Zapewnia przestrzeganie nakazów prawnych i norm etycznych w korzystaniu z TI w pracowni szkolnej. Zapewnia i chroni bezpieczeństwo danych oraz prywatność innych osób.

2.3. Stosuje pełne możliwości: edytorów (tekstu, grafiki i muzyki), systemów składu i arkusza kalkulacyjnego.

2.4. Projektuje i tworzy bazy danych (w tym relacyjne) i korzysta z systemów zarządzania bazą danych.

2.5. Wykorzystuje pełne możliwości narzędzi sieciowych do: poszukiwania, gromadzenia i przetwarzania informacji, komunikacji elektronicznej, tworzenia stron i serwisów internetowych, prezentacji multimedialnych oraz udostępniania ich w sieci.

2.6. Stosuje pełne możliwości środowiska, umożliwiającego integrowanie: edytorów tekstu, grafiki i muzyki, arkusza kalkulacyjnego, systemu zarządzania bazą danych, programów komunikacyjnych, usług internetowych i innych mediów.

2.7. Korzysta z oprogramowania edukacyjnego i systemów autorskich do tworzenia mediów dydaktycznych. Tworzy własne pomoce z wykorzystaniem narzędzi informatycznych, np. języka programowania.

2.8. Określa działania i środki wspomagające swój profesjonalny rozwój w zakresie informatyki i TI, uwzględnia w tym zachodzące zmiany technologiczne.

3. Metodyka nauczania informatyki i jej zastosowań

Nauczyciel wzbogaca kompetencje wymienione w punkcie 3 o umiejętności nauczania informatyki jako dziedziny akademickiej, przynajmniej w zakresie wymienionym w punktach 3.1 i 3.2.

W szczególności:

3.1. Określa zakres nauczania informatyki w szkole ponadgimnazjalnej na podstawie obowiązującej podstawy programowej tego przedmiotu i programów jego nauczania oraz programów studiów informatycznych.

3.2. Dostosowuje program nauczania i metody nauczania do zmian zachodzących w informatyce, w technologii informatycznej i w technologii informacyjnej.

3.3. Opracowuje i stosuje praktycznie metody nauczania pojęć i kształtowania umiejętności informatycznych, w tym również w pracowni komputerowej, związanych z:

- posługiwaniem się systemami komputerowymi i sieciowymi oraz technologią informacyjną,
- korzystaniem z oprogramowania użytkowego i edukacyjnego oraz języków programowania,
- docieraniem do lokalnych i rozproszonych źródeł informacji, korzystaniem z nich oraz przetwarzaniem i prezentowaniem informacji w różnej postaci.

3.4. Opracowuje i stosuje praktycznie metody: modelowania problemów i ich rozwiązywania, przedstawiania rozwiązań w postaci algorytmicznej oraz otrzymywania rozwiązań za pomocą wybranych narzędzi informatycznych, w tym również za pomocą programów własnych.

3.5. Zna podstawowe zastosowania informatyki w innych dziedzinach kształcenia i stosuje praktycznie nauczanie informatyki zintegrowane z innymi przedmiotami; współpracuje w tym zakresie z nauczycielami innych przedmiotów.

3.6. Projektuje i stosuje w nauczaniu zespołowe metody pracy, w tym zespołowy projekt programistyczny.

3.7. Doradza uczniom w zakresie wyboru drogi dalszego kształcenia się w szkole wyższej na specjalności informatycznej.

3.8. Projektuje i rozwija metody oceniania osiągnięć uczniów w zdobywaniu umiejętności i kompetencji informatycznych.

4. Aspekty humanistyczne, etyczno-prawne i społeczne w dostępie do TI i w korzystaniu z tej technologii

Nauczyciel jest świadomy, że TI może powodować (w tym również w szkole) powstawanie wielu kwestii prawnych, etycznych i społecznych, a także zagrożeń w tych sferach. Dbą

o przestrzeganie norm prawnych i etycznych oraz zasad równouprawnienia w dostępie do komputerów i technologii informacyjnej oraz w posługiwaniu się nią przez uczniów. Przestrzega i wpaja uczniom normy współżycia w kształtującym się społeczeństwie informacyjnym. Wszystkie te kwestie potrafi przedstawić i uzasadnić uczniom. W szczególności:

- 4.1. Przestrzega norm prawnych i etycznych w korzystaniu ze źródeł informacji oraz w posługiwaniu się informacją w swojej pracy i w pracy z uczniami. Wyjaśnia pochodzenie i zasadność stosowania tych norm. Odróżnia i wyjaśnia różnice między korzystaniem z cudzej własności intelektualnej z powołaniem się na autora a plagiatem. Szanuje prywatność innych użytkowników TI - w tym również uczniów - i chroni ich dane oraz zasoby.
- 4.2. Uwzględnia w nauczaniu humanistyczne, etyczno-prawne i społeczne aspekty stosowania informatyki przez uczniów, w szkole i poza nią, w tym również w celach osobistych
- 4.3. Jest świadomy różnic kulturowych tkwiących w informacji, sposobie jej przedstawiania oraz udostępniania. Dbą o zagwarantowanie uczniom równych praw dostępu do komputerów, TI oraz informacji, bez względu na pochodzenie społeczne i kulturowe, płeć, zamożność i wcześniejsze przygotowanie.
- 4.4. Zna podstawowe zasady etyki w korzystaniu z mediów. Stosuje je w świadomym i krytycznym odbiorze komunikatów medialnych.
- 4.5. Zna zagrożenia (w tym etyczne i prawne) wynikające z niewłaściwego posługiwania się komputerami oraz korzystania z nieodpowiedniego dla uczniów oprogramowania i źródeł informacji. Potrafi skutecznie przeciwdziałać tym zagrożeniom, chroniąc przed nimi uczniów oraz kształtując ich świat pozytywnych wartości.
- 4.6. Zna zagrożenia psychiczne i fizyczne dla zdrowia, wynikające z nadmiernego lub niewłaściwego korzystania ze środków TI. Wystrzega się tych zagrożeń, uprzedza o nich uczniów i chroni ich przed nimi.
- 4.7. Jest świadomy wpływu na demokrację swobodnego dostępu do informacji oraz nieskrępowanej komunikacji. Dostrzega zagrożenia przez to powodowane, związane z globalizacją procesów społecznych.
- 4.8. Jest świadomy wpływu TI na zachowania społeczne. Potrafi przedstawić trendy w rozwoju techniki i technologii informatycznej i informacyjnej, oraz konsekwencje tego rozwoju dla życia, w tym zawodowego, jednostki i całych społeczności w tworzącym się społeczeństwie informacyjnym. Zna i przedstawia oficjalne wskazania gremiów krajowych i międzynarodowych, odnoszące się do kierunków zmian powodowanych, rozwojem informatyki i TI w skali państwa, Europy i świata.

Analiza podstawy programowej kształcenia zawodowego

Cele kształcenia w zawodzie

Absolwent szkoły kształcącej w zawodzie technik informatyk powinien być przygotowany do wykonywania następujących zadań zawodowych:

- 1) montowania oraz eksploatacji komputera i urządzeń peryferyjnych;
- 2) projektowania i wykonywania lokalnych sieci komputerowych, administrowania tymi sieciami;
- 3) projektowania baz danych i administrowania bazami danych;
- 4) tworzenia stron www i aplikacji internetowych, administrowania tymi stronami i aplikacjami.

Efekty kształcenia

Do wykonywania wyżej wymienionych zadań zawodowych niezbędne jest osiągnięcie zakładanych efektów kształcenia, na które składają się:

- efekty kształcenia wspólne dla wszystkich zawodów;

(BHP). Bezpieczeństwo i higiena pracy

Uczeń:

- 1) rozróżnia pojęcia związane z bezpieczeństwem i higieną pracy, ochroną przeciwpożarową, ochroną środowiska i ergonomią;
- 2) rozróżnia zadania i uprawnienia instytucji oraz służb działających w zakresie ochrony pracy i ochrony środowiska w Polsce;
- 3) określa prawa i obowiązki pracownika oraz pracodawcy w zakresie bezpieczeństwa i higieny pracy;
- 4) przewiduje zagrożenia dla zdrowia i życia człowieka oraz mienia i środowiska związane z wykonywaniem zadań zawodowych;
- 5) określa zagrożenia związane z występowaniem szkodliwych czynników w środowisku pracy;
- 6) określa skutki oddziaływania czynników szkodliwych na organizm człowieka;
- 7) organizuje stanowisko pracy zgodnie z obowiązującymi wymaganiami ergonomii, przepisami bezpieczeństwa i higieny pracy, ochrony przeciwpożarowej i ochrony środowiska;
- 8) stosuje środki ochrony indywidualnej i zbiorowej podczas wykonywania zadań zawodowych;
- 9) przestrzega zasad bezpieczeństwa i higieny pracy oraz stosuje przepisy prawa dotyczące ochrony przeciwpożarowej i ochrony środowiska;

- 10) udziela pierwszej pomocy poszkodowanym w wypadkach przy pracy oraz w stanach zagrożenia zdrowia i życia.

(PDG). Podejmowanie i prowadzenie działalności gospodarczej

Uczeń:

- 1) stosuje pojęcia z obszaru funkcjonowania gospodarki rynkowej;
- 2) stosuje przepisy prawa pracy, przepisy prawa dotyczące ochrony danych osobowych oraz przepisy prawa podatkowego i prawa autorskiego;
- 3) stosuje przepisy prawa dotyczące prowadzenia działalności gospodarczej;
- 4) rozróżnia przedsiębiorstwa i instytucje występujące w branży i powiązania między nimi;
- 5) analizuje działania prowadzone przez przedsiębiorstwa funkcjonujące w branży;
- 6) inicjuje wspólne przedsięwzięcia z różnymi przedsiębiorstwami z branży;
- 7) przygotowuje dokumentację niezbędną do uruchomienia i prowadzenia działalności gospodarczej;
- 8) prowadzi korespondencję związaną z prowadzeniem działalności gospodarczej;
- 9) obsługuje urządzenia biurowe oraz stosuje programy komputerowe wspomagające prowadzenie działalności gospodarczej;
- 10) planuje i podejmuje działania marketingowe prowadzonej działalności gospodarczej;
- 11) optymalizuje koszty i przychody prowadzonej działalności gospodarczej.

(JOZ). Język obcy ukierunkowany zawodowo

Uczeń:

- 1) posługuje się zasobem środków językowych (leksykalnych, gramatycznych, ortograficznych oraz fonetycznych), umożliwiających realizację zadań zawodowych;
- 2) interpretuje wypowiedzi dotyczące wykonywania typowych czynności zawodowych artykułowane powoli i wyraźnie, w standardowej odmianie języka;
- 3) analizuje i interpretuje krótkie teksty pisemne dotyczące wykonywania typowych czynności zawodowych;
- 4) formułuje krótkie i zrozumiałe wypowiedzi oraz teksty pisemne umożliwiające komunikowanie się w środowisku pracy;
- 5) korzysta z obcojęzycznych źródeł informacji.

(KPS). Kompetencje personalne i społeczne

Uczeń:

- 1) przestrzega zasad kultury i etyki;
- 2) jest kreatywny i konsekwentny w realizacji zadań;
- 3) przewiduje skutki podejmowanych działań;
- 4) jest otwarty na zmiany;
- 5) potrafi radzić sobie ze stresem;
- 6) aktualizuje wiedzę i doskonali umiejętności zawodowe;
- 7) przestrzega tajemnicy zawodowej;
- 8) potrafi ponosić odpowiedzialność za podejmowane działania;
- 9) potrafi negocjować warunki porozumień;
- 10) współpracuje w zespole.

(OMZ). Organizacja pracy małych zespołów (wyłącznie dla zawodów nauczanych na poziomie technika)

Uczeń:

- 1) planuje pracę zespołu w celu wykonania przydzielonych zadań;
 - 2) dobiera osoby do wykonania przydzielonych zadań;
 - 3) kieruje wykonaniem przydzielonych zadań;
 - 4) ocenia jakość wykonania przydzielonych zadań;
 - 5) wprowadza rozwiązania techniczne i organizacyjne wpływające na poprawę warunków i jakość pracy;
 - 6) komunikuje się ze współpracownikami.
- efekty kształcenia wspólne dla zawodów w ramach obszaru elektryczno-elektronicznego, stanowiące podbudowę do kształcenia w zawodzie lub grupie zawodów PKZ(E.b);

PKZ(E.b) Umiejętności stanowiące podbudowę do kształcenia w zawodach: technik informatyk, technik tyfloinformatyk, technik teleinformatyk

Uczeń:

- 1) rozpoznaje symbole graficzne i oznaczenia podzespołów systemu komputerowego;
- 2) dobiera elementy i konfiguracje systemu komputerowego;
- 3) dobiera oprogramowanie użytkowe do realizacji określonych zadań;
- 4) stosuje zabezpieczenia sprzętu komputerowego i systemu operacyjnego;
- 5) rozróżnia parametry sprzętu komputerowego;

- 6) charakteryzuje informatyczne systemy komputerowe;
 - 7) określa funkcje systemu operacyjnego;
 - 8) posługuje się terminologią dotyczącą lokalnych sieci komputerowych;
 - 9) charakteryzuje urządzenia sieciowe;
 - 10) charakteryzuje rodzaje oprogramowania użytkowego;
 - 11) korzysta z publikacji elektronicznych;
 - 12) przestrzega zasad zarządzania projektem w trakcie organizacji i planowania pracy;
 - 13) stosuje programy komputerowe wspomagające wykonywanie zadań.
- efekty kształcenia właściwe dla kwalifikacji wyodrębnionych w zawodzie technik informatyk opisane w części II:

E.12. Montaż i eksploatacja komputerów osobistych oraz urządzeń peryferyjnych

1. Przygotowanie stanowiska komputerowego do pracy

Uczeń:

- 1) stosuje systemy liczbowe używane w technice komputerowej;
- 2) wymienia funkcje i przestrzega zasad działania poszczególnych elementów jednostki centralnej komputera;
- 3) dobiera urządzenia techniki komputerowej do określonych warunków technicznych;
- 4) montuje komputer osobisty z podzespołów;
- 5) modernizuje i rekonfiguruje komputery osobiste;
- 6) planuje przebieg prac związanych z przygotowaniem komputera osobistego do pracy;
- 7) instaluje i aktualizuje systemy operacyjne i aplikacje;
- 8) stosuje polecenia systemów operacyjnych do zarządzania systemem;
- 9) instaluje i konfiguruje sterowniki urządzeń;
- 10) konfiguruje ustawienia personalne użytkownika w systemie operacyjnym;
- 11) stosuje oprogramowanie narzędziowe systemu operacyjnego;
- 12) stosuje oprogramowanie zabezpieczające;
- 13) odczytuje dokumentację techniczną informatycznych systemów komputerowych;
- 14) opracowuje wskazania do użytkowania systemu operacyjnego;
- 15) sporządza cenniki i kosztorysy stanowisk komputerowych;
- 16) opracowuje dokumentację techniczną stanowiska komputerowego;

17) stosuje przepisy prawa autorskiego w zakresie dotyczącym systemów informatycznych;

18) rozpoznaje rodzaje licencji oprogramowania komputerowego;

19) stosuje przepisy prawa dotyczące certyfikacji CE i recyklingu.

2. Użytkowanie urządzeń peryferyjnych komputera osobistego

Uczeń:

1) wyjaśnia zasadę działania interfejsów komputera osobistego;

2) wyjaśnia zasadę działania urządzeń peryferyjnych komputera osobistego;

3) przygotowuje urządzenia peryferyjne komputera osobistego do pracy;

4) stosuje przepisy prawa dotyczące gospodarki odpadami niebezpiecznymi;

5) dobiera i wymienia materiały eksploatacyjne urządzeń peryferyjnych komputera osobistego;

6) wykonuje konserwację urządzeń peryferyjnych komputera osobistego;

7) instaluje sterowniki urządzeń peryferyjnych komputera osobistego;

8) konfiguruje urządzenia peryferyjne komputera osobistego.

3. Naprawa komputera osobistego

Uczeń:

1) posługuje się narzędziami do naprawy sprzętu komputerowego;

2) określa kody błędów uruchamiania komputera osobistego;

3) lokalizuje oraz usuwa uszkodzenia sprzętowe podzespołów komputera osobistego;

4) lokalizuje oraz usuwa usterki systemu operacyjnego i aplikacji;

5) lokalizuje uszkodzenia urządzeń peryferyjnych komputera osobistego;

6) sporządza harmonogram prac związanych z lokalizacją i usuwaniem usterek komputera osobistego;

7) dobiera oprogramowanie diagnostyczne i monitorujące pracę komputera osobistego;

8) odzyskuje z komputera osobistego dane użytkownika;

9) tworzy kopie bezpieczeństwa danych;

10) formułuje wskazania dla użytkownika po wykonaniu naprawy komputera osobistego;

11) sporządza kosztorys naprawy komputera osobistego.

E.13. Projektowanie lokalnych sieci komputerowych i administrowanie sieciami

1. Projektowanie i wykonywanie lokalnej sieci komputerowej



Uczeń:

- 1) rozpoznaje topologie lokalnych sieci komputerowych;
- 2) rozpoznaje i stosuje normy dotyczące okablowania strukturalnego;
- 3) rozpoznaje protokoły sieci lokalnych i protokoły dostępu do sieci rozległej;
- 4) rozpoznaje urządzenia sieciowe na podstawie opisu, symboli graficznych i wyglądu;
- 5) określa funkcje komputerowego systemu sieciowego;
- 6) wykonuje projekt lokalnej sieci komputerowej;
- 7) dobiera elementy komputerowej sieci strukturalnej, urządzenia i oprogramowanie sieciowe;
- 8) sporządza kosztorys projektowanej sieci komputerowej;
- 9) dobiera medium do budowy lokalnej sieci komputerowej;
- 10) dobiera przyrządy i urządzenia do montażu okablowania strukturalnego;
- 11) montuje okablowanie sieciowe;
- 12) wykonuje pomiary okablowania strukturalnego;
- 13) opisuje i analizuje klasy adresów IP;
- 14) projektuje strukturę adresów IP w sieci;
- 15) wykonuje pomiary i testy sieci logicznej;
- 16) opracowuje dokumentację powykonawczą lokalnej sieci komputerowej.

2. Konfigurowanie urządzeń sieciowych

Uczeń:

- 1) modernizuje i rekonfiguruje serwery;
- 2) konfiguruje przełączniki lokalnych sieci komputerowych;
- 3) konfiguruje sieci wirtualne w lokalnych sieciach komputerowych;
- 4) konfiguruje routery i urządzenia zabezpieczające typu zaporą sieciową (ang. firewall);
- 5) konfiguruje urządzenia dostępu do lokalnej sieci komputerowej bezprzewodowej;
- 6) konfiguruje urządzenia telefonii internetowej;
- 7) dobiera i stosuje narzędzia diagnostyczne;
- 8) tworzy sieci wirtualne za pomocą połączeń internetowych;
- 9) monitoruje pracę urządzeń lokalnych sieci komputerowych.

3. Administrowanie sieciami systemami operacyjnymi

Uczeń:

- 1) instaluje sieciowe systemy operacyjne;
- 2) konfiguruje interfejsy sieciowe;
- 3) udostępnia zasoby lokalnej sieci komputerowej;
- 4) charakteryzuje usługi serwerowe;
- 5) określa funkcje profili użytkowników i zasady grup użytkowników;
- 6) zarządza kontami użytkowników i grup użytkowników systemu operacyjnego lub komputera;
- 7) konfiguruje usługi katalogowe lokalnej sieci komputerowej;
- 8) zarządza centralnie stacjami roboczymi;
- 9) rozpoznaje protokoły aplikacyjne;
- 10) monitoruje działania użytkowników lokalnej sieci komputerowej;
- 11) modernizuje lokalną sieć komputerową;
- 12) przestrzega zasad udostępniania i ochrony zasobów sieciowych;
- 13) wyjaśnia zasady działania protokołów lokalnej sieci komputerowej;
- 14) konfiguruje usługi odpowiedzialne za adresację hostów (adresację IP), system nazw, ruting, zabezpieczenie przed wszelkiego rodzaju atakami z sieci (firewall);
- 15) podłącza lokalną sieć komputerową do Internetu;
- 16) konfiguruje usługi serwerów internetowych;
- 17) określa rodzaje awarii lub wadliwego działania lokalnej sieci komputerowej;
- 18) lokalizuje i usuwa przyczyny wadliwego działania systemów sieciowych;
- 19) zabezpiecza komputery przed zawirusowaniem, niekontrolowanym przepływem informacji oraz utratą danych.

E.14. Tworzenie aplikacji internetowych i baz danych oraz administrowanie bazami

1. Tworzenie stron internetowych

Uczeń:

- 1) posługuje się hipertekstowymi językami znaczników;
- 2) tworzy strony internetowe za pomocą hipertekstowych języków znaczników;
- 3) tworzy kaskadowe arkusze stylów (CSS);
- 4) wykorzystuje kaskadowe arkusze stylów (CSS) do opisu formy prezentacji strony internetowej;
- 5) rozpoznaje funkcje edytorów spełniających założenia WYSIWYG;



- 6) tworzy strony internetowe za pomocą edytorów spełniających założenia WYSIWYG;
- 7) projektuje strukturę witryny internetowej;
- 8) wykonuje strony internetowe zgodnie z projektami;
- 9) stosuje reguły walidacji stron internetowych;
- 10) testuje i publikuje witryny internetowe;
- 11) stosuje różne modele barw;
- 12) przestrzega zasad cyfrowego zapisu obrazu;
- 13) wykonuje projekt graficzny witryny internetowej;
- 14) tworzy grafikę statyczną i animacje jako elementy stron internetowych;
- 15) zmienia atrybuty obiektów graficznych i modyfikuje obiekty graficzne;
- 16) przetwarza i przygotowuje elementy graficzne, obraz i dźwięk do publikacji w Internecie;
- 17) przestrzega zasad komputerowego przetwarzania obrazu i dźwięku.

2. Tworzenie baz danych i administrowanie bazami danych

Uczeń:

- 1) korzysta z funkcji strukturalnego języka zapytań;
- 2) posługuje się strukturalnym językiem zapytań do obsługi baz danych;
- 3) projektuje i tworzy relacyjne bazy danych;
- 4) importuje dane do bazy danych;
- 5) tworzy formularze, zapytania i raporty do przetwarzania danych;
- 6) instaluje systemy baz danych i systemy zarządzania bazami danych;
- 7) modyfikuje i rozbudowuje struktury baz danych;
- 8) dobiera sposoby ustawiania zabezpieczeń dostępu do danych;
- 9) zarządza bazą danych i jej bezpieczeństwem;
- 10) określa uprawnienia poszczególnych użytkowników i zabezpieczenia dla nich;
- 11) udostępnia zasoby bazy danych w sieci;
- 12) zarządza kopiami zapasowymi baz danych i ich odzyskiwaniem;
- 13) kontroluje spójność baz danych;
- 14) dokonuje naprawy baz danych.

3. Tworzenie aplikacji internetowych

Uczeń:

- 1) korzysta z wbudowanych typów danych;
- 2) tworzy własne typy danych;
- 3) przestrzega zasad programowania;
- 4) stosuje instrukcje, funkcje, procedury, obiekty, metody wybranych języków programowania;
- 5) tworzy własne funkcje, procedury, obiekty, metody wybranych języków programowania;
- 6) wykorzystuje środowisko programistyczne: edytor, kompilator i debugger;
- 7) kompiluje i uruchamia kody źródłowe;
- 8) wykorzystuje języki programowania do tworzenia aplikacji internetowych realizujących zadania po stronie serwera;
- 9) stosuje skrypty wykonywane po stronie klienta przy tworzeniu aplikacji internetowych;
- 10) wykorzystuje frameworki do tworzenia własnych aplikacji;
- 11) pobiera dane aplikacji i przechowuje je w bazach danych;
- 12) testuje tworzoną aplikację i modyfikuje jej kod źródłowy;
- 13) dokumentuje tworzoną aplikację;
- 14) zamieszcza opracowane aplikacje w Internecie;
- 15) zabezpiecza dostęp do tworzonych aplikacji.

Warunki realizacji kształcenia w zawodzie

Szkoła podejmująca kształcenie w zawodzie technik informatyk powinna posiadać następujące pomieszczenia dydaktyczne:

- pracownię urządzeń techniki komputerowej, wyposażoną w: stanowiska komputerowe (jedno stanowisko dla jednego ucznia); podzespoły umożliwiające montaż komputera osobistego; dodatkowe elementy komputera osobistego umożliwiające jego rekonfigurację; oprogramowanie do wirtualizacji; różne systemy operacyjne stacji roboczej; oprogramowanie narzędziowe, diagnostyczne i zabezpieczające; drukarkę laserową, atramentową, igłową; skaner, ploter, tablicę interaktywną, palmtop PDA (Personal Digital Assistant), tablet, projektor multimedialny, klawiaturę i mysz bezprzewodową, czytnik kart podpisu elektronicznego; adapter Bluetooth; stół monterski z matą i opaską antystatyczną; zestaw urządzeń monterskich; podłączenie do sieci lokalnej z dostępem do Internetu;
- pracownię lokalnych sieci komputerowych, wyposażoną w: stanowiska komputerowe (jedno stanowisko dla jednego ucznia); szafę dystrybucyjną 19" z wyposażeniem, połączoną korytkową instalacją okablowania strukturalnego z czterema punktami elektryczno-logicznymi; serwer stelażowy z kontrolerem pamięci masowej; zasilacz awaryjny z zasilaniem; napęd taśmowy do

archiwizacji; komputer typu notebook z obsługą lokalnej sieci bezprzewodowej; przełącznik zarządzany z obsługą lokalnych sieci wirtualnych i portami zasilania przez Ethernet; koncentrator xDSL z obsługą protokołu PPP; ruter z modemem xDSL, z portem Ethernet i obsługą protokołu PPP, oprogramowanie typu firewall z obsługą wirtualnych sieci prywatnych; punkt dostępu do lokalnej sieci bezprzewodowej z różnego typu antenami zewnętrznymi i portem zasilania przez Ethernet; telefon internetowy; tester okablowania; reflektometr; różne sieciowe systemy operacyjne przeznaczone dla serwera; oprogramowanie do wirtualizacji; oprogramowanie komputerowego wspomaganie projektowania (Computer Aided Design) z biblioteką elementów sieci lokalnej; oprogramowanie do monitorowania pracy sieci; stół monterski z matą i opaską antystatyczną; zestaw narzędzi monterskich; podłączenie do sieci lokalnej z dostępem do Internetu;

- pracownię sieciowych systemów operacyjnych, wyposażoną w: stanowiska komputerowe (jedno stanowisko dla jednego ucznia); laptop lub notebook dla nauczyciela z oprogramowaniem do wirtualizacji; różne serwerowe systemy operacyjne z usługami katalogowymi i internetowymi; serwerowe oprogramowanie typu firewall; oprogramowanie do analizy protokołów sieciowych; oprogramowanie do monitorowania pracy sieci; podłączenie do sieci lokalnej z dostępem do Internetu;
- pracownię aplikacji internetowych, wyposażoną w: stanowiska komputerowe (jedno stanowisko dla jednego ucznia); laptop lub notebook dla nauczyciela z oprogramowaniem do wirtualizacji; edytor WYSIWYG stron internetowych z możliwością edycji hipertekstowego języka znaczników i kaskadowych arkuszy stylów, z możliwością walidacji strony; oprogramowanie do tworzenia grafiki i animacji, obróbki materiałów audio i wideo; oprogramowanie serwera relacyjnej bazy danych z programami narzędziowymi; oprogramowanie umożliwiające tworzenie aplikacji internetowych po stronie serwera i klienta w wybranych językach programowania; podłączenie do sieci lokalnej z dostępem do Internetu; dostęp do serwera umożliwiającego publikację stron www i aplikacji internetowych; dostęp do portalu wspierającego pracę grupową, komunikację, publikację wiadomości i materiałów.

W szkole prowadzącej kształcenie w zawodzie technik informatyk językiem obcym ukierunkowanym zawodowo jest język angielski.

Kształcenie praktyczne może odbywać się w: pracowniach szkolnych, placówkach kształcenia ustawicznego, placówkach kształcenia praktycznego oraz podmiotach stanowiących potencjalne miejsca zatrudnienia absolwentów szkół kształcących w zawodzie.

Szkoła organizuje praktyki zawodowe w podmiocie zapewniającym rzeczywiste warunki pracy właściwe dla nauczanego zawodu w wymiarze 4 tygodni (160 godzin).

Minimalna liczba godzin kształcenia zawodowego

Efekty kształcenia wspólne dla wszystkich zawodów oraz efekty kształcenia wspólne dla zawodów w ramach obszaru elektryczno-elektronicznego stanowiące podbudowę do kształcenia w zawodzie lub grupie zawodów - 270 godz.

E.12. Montaż i eksploatacja komputerów osobistych oraz urządzeń peryferyjnych -360 godz.

E.13. Projektowanie lokalnych sieci komputerowych i administrowanie sieciami - 300 godz.

E.14. Tworzenie aplikacji internetowych i baz danych oraz administrowanie bazami - 420 godz.

W szkole liczbę godzin kształcenia zawodowego należy dostosować do wymiaru godzin określonego w przepisach w sprawie ramowych planów nauczania w szkołach publicznych, przewidzianego dla kształcenia zawodowego, zachowując, z wyjątkiem szkoły policealnej dla dorosłych, minimalną liczbę godzin wskazanych w tabeli odpowiednio dla efektów kształcenia: wspólnych dla wszystkich zawodów i wspólnych dla zawodów w ramach obszaru kształcenia stanowiących podbudowę do kształcenia w zawodzie lub grupie zawodów oraz właściwych dla kwalifikacji wyodrębnionych w zawodzie.

Możliwości uzyskiwania dodatkowych kwalifikacji w zawodach w ramach obszaru kształcenia określonego w klasyfikacji zawodów szkolnictwa zawodowego

Absolwent szkoły kształcącej w zawodzie technik informatyk po potwierdzeniu kwalifikacji

E.12. Montaż i eksploatacja komputerów osobistych oraz urządzeń peryferyjnych,

E.13. Projektowanie lokalnych sieci komputerowych i administrowanie sieciami

E.14. Tworzenie aplikacji internetowych i baz danych oraz administrowanie bazami może uzyskać dyplom potwierdzający kwalifikacje w zawodzie technik teleinformatyk, po potwierdzeniu dodatkowo kwalifikacji

E.15. Uruchamianie oraz utrzymanie terminali i przyłączy abonenckich

E.16. Montaż i eksploatacja sieci rozległych.

Moduł szkoleniowy

JAVA

Wprowadzenie

Powstanie języka Java miało być remedium na niedoskonałości języka C++, a zarazem krzywa nauczania dla osób przechodzących z tego języka miała być łagodna. W osiągnięciu tego śmiałego celu pomóc twórcom miały następujące założenia:

1. **Obiektowość** – w Javie podstawową jednostką programów jest obiekt. Posiada on swój stan (wartości pól) i zachowanie (metody, które mogą zmieniać jego stan). Java jest silnie ukierunkowana obiektowo, w przeciwieństwie to C++ gdzie możliwe jest programowanie proceduralno-obiektowe.
2. **Dziedziczenie** – w C++ możliwe było dziedziczenie wielobazowe, w Javie jednak projektanci zdecydowali, że należy uprościć ten model i pozwolić jedynie na dziedziczenie z jednego obiektu. Decyzja ta usunęła niedoskonałość języka C++ polegającą na powstawaniu konfliktów w przypadku, gdy dwie klasy nadrzędne posiadały składowe wzajemnie się wykluczające. Java wprowadziła nową konstrukcję zwaną interfejsem, która to miała na celu przywrócić elastyczność utraconą poprzez pojedyncze dziedziczenie. Kolejną istotną zmianą jest to, iż w Javie każda klasa posiada klasę rodzicielską. U korzenia dziedziczenia klas zawsze znajduje się klasa podstawowa `Object`.
3. **Niezależność od architektury** – jest to jedna z największych przewag Javy nad innymi językami programowania. Kod źródłowy kompiluje się tylko raz, niezależnie od architektury docelowej, na której będzie on uruchamiany. Kompilacja powoduje powstanie bytecode'u, który z kolei uruchamiany jest w maszynie wirtualnej. Podejście to tworząc dodatkową abstrakcyjną warstwę uwolniło programy od konieczności dostosowania się do danej architektury, jednak wprowadziło spadek wydajności (było to zauważalne, zwłaszcza w pierwszych wersjach Javy)
4. **Bezpieczeństwo i niezawodność** – w Javie wprowadzono tak zwany system wyjątków. Pozwala on na obsługę nieprzewidywalnych sytuacji w programie, takich jak stracone połączenie do pliku czy próba odczytu indeksu tablicy z poza zakresu tej tablicy. Dzięki umiejętnemu posługiwaniu się tym mechanizmem można obsłużyć takie nietypowe sytuacje, a program może podjąć kolejną próbę wykonania danego zadania lub też kontynuować swoje wykonanie. W języku C++ często takie zdarzenie prowadziło do zamknięcia działania całego programu.
5. **Sieciowość i obsługa programowania rozproszonego** – w Javie od samego początku wbudowane są obiekty pozwalające na komunikację zdalną (socket, http, ftp), wywoływanie metod czy przesyłanie całych obiektów między dwoma programami Javy działającymi nawet na odrębnych maszynach (RMI). Java może być uruchamiana w przeglądarkach internetowych pod postacią apletów, jak i działać po stronie serwera.

Po latach, kiedy język dobrze się przyjął i ma rzeszę programistów można śmiało stwierdzić, iż podstawowy cel zastąpienia C++ nie został osiągnięty i raczej nigdy nie zostanie. Prostota języka jest kwestią dyskusyjną, jest on mniej elastyczny niż C++, zarazem chroniąc programistę przed popełnianiem prostych błędów. W chwili obecnej należy traktować Javę, jako w pełni wykształcony odrębny język programowania, który swoją popularności ustępuje jedynie całej rodzinie języków z rodziny C razem wziętych (C, C++, Objective C).

Cele

Poznanie zastosowania i zalet języka programowania Java. Umiejętność wykorzystania podstawowych możliwości języka. Praktyczne wykorzystanie elementów środowiska pracy. Podstawowa umiejętność projektowania aplikacji zorientowanej obiektowo. Tworzenie wieloplatformowych aplikacji z interfejsem graficznym. Zwiększenie efektywności pracy poprzez umiejętność korzystania i tworzenia z dokumentacji.

Opis sposobu realizacji celów

10 półtoragodzinnych lekcji składających się z treści teoretycznych wraz z ćwiczeniami. Praktyczne uwagi dla osób początkujących dotyczące środowiska pracy Eclipse. Po zakończeniu całego cyklu - przeprowadzenie egzaminu sprawdzającego wiedzę.

Dodatkowo wprowadzono lekcję 11 zawierającą część projektu interdyscyplinarnego łączącego 4 dziedziny wiedzy z zakresu informatyki (SQL, PHP, JavaScript oraz JAVA) w zakresie stworzenia aplikacji mobilnej na platformę Android, wykorzystującej API REST z aplikacji napisanej w PHP.

Treści kształcenia

Treść kursu została podzielona na 10 bloków tematycznych – po jednym do każdej lekcji:

1. Wstęp do Javy i środowiska pracy,
2. Typy podstawowe i instrukcje sterujące,
3. Wstęp do programowania obiektowego,
4. Interfejsy, klasy abstrakcyjne i dziedziczenie,
5. Polimorfizm, instrukcja „final”, klasy wewnętrzne i anonimowe,
6. Typ wyliczeniowy, JavaDocs,
7. Tablice jednowymiarowe i wielowymiarowe, kolekcje,
8. Operacje wejścia – wyjścia i wyjątki,
9. Graficzny interfejs użytkownika - Swing,
10. Przechwytywanie zdarzeń, tworzenie plików wykonywalnych.
11. Projekt interdyscyplinarny – aplikacja mobilna dla systemu Android umożliwiająca przeglądanie bazy kolegów i koleżanek ze szkoły

Opis założonych osiągnięć ucznia

Uczeń po odbyciu kursu pozna teoretyczne podstawy języka Java, będzie umiał zaprojektować prostą aplikację zorientowaną obiektowo z interfejsem graficznym.

Korelacja z treściami podstawy programowej

Domyślnym podmiotem celów kształcenia jest „Uczeń”.

Cele kształcenia	Treść kształcenia	Podstawa programowa
<ul style="list-style-type: none"> - zna podstawowe typy danych - umie przechowywać obiekty w tablicach, lub kolekcjach - potrafi wyjaśnić pojęcie listy - umie używać klas zdefiniowanych w frameworku Swing 	Lekcje: 2, 7, 8	Uczeń korzysta z wbudowanych typów danych
<ul style="list-style-type: none"> - poznaje ideę programowania obiektowego - umie budować klasy, klasy abstrakcyjne, interfejsy w Javie - umie tworzyć konstruktory i metody - poznaje klasy wewnętrzne - rozumie specyfikę typu wyliczeniowego - tworzy własne wyjątki 	Lekcje: 3, 4, 5, 6, 8	Uczeń tworzy własne typy danych
<ul style="list-style-type: none"> - potrafi bezpiecznie przechowywać dane w zmiennych - poznaje ideę programowania obiektowego - umie efektywnie tworzyć aplikacje - umie zapewnić bezpieczeństwo integralności aplikacji poprzez stosowanie dziedziczenia - rozumie pojęcie polimorfizmu 	Lekcje: 2, 3, 4, 5	Uczeń przestrzega zasad programowania
<ul style="list-style-type: none"> - poznaje pętle i instrukcje sterujące - umie zarządzać przebiegiem wykonania programu - umie przechowywać obiekty w tablicach, lub kolekcjach - potrafi wyjaśnić pojęcie listy - umie wykorzystywać widżety z biblioteki Swing 	Lekcje: 2, 7, 8	Uczeń stosuje instrukcje, funkcje, procedury, obiekty, metody wybranych języków programowania
<ul style="list-style-type: none"> - umie budować klasy, klasy abstrakcyjne, interfejsy w Javie - umie tworzyć konstruktory i metody - poznaje klasy wewnętrzne 	Lekcje: 3, 4, 5	Uczeń tworzy własne funkcje, procedury, obiekty, metody wybranych języków programowania
<ul style="list-style-type: none"> - poznaje specyfikę języka Java, działanie, wady i zalety - umie poprawnie skonfigurować środowisko pracy 	Lekcje: 1	Uczeń wykorzystuje środowisko programistyczne: edytor, kompilator, debugger
<ul style="list-style-type: none"> - uruchamia i buduje programy - umie budować pliki wykonywalne jar 	Lekcje: 1, 10	Uczeń kompiluje i uruchamia kody źródłowe
<ul style="list-style-type: none"> - poznaje środowiska, w których Java jest wykorzystywana 	Lekcje: 1 oraz moduł dotyczący języka PHP	Uczeń wykorzystuje języki programowania do tworzenia aplikacji internetowych realizujących zadania po stronie serwera

Cele kształcenia	Treść kształcenia	Podstawa programowa
- poznaje możliwości zastosowania Javy w wykonywaniu programu po stronie klienta	Lekcje: 1 oraz moduł dotyczący języka JavaScript	Uczeń stosuje skrypty wykonywane po stronie klienta przy tworzeniu aplikacji internetowych
- zna możliwości graficznego frameworku Swing - umie obsługiwać zdarzenia w frameworku Swing	Lekcje: 9, 10	Uczeń wykorzystuje frameworki do tworzenia własnych aplikacji
- umie wymieniać dane z zasobami z i poza programu (sieć, pliki lokalne). - rozumie, kiedy używać plikowej bazy danych	Lekcje: 8, 9 oraz moduł dotyczący SQL	Uczeń pobiera dane aplikacji i przechowuje je w bazie danych
- umie edytować kod źródłowy aplikacji	Lekcje: 2, 3, 4, 5, 6, 7, 8, 9, 10	Uczeń testuje tworzoną aplikację i modyfikuje jej kod źródłowy
- umie dokumentować kod komentarzami - poznaje metodę dokumentowania kodu za pomocą JavaDocs	Lekcje: 1, 6	Uczeń dokumentuje tworzoną aplikację
- poznaje sposób integracji Javy na stronach internetowych	Lekcje: 1	Uczeń zamieszcza opracowane aplikacje w Internecie
- wie jak zabezpieczyć integralności aplikacji poprzez właściwe użycie typów widoczności - umie zabezpieczać aplikację przed błędami	Lekcje: 4, 8	Uczeń zabezpiecza dostęp do stworzonych aplikacji

Sposoby osiągnięcia celów

Po odbyciu każdej lekcji uczeń powinien samodzielnie wykonać proste ćwiczenia w celu dokładnego zrozumienia omawianych mechanizmów. Programowanie w języku Java bez odpowiedniego środowiska jest bardzo trudne, dlatego jednym z celów jest jego poznanie. Umożliwi to wykonywanie ćwiczeń w czasie efektywnym.

Lista czynności w zależności od roli

Czynności nauczyciela	Czynności ucznia
Prezentowanie przygotowanego materiału.	Sporządzanie notatek z prezentowanego materiału
Zadawanie pytań kontrolnych odnośnie już pokazanego materiału	Udzielanie odpowiedzi na pytania nauczyciela
Zadawanie pytań odnośnie materiału, który dopiero będzie pokazany, jeśli możliwe jest uzyskanie odpowiedzi w drodze dedukcji	
Objaśnianie ćwiczeń lekcyjnym, pilnowanie czasu ich wykonania przez uczniów, udzielanie wskazówek.	Rozwiązywanie zadań lekcyjnych
Objaśnienie zadań na koniec lekcji, udzielanie wskazówek podczas rozwiązywania, zlecenie	Odrabianie zadań domowych

dokończenia zadań, jako pracę domową.	
Sprawdzanie ćwiczeń lekcyjnych i zadań domowych: należy zwrócić uwagę na poprawne formatowanie kodu (wcięcia) poprawne użycie konstrukcji języka, dokumentowanie kodu komentarzami, otrzymywanie poprawnych rezultatów.	
Zapisywanie uwag odnośnie materiałów szkoleniowych i sugestie ewentualnych korekt.	

Propozycje kryteriów oceny i metod sprawdzania osiągnięć ucznia

Ocena końcowa powinna uwzględniać wiele aspektów aktywności ucznia. Podlegać ocenie powinny:

1. ćwiczenia wykonywane podczas lekcji
2. odpowiedzi na pytania nauczyciela
3. zadania domowe
4. aktywność podczas lekcji
5. ćwiczenia sprawdzające

W celu umożliwienia wiarygodnej oceny ucznia materiał szkoleniowy zawiera wiele przykładowych ćwiczeń lekcyjnych, pytań nauczyciela do uczniów, ćwiczeń na zakończenie każdej lekcji, których dokończenie może być zadaniem domowym oraz propozycją końcowego testu sprawdzającego wiedzę teoretyczną.

Wymagania na poszczególne oceny szkolne

Określone wymagania uwzględniają jedynie znajomość ucznia z zakresu programowania w języku Java. Należy je traktować, jako składową oceny końcowej, uwzględniającej inne kryteria.

Domyślny podmiot to uczeń:

Zagadnienie	2	3	4	5	6
Prymitywne typy danych	umie rozróżnić rodzaje: łańcuch znaków (tekst), liczba całkowita, liczby zmiennoprzecinkowe, logiczna	umie wymienić i zadeklarować większość (może zapomnieć do 3 typów numerycznych)	umie wymienić i zadeklarować zmienne wszystkich typów prymitywnych	rozumie kiedy i jak używać typów zmiennoprzecinkowych	zna zakresy dla każdego typu prymitywnego
Przechowywanie obiektów w tablicach, lub kolekcjach	umie wyjaśnić różnicę między tablicą a kolekcją	umie poprawnie zadeklarować tablicę	umie przepisać elementy z tablicy do listy i odwrotnie	rozumie tablice wielowymiarowe, umie poprawnie je zadeklarować	uczeń umie posortować tablicę lub kolekcję
Framework Swing	wie, co to jest i do czego służy	umie czytając przykładowy kod źródłowy wytłumaczyć każdą linię oraz opisać, co zostanie wyświetlone wie, że można obsłużyć zdarzenia myszki i klawiatury	umie sam napisać prostą aplikację okienkową. W szczególności wie jak poprawnie zaimplementować wyłączenie i inicjalizację. umie obsługiwać zdarzenia myszki i klawiatury wie, co to jest manager	umie użyć managera layoutu i uzasadnić wybór takiego a nie innego w zależności od zaistniałym sytuacji	umie napisać własny manager layoutu

Zagadnienie	2	3	4	5	6
			layoutu		
Paradygmat programowania obiektowego	umie wytłumaczyć, co to jest obiekt, podać prosty przykład modelu przedmiotu z świata rzeczywistego np. samochód z kołami, z felgami i radiem	umie rozróżnić pojęcie klasy od obiektu zna składowe klasy (metody i pola) umie zaimplementować prostą klasę w osobnym pliku wie co to jest dziedziczenie i z ilu klas można naraz dziedziczyć	wie, co to są typy widoczności umie zaimplementować klasę zawierającą, jako pole inną klasę umie stworzyć klasę, która dziedziczy z innej klasy umie wyjaśnić, co to jest i do czego służy słowo „this”	umie wyjaśnić widoczność pól i metod w aspekcie dziedziczenia umie wywoływać metody z klasy rodzica (super)	umie tworzyć konstruktory umie wywoływać konstruktor rodzica umie wywoływać z jednego konstruktora, drugi konstruktor tej samej klasy
Pojęcie interfejs, klasa abstrakcyjna, klasa anonimowa	umie wytłumaczyć każdy z terminów	wie jak stworzyć interfejs i klasę abstrakcyjną	umie zaimplementować 2 i więcej interfejsów w jednej nowej klasie umie dziedziczyć po klasie abstrakcyjnej	umie stworzyć obiekt klasy anonimowej z interfejsu oraz klasy abstrakcyjnej	wie, że klasa abstrakcyjna może dziedziczyć z innej klasy (również abstrakcyjnej) wie że interfejsy mogą dziedziczyć po innych interfejsach
Klasy wewnętrzne	umie powiedzieć czy jest możliwe zdefiniowanie klasy wewnątrz innej	umie zdefiniować klasę wewnątrz innej klasy	wie, jaki typ widoczności może posiadać klasa wewnętrzna	umie stworzyć obiekt klasy wewnętrznej w dowolnej innej części	umie odwołać się do pola klasy zewnętrznej, które ma taką samą nazwę jak

Zagadnienie	2	3	4	5	6
	klasy			programu	pole w klasie wewnętrznej
Typ wyliczeniowy	wie że istnieje coś takiego i kiedy powinno się go użyć	umie zaimplementować przykładowy typ wyliczeniowy			umie stworzyć pole wewnątrz typu wyliczeniowego rozumie, iż jest to specyficzny typ klasy, dla którego nie można stworzyć obiektu za pomocą słowa new
Wyjątki	wie, po co są wyjątki wie, że można je łapać	umie rzucić wyjątek i go złapać	umie stworzyć własny wyjątek	umie stworzyć własny wyjątek zawierający dodatkowe pole i łapać ten wyjątek użyć tej dodatkowej informacji	potrafi rozróżnić RuntimeException od innych podklas Exception
Polimorfizm		wie, co to jest i potrafi przytoczyć przykłady z życia codziennego	umie użyć polimorfizmu w programie potrafi sprawdzić czy dany obiekt jest typu bardziej szczegółowego	wie jak działa nadpisywanie metod i która wersja metody zostanie wywołana w zależności o typu referencji do obiektu	
Pętle i instrukcje warunkowe	potrafi wyjaśnić, po co są pętle.	umie napisać poprawną pętlę for oraz while	umie napisać poprawne zagnieżdżone instrukcje	zna i umie użyć trójargumentowego	Potrafi zamienić prostą pętlę na rekurencyjne

Zagadnienie	2	3	4	5	6
	potrafi podać przykład instrukcji warunkowej rozpoznaje słowa kluczowe	umie napisać poprawny warunek if/else	if/else umie napisać poprawną pętlę do-while oraz for w wersji foreach (bez trzech członów)	operatora warunkowego	wywołanie metody
Specyfika języka Java	wie, że Java to język programowania wie, że uruchamia się niezależnie od platformy, gdyż wykonuje się w maszynie wirtualnej potrafi podać kilka przykładów środowisk, w jakich można spotkać aplikacje napisane w Javie	wie, że Java kompiluje się do bytecode'u wie jak uruchomić program w Javie z linii komend rozdzieli pojęcia JRE oraz SDK	wie jak skompilować program w Javie zna rozszerzenia plików kodu źródłowego, jak i plików po kompilacji wie jak uruchomić program z pliku JAR w javie wie, co to są pakiety i jak importować klasy	wie, że każda klasa czy interfejs po kompilacji łączy się w osobnym pliku umie stworzyć plik JAR w elipsie	umie stworzyć plik JAR w linii komend umie uruchomić aplikację składającą się z kilku osobnych plików JAR
Obsługa wejścia/wyjścia	wie, że Java potrafi czytać i zapisywać pliki na dysku lokalnym wie, że Java umie komunikować się przez sieć	umie napisać program czytający lub zapisujący plik	Poprawnie obsługuje możliwe wyjątki I/O	umie napisać program łączy się z serwerem	umie napisać program czytający plik binarny, wyszukujący zadany ciąg bitów, zamieniający ten ciąg na inny i zapisujący wynik w nowym pliku
Środowisko programistyczne	wie, że pliki źródłowe są zwykłymi plikami tekstowymi	umie stworzyć nowy projekt w Eclipse umie uruchomić aplikację	korzysta z znanych skrótów klawiaturowych: podpowiadanie składni,	korzysta z bardziej zaawansowanych funkcji Eclipse:	umie używać debugera

Zagadnienie	2	3	4	5	6
		w Eclipse	kopiuj, wklej	refactoring/zmiana nazwy zmiennej/metody/klasy, automatyczne czyszczenie importów	
Dokumentacja kodu	potrafi wyjaśnić, po co dokumentuje się kod potrafi podać przykład komentarza	rozdziela trzy typy komentarzy: JavaDoc, blokowy i do końca linii używa komentarzy w swoim kodzie	potrafi napisać poprawny komentarz JavaDoc używa JavaDoc w swoim kodzie	potrafi użyć anotacji w JavaDoc	potrafi wygenerować dokumentację HTML z kodu aplikacji



Końcowy test teoretyczny

Ocena końcowa powinna zależeć od wyniku testu sprawdzającego wiadomości teoretyczne (20 pytań zamkniętych pojedynczego wyboru - 20 punktów).

Propozycje kryteriów oceny:

1. 0-9 punktów - ocena 1,
2. 10-12 punktów - ocena 2,
3. 13-15 punktów - ocena 3,
4. 16-18 punktów - ocena 4,
5. 19-20 punktów - ocena 5, Test końcowy sprawdzający wiedzę

20 zadań na 30 minutowy test sprawdzający wiedzę. Pogrubioną czcionką zaznaczono prawidłowe odpowiedzi.

1. Java kompiluje się do:
 1. Kodu maszynowego
 2. **Kodu bajtowego**
 3. Pliku skryptowego
2. Kompilator Javy znajduje się w pakiecie:
 1. **JDK**
 2. JRE
 3. Eclipse
3. Aby przechować wartość 100 jakiego najmniejszego typu możemy użyć:
 1. long
 2. int
 3. **short**
4. Która pętla wykona jeden krok przebiegu nawet w przypadku fałszywego warunku:
 1. for
 2. **do-while**
 3. while
5. Słowo „new” tworzy:
 1. nową klasę;
 2. **nowy obiekt**
 3. nowy typ
6. Czy kiedy w klasie występuje konstruktor „public MojaKlasa(int x) „, mogę stworzyć obiekt za pomocą instrukcji „new MojaKlasa()”?
 1. tak
 2. nie
 3. **tylko wtedy, gdy istnieje drugi konstruktor „public MojaKlasa()”**
7. Która ze struktur nie posiada implementacji metod:
 1. klasa
 2. **interfejs**
 3. klasa niewłaściwa
8. Jeżeli chcemy, aby dana metoda była widoczna tylko w bieżącym pakiecie, to jakiego typu widoczności użyjemy?

1. private
2. public
3. **nie podajemy typu, co oznacza użycie widoczności package (default)**
9. Które zdanie jest nieprawdziwe:
 1. Każdy obiekt typu Integer jest również obiektem typu Object
 2. Mogę używać obiektu typu Integer tak, jak każdego innego obiektu typu Object
 3. **Mogę stworzyć obiekt, który jest jednocześnie typu Integer i String.**
10. Zaznacz prawidłowe stwierdzenie
 1. **Mogę stworzyć instancję interfejsu, pod warunkiem, że zaimplementuję jego metody**
 2. Jedyną możliwością stworzenia instancji interfejsu jest użycie słowa „implements”
 3. Mogę stworzyć instancję interfejsu tak, jak zwykłej klasy.
11. Pisanie JavaDoc rozpoczynamy od znaku:
 1. **/****
 2. **//**
 3. **@**
12. Podpowiadanie składni w środowisku Eclipse wywołujemy skrótem:
 1. **„Ctrl” + „Shift” + „O”**
 2. **„Ctrl” + „Spacja”**
 3. **„F5”**
13. Jeżeli chcemy udostępnić innym programistom wybór jednego z 5 słów, powinniśmy:
 1. **Skorzystać z typu wyliczeniowego - enum**
 2. Umieścić słowa w tablicy
 3. Poprosić o liczbę z przedziału od 1 do 5
14. Jeżeli chcemy wstawiać liczby w różne miejsca zbioru, powinniśmy skorzystać z:
 1. tablicy
 2. typu wyliczeniowego
 3. **listy**
15. Odczytując plik używamy obiektu:
 1. System.in
 2. OutputStream
 3. **InputStream**
16. Aby stworzyć własny wyjątek należy dziedziczyć po klasie:
 1. **Exception**
 2. Error
 3. StackTrace
17. Aby umożliwić użytkownikowi wybór kilku opcji w interfejsie graficznym, skorzystamy z:
 1. JComboBox
 2. **JCheckBox**
 3. JRadioButton
18. Aby stworzyć obiekt reagujący na zdarzenia myszy, należy zaimplementować interfejs:
 1. LayoutManager
 2. MouseEvent
 3. **MouseListener**

19. Zarządca rozkładu w interfejsie graficznym to w przypadku Swinga:

1. **Layout**
2. Listener
3. JPanel

20. Plik wykonywalny JAR to:

1. **Archiwum ZIP**
2. Plik z pojedynczym kodem bajtowym
3. Kod maszynowy

Lekcje

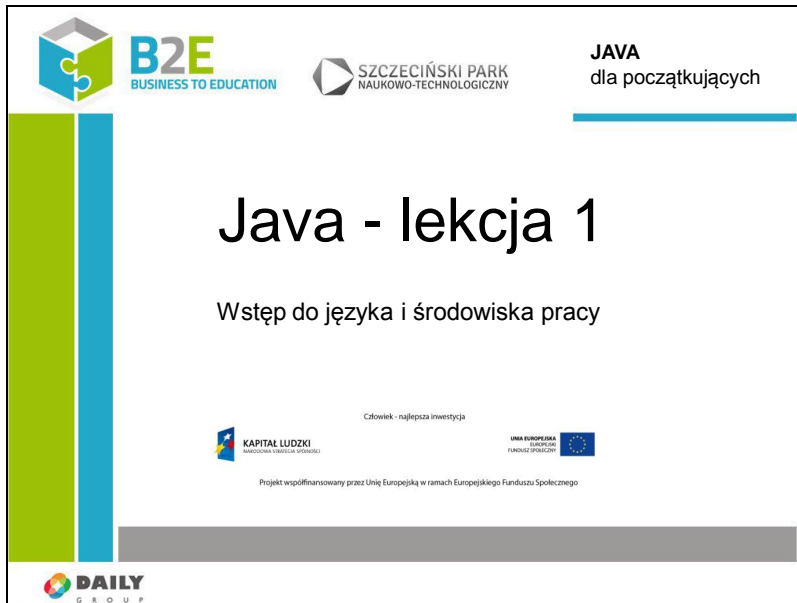
Lekcja 1 Wstęp do Javy i środowiska pracy

Cel lekcji

Celem lekcji jest poznanie, czym jest język programowania Java, w jaki sposób działa i jakie są jego zalety. Wytłumaczone jest jak poprawnie skonfigurować środowisko pracy i co jest potrzebne, aby móc uruchamiać zbudowane programy.

Treść - slajdy z opisem

Slajd 1



Slajd 1

B2E
BUSINESS TO EDUCATION

SZCZECIŃSKI PARK
NAUKOWO-TECHNOLOGICZNY

JAVA
dla początkujących

Java - lekcja 1

Wstęp do języka i środowiska pracy

Człowiek - najlepsza inwestycja

KAPITAŁ LUDZKI
NARODOWA STRATEGIA SPÓJNOŚCI

UNIA EUROPEJSKA
EUROPEJSKI
FUNDUSZ SPOŁECZNY

Projekt współfinansowany przez Unię Europejską w ramach Europejskiego Funduszu Społecznego

DAILY
GROUP



Slajd 2



B2E
BUSINESS TO EDUCATION



SZCZECIŃSKI PARK
NAUKOWO-TECHNOLOGICZNY

JAVA
dla początkujących

Główne koncepcje:

- Obiektość
- Niezależność od architektury
- Sieciowość i obsługa programowania rozproszonego
- Niezawodność i bezpieczeństwo







DAILY
GROUP

Wstęp do języka i środowiska pracy

Java jest językiem kompilowanym do kodu bajtowego (bytecode). Oznacza to, że aplikacje napisane w Javie i skompilowane nie mogą zostać uruchomione bezpośrednio przez system operacyjny. Niezbędna jest maszyna wirtualna javy (JVM), która interpretuje i wykonuje kod bajtowy.

Java została stworzona przez grupę roboczą pod kierunkiem Jamesa Goslinga z firmy Sun Microsystems, obecnym właścicielem tej technologii jest Oracle Corporation.

Jego podstawowe koncepcje zostały przejęte z języka Smalltalk (maszyna wirtualna, zarządzanie pamięcią) oraz z języka C++ (duża część składni i słów kluczowych).

Główne koncepcje

Autorzy języka Java określili kilkanaście kluczowych koncepcji swojego języka. Najważniejsze z nich to:

Obiektość

W przeciwieństwie do proceduralno-obiektowego języka C++, Java jest silnie ukierunkowana na obiektość. O obiekcie można myśleć jako o samoistnej części programu, która może przyjmować określone stany i ma określone zachowania, które mogą zmieniać te stany bądź przysyłać dane do innych obiektów. Wyjątkiem od całkowitej obiektości są typy proste (inaczej: typy wbudowane, prymitywy).

Dziedziczenie

W Javie wszystkie obiekty są pochodną obiektu nadrzędnego (jego klasa nazywa się po prostu Object), z którego dziedziczą podstawowe zachowania i właściwości. Dzięki temu wszystkie mają wspólny podzbiór podstawowych możliwości, takich jak ich: identyfikacja, porównywanie, kopiowanie, niszczenie czy wsparcie dla programowania współbieżnego.

Niezależność od architektury

Tę właściwość Java ma dzięki temu, że kod źródłowy programów pisanych w Javie kompiluje się do kodu pośredniego (kodu bajtowego). Powstały kod jest niezależny od systemu operacyjnego i procesora, a wykonuje go tzw. wirtualna maszyna Javy, która (między innymi) tłumaczy kod uniwersalny na kod dostosowany do specyfiki konkretnego

systemu operacyjnego i procesora. W tej chwili wirtualna maszyna Javy jest już dostępna dla większości systemów operacyjnych i procesorów. Jednak z uwagi na to, że kod pośredni jest interpretowany, taki program jest wolniejszy niż kompilowany do kodu maszynowego.

Sieciowość i obsługa programowania rozproszonego

Dzięki wykorzystaniu reguł obiektowości, Java nie widzi różnicy między danymi płynącymi z pliku lokalnego a danymi z pliku dostępnego przez HTTP czy FTP.

Niezawodność i bezpieczeństwo

W zamierzeniu Java miała zastąpić C++ – obiektowego następcę języka C. Jej projektanci zaczęli od rozpoznania cech języka C++, które są przyczyną największej liczby błędów programistycznych, by stworzyć język prosty w użyciu, bezpieczny i niezawodny. O ile po siedmiu odsłonach Javy jej prostota jest dyskusyjna, o tyle język faktycznie robi dużo, by utrudnić programiście popełnienie błędu.

Slajd 3

B2E
BUSINESS TO EDUCATION

SZCZECIŃSKI PARK
NAUKOWO-TECHNOLOGICZNY

JAVA
dla początkujących

Możliwości zastosowania:

Typy aplikacji:

- desktop (wiersza poleceń i okienkowe)
- serwerowe (strony WWW)
- mobilne (telefony,)
- wbudowane (centrale telefoniczne, telewizory, dekodery telewizyjne, pralki, piekarniki, karty SIM...)

Wirtualna maszyna Javy – jako uniwersalne środowisko uruchomieniowe

ORACLE®

DAILY
GROUP

Wstęp do języka i środowiska pracy

Z uwagi na swoją popularność Java dorobiła się ogromnej ilości bibliotek i projektów. Dzięki takiej popularności można ją spotkać praktycznie w każdym środowisku.

Aplikacje – jako pełnoprawny język programowania Java umożliwia tworzenie standardowych aplikacji uruchamianych przez użytkownika. Mogą to być aplikacje linii komend lub też okienkowe. Środowisko okienkowe tworzymy korzystając z frameworka Swing lub zyskującej coraz większą popularność JavvyFX.

WWW – w dzisiejszych czasach Java głównie dostarcza zaplecza serwerowego dla stron WWW. Pozwala na tworzenie bezpiecznych i skalowalnych aplikacji działających po stronie serwera. Frontend często tworzony przy pomocy innych technologii komunikuje się z stroną serwerową aby uzyskać lub przekazać potrzebne dane. Obecnie mniej modnym i zarazem mniej dynamicznym rozwiązaniem jest generowanie po stronie serwera całych stron HTML. W tym przypadku możemy mówić o tym iż Java dostarcza również frontend, jednak interakcja z aplikacją powoduje zazwyczaj przeładowanie strony. Istnieją jeszcze technologie oparte na Javie, które pozwalają na częściowe

renderowanie zawartości strony WWW w przeglądarce np.. Java Server Faces (JSF), jednak po stronie przeglądarki nie znajdziemy ani odrobiny Javy jako takiej. W JSF komponenty generują kod HTML (widok) oraz JavaScript odpowiedzialny za interakcję z użytkownikiem, komunikację z serwerem i uaktualnienie widoku.

Aplet – jest to program napisany w Javie, przeglądarka korzystając z wtyczki maszyny wirtualnej uruchamia go w oknie przeglądarki. Obecnie technologia uznana za przestarzałą, jednak czasem jeszcze spotykana w specyficznych rozwiązaniach. Zazwyczaj używana gdy strona WWW potrzebuje dostępu do zasobów lokalnych np.. czytnika kart podpisu cyfrowego lub też operacji na plikach np.. policzenie hash'a z pliku wideo i automatyczne znalezienie pasujących napisów w serwisie.

Aplikacje mobilne – tutaj znów możemy wyróżnić dwa podstawowe warianty

Jako aplikacje Java ME (Java mobile edition) – jest to zubożona wersja Javy, dostosowana do słabych obliczeniowo i z małą ilością pamięci urządzeń, takich jak poprzednie generacje telefonów komórkowych, dekodery telewizyjne, karty SIM czy chipy w kartach bankowych. Nie zawiera wszystkich bibliotek, jednak sama składnia języka jest taka sama.

Jako aplikacje na ostatnio bardzo popularną platformę Android. Java jest językiem wybranym przez twórców platformy, jednak kompiluje się do niestandardowej formy bytecode'u, interpretowanej przez maszynę wirtualną Dalvik.

JVM – sama maszyna wirtualna okazała się być dobrą, szybką i niezawodną platformą uruchomieniową. W związku z tym powstały inne języki programowania kompilujące się do kodu bajtowego. Oto kilka bardziej popularnych:

- a. Clojure, dialekt języka funkcyjnego Lisp
- b. Groovy, język skryptowo-programistyczny
- c. Scala, programowanie funkcyjno-objektowe
- d. JRuby, implementacja Ruby
- e. Jython, implementacja Python'a



Slajd 4



The slide features a header with three logos: a stylized cube logo, 'B2E BUSINESS TO EDUCATION', and 'SZCZECIŃSKI PARK NAUKOWO-TECHNOLOGICZNY'. The title 'JAVA dla początkujących' is positioned in the top right. The main text lists resources for setting up a Java environment: 'www.oracle.com', 'Java JRE (Java Runtime Environment)', 'Java JDK (Java Development Kit)', and 'http://www.eclipse.org/downloads/Eclipse Classic'. A footer includes the 'DAILY GROUP' logo and the text 'Wstęp do języka i środowiska pracy'.

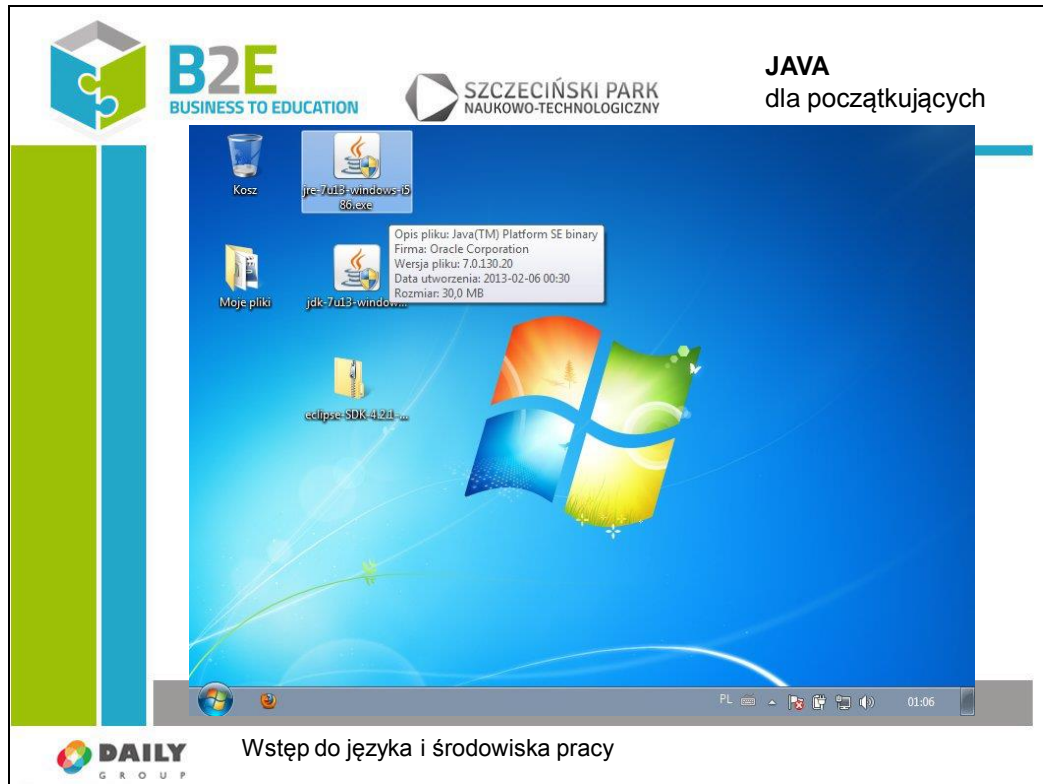
Aby móc uruchomić aplikację napisaną w Javie musimy mieć maszynę wirtualną. Znajduje się ona w pakiecie Java JRE (Java Runtime Environment). Można go łatwo wyszukać korzystając z wyszukiwarki Google lub pobrać bezpośrednio ze strony www.oracle.com. Aby móc stworzyć aplikację napisaną w Javie, musimy mieć większy zestaw narzędzi o nazwie Java JDK (Java Development Kit). Znajduje się tam przede wszystkim kompilator. Cały pakiet dostępny jest również na stronie www.oracle.com.

Należy przede wszystkim pamiętać, że Java stworzona jest na licencji GNU (General Public License). Na slajdzie przedstawione są jedynie oficjalne narzędzia. Każdy może napisać własną maszynę wirtualną i istnieje wiele alternatywnych źródeł. Np. otwarta implementacja Javy - OpenJDK.

Bardzo ważnym elementem jest również środowisko pracy. Stanowczo odradzam korzystanie z prostych edytorów tekstowych. Jednym z najpopularniejszych środowisk jest Eclipse. Można pobrać również środowisko NetBeans lub IntelliJ IDEA. Wszystkie trzy oferują podobne możliwości i są powszechnie wykorzystywane na całym świecie. Zalecam pobranie pakietu Eclipse Classic, ponieważ właśnie w nim będę pokazywał przykłady.

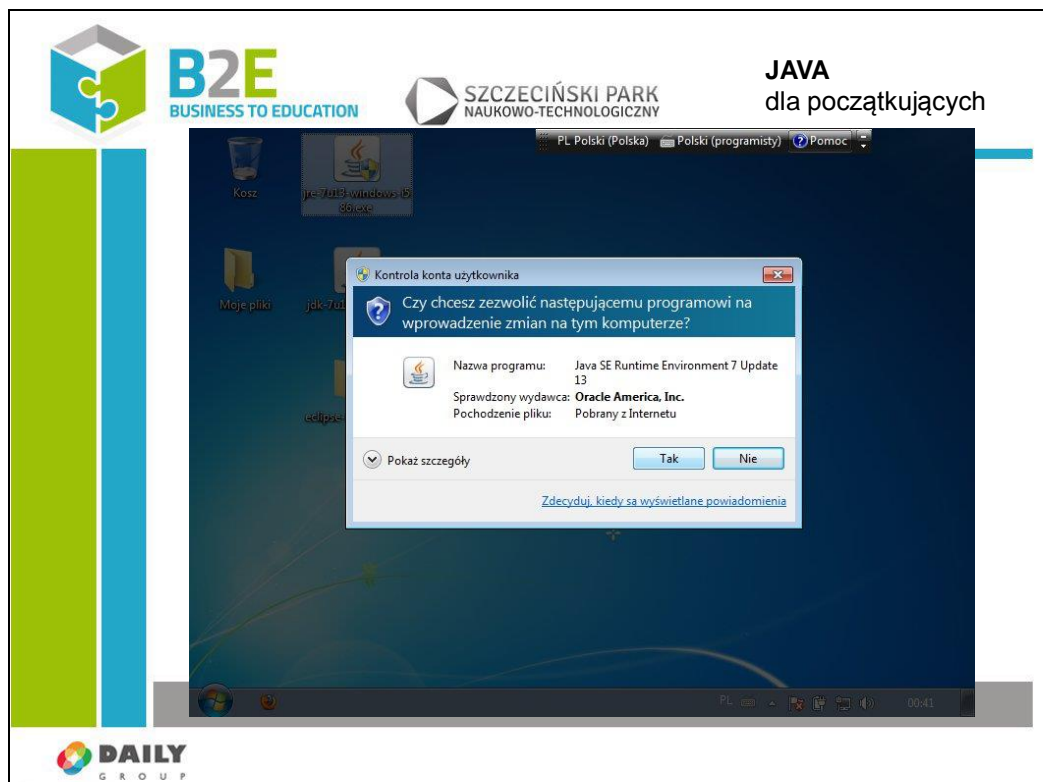


Slajd 5

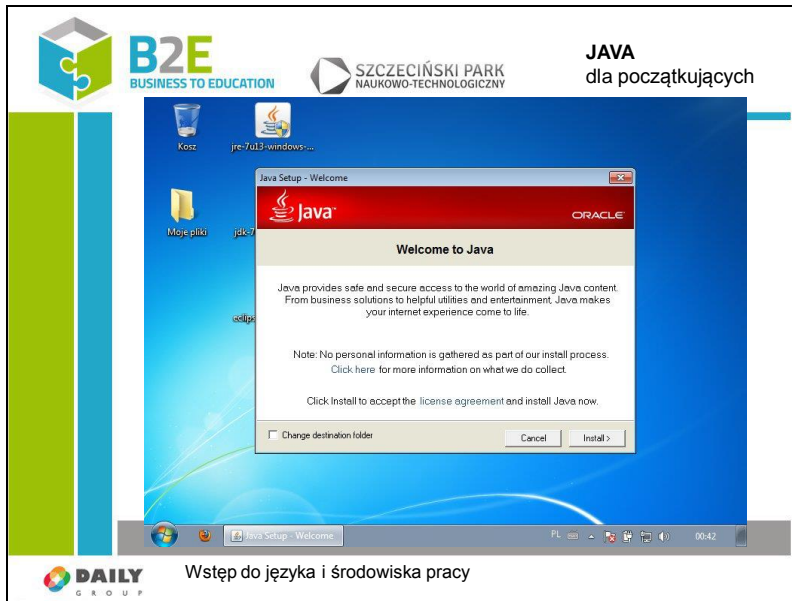


W pierwszej kolejności instalujemy pakiet Java JRE

Slajd 6

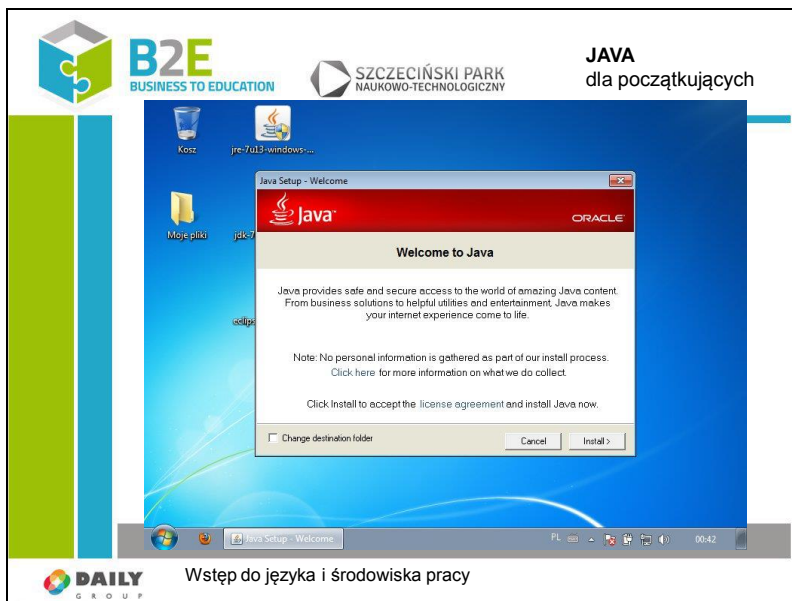


Slajd 7



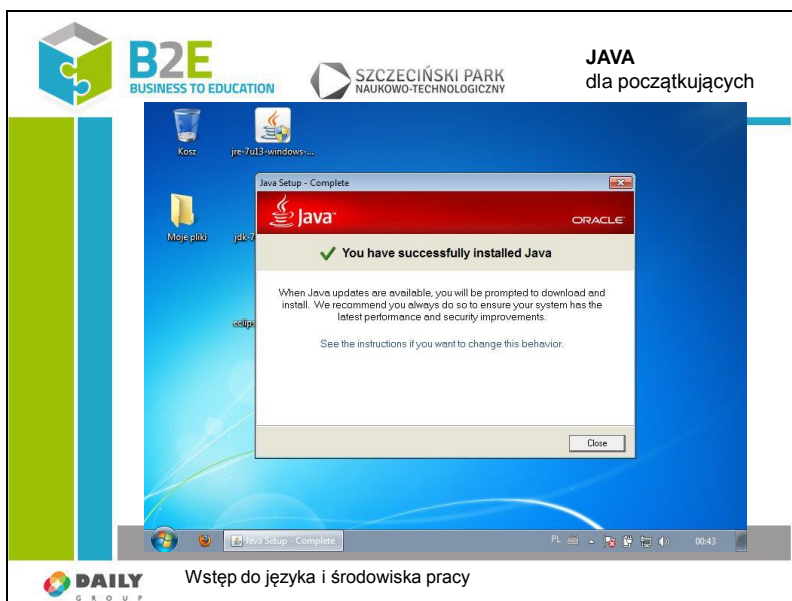
Przechodzimy
kolejne kroki
instalacji.

Slajd 8

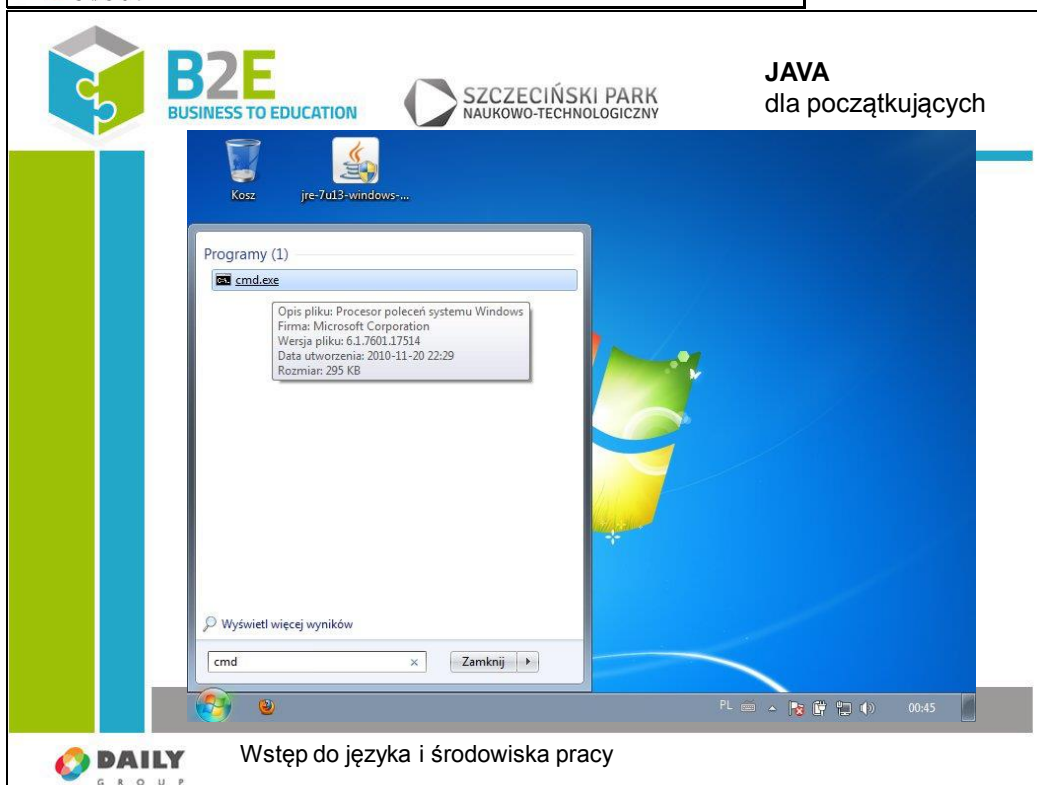




Slajd 9



Slajd 10



Po zakończeniu instalacji możemy korzystać z oprogramowania stworzonego w języku Java.

W dalszych slajdach pokazane będzie jak uruchomić aplikację. Dla zaprezentowania posiadam już gotowy plik wykonywalny. Proszę nie wykonywać tej części samodzielnie. W tym celu należy uruchomić program cmd.exe.



Slajd 11



B2E
BUSINESS TO EDUCATION



SZCZECIŃSKI PARK
NAUKOWO-TECHNOLOGICZNY

JAVA
dla początkujących

```
C:\Windows\system32\cmd.exe
C:\tmp\szkolenie\lekcja01\przyklad1>dir *.class
Wolumin w stacji C nie ma etykiety.
Numer seryjny woluminu: 665F-1429

Katalog: C:\tmp\szkolenie\lekcja01\przyklad1
2013-02-18 23:28          419 MojaKlasa.class
                1 plik(ów)          419 bajtów
                0 katalog(ów) 8 644 915 200 bajtów wolnych
C:\tmp\szkolenie\lekcja01\przyklad1>
```

 **DAILY**
GROUP

Wstęp do języka i środowiska pracy

Pliki wykonywalne dla maszyny wirtualnej Javy posiadają rozszerzenie class.

Slajd 12



B2E
BUSINESS TO EDUCATION



SZCZECIŃSKI PARK
NAUKOWO-TECHNOLOGICZNY

JAVA
dla początkujących

```
C:\Windows\system32\cmd.exe
C:\tmp\szkolenie\lekcja01\przyklad1>java MojaKlasa
```

 **DAILY**
GROUP

Wstęp do języka i środowiska pracy

Poleceniem java uruchamiamy maszynę podając jako parametr nazwę pliku bez rozszerzenia.



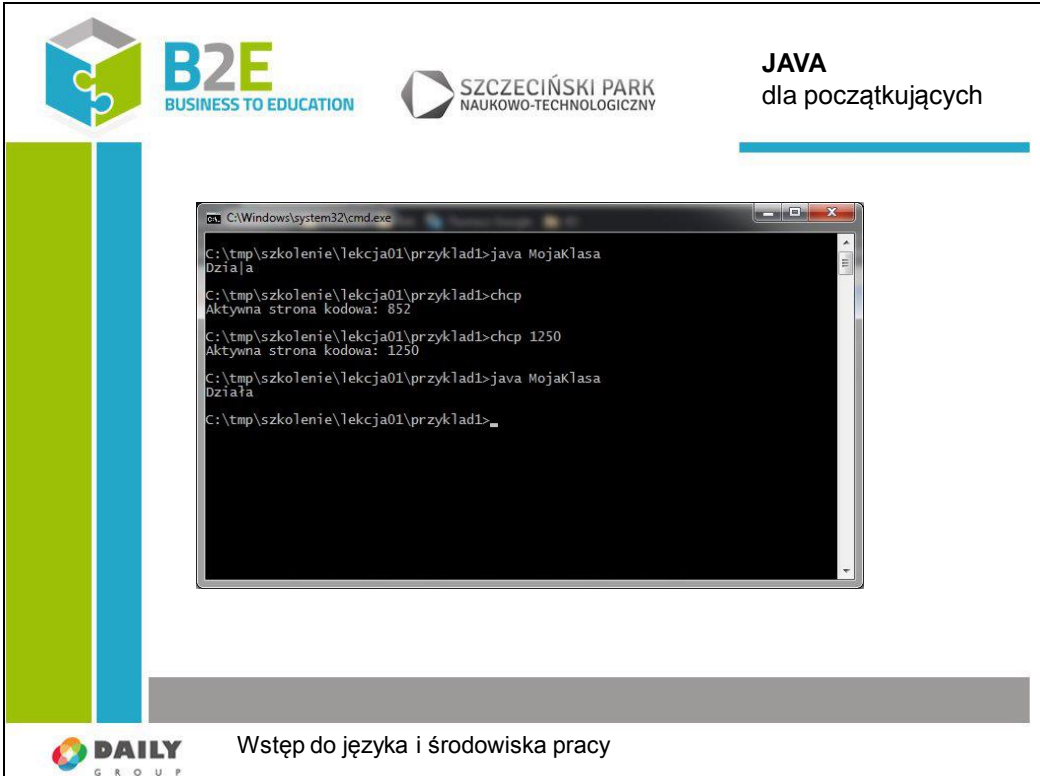
Slajd 13



Slide 13 features a header with logos for B2E (Business to Education), Szczeciński Park Naukowo-Technologiczny, and the title "JAVA dla początkujących". A central screenshot of a Windows command prompt shows the command `java MojaKlasa` being executed, resulting in the output `Działa`. The slide footer includes the DAILY GROUP logo and the text "Wstęp do języka i środowiska pracy".

W wyniku otrzymaliśmy napis „Działa!”. Nie wyświetliła się jednak literka „ł”. Spowodowane jest to kodowaniem znaków w programie cmd.exe. Sama Java nie ma problemu z kodowaniem znaków, gdyż domyślnie korzysta z 16 bitów na symbol, więc bez żadnych ograniczeń można stosować znaki regionalne.

Slajd 14



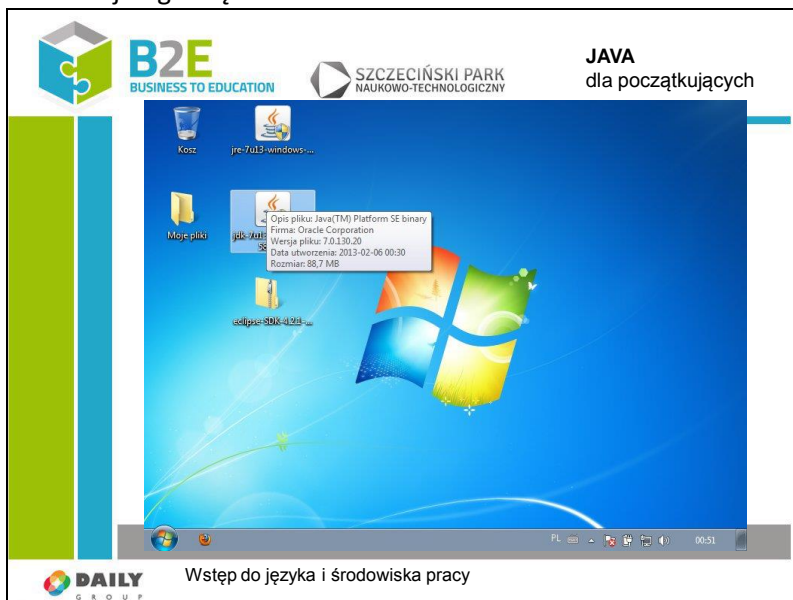
Slide 14 features the same header as slide 13. The central screenshot of the Windows command prompt shows a sequence of commands and outputs: `java MojaKlasa` outputs `Działa`; `chcp` outputs `Aktywna strona kodowa: 852`; `chcp 1250` outputs `Aktywna strona kodowa: 1250`; and `java MojaKlasa` outputs `Działa`. The slide footer includes the DAILY GROUP logo and the text "Wstęp do języka i środowiska pracy".

Problem kodowania polega na tym iż Java w systemach Windows domyślnie wypisuje



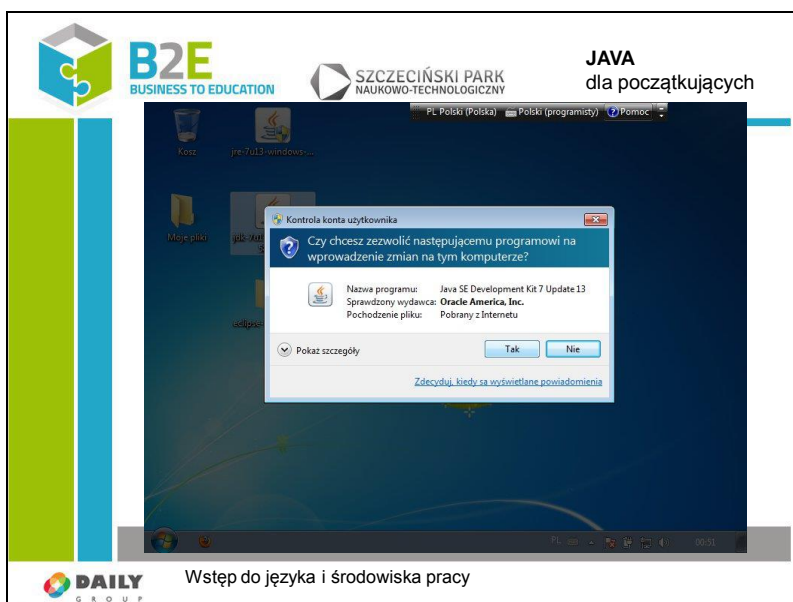
używając kodowania CP-1250. Niestety konsola domyślnie jest skonfigurowana do używania kodowania CP-852. Zmieniając ustawienia konsoli można pozbyć się wcześniejszego błędu.

Slajd 15



Aby móc zacząć programować w Javie, potrzebujemy kompilatora z pakietu Java JDK.

Slajd 16



Kolejno wykonujemy polecenia instalatora.



Projekt współfinansowany przez Unię Europejską w ramach Europejskiego Funduszu Społecznego

Slajd 17

JAVA
dla początkujących

Wstęp do języka i środowiska pracy

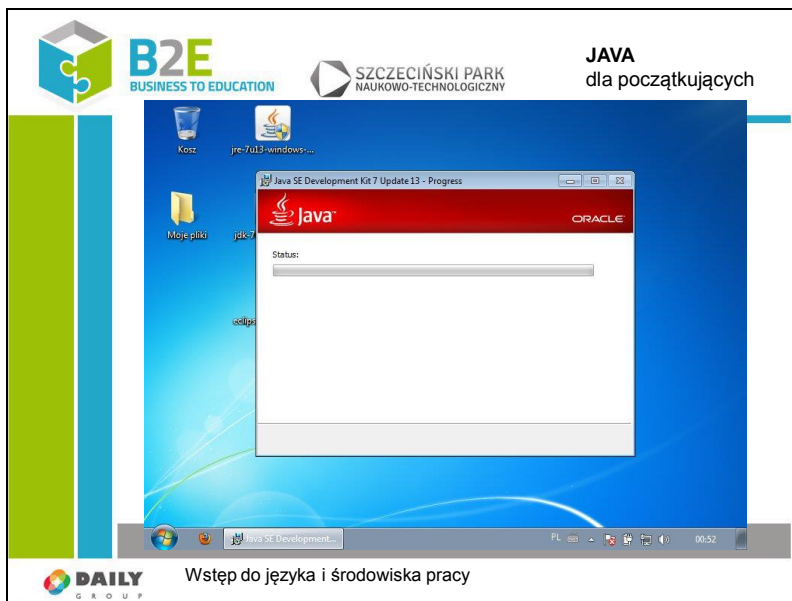
Slajd 18

JAVA
dla początkujących

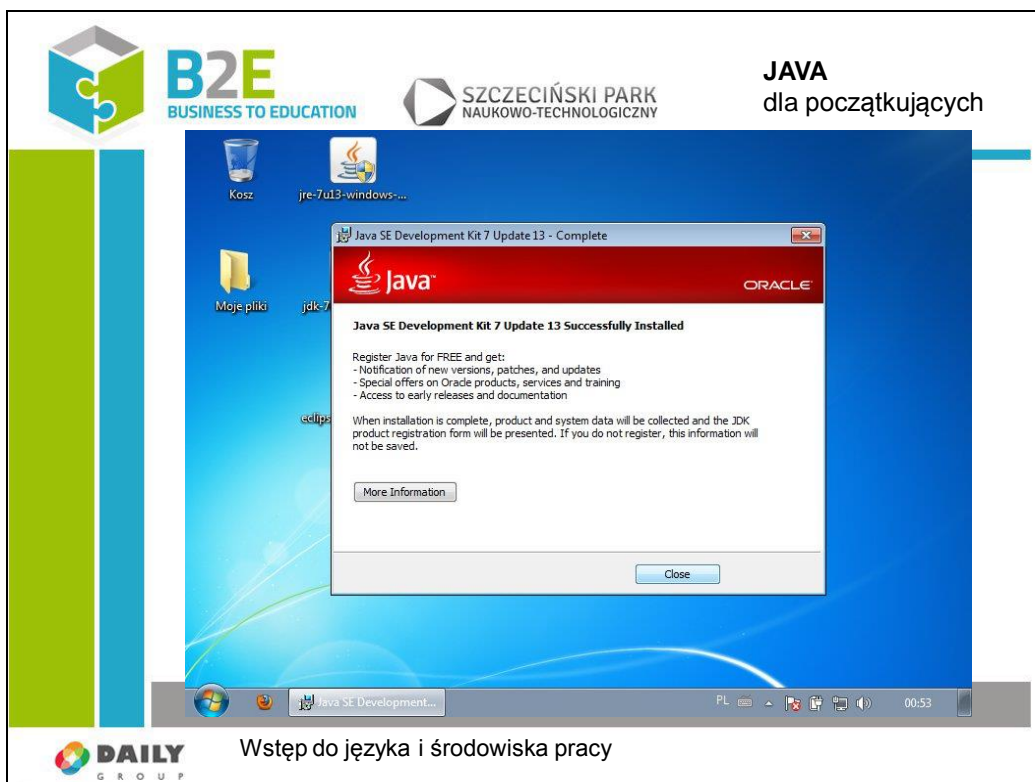
Wstęp do języka i środowiska pracy



Slajd 19




Slajd 20





Slajd 21





SZCZECIŃSKI PARK
NAUKOWO-TECHNOLOGICZNY

JAVA
dla początkujących






Wstęp do języka i środowiska pracy


Po zakończonej instalacji Java JDK. Możemy napisać pierwszy program.

Możemy już wykonać pierwsze praktyczne ćwiczenie.

Stwórzmy plik o dowolnej nazwie, bez białych znaków i zaczynający się dużą literą, z rozszerzeniem *.java.

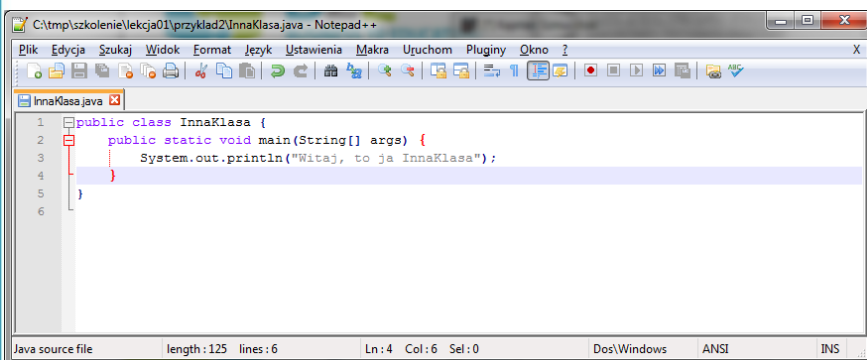
Slajd 22






SZCZECIŃSKI PARK
NAUKOWO-TECHNOLOGICZNY

JAVA
dla początkujących





Wstęp do języka i środowiska pracy

Do pliku wstawiamy ten fragment kodu.



Uwaga!

Jeżeli użyta została inna nazwa pliku niż „InnaKlasa” to należy ją wpisać w pierwszym wierszu (bez rozszerzenia) zamiast wyrażenia „InnaKlasa”.

Slajd 23

B2E
BUSINESS TO EDUCATION

SZCZECIŃSKI PARK
NAUKOWO-TECHNOLOGICZNY

JAVA
dla początkujących

```
C:\Windows\system32\cmd.exe  
C:\tmp\szkolenie\lekcja01\przyklad2>javac InnaKlasa.java
```

DAILY
GROUP

Wstęp do języka i środowiska pracy

Kompilujemy nasz kod źródłowy widocznym poleceniem.

Pierwsza część jest ścieżką do kompilatora Javy – javac.exe.

Argumentem przyjmowanym przez kompilator jest ścieżka do pliku z kodem źródłowym.



Slajd 24

B2E BUSINESS TO EDUCATION

SZCZECIŃSKI PARK
NAUKOWO-TECHNOLOGICZNY

JAVA
dla początkujących

```
C:\Windows\system32\cmd.exe
C:\tmp\szkolenie\lekcja01\przyklad2>javac InnaKlasa.java
C:\tmp\szkolenie\lekcja01\przyklad2>
```

DAILY GROUP Wstęp do języka i środowiska pracy

Tak wygląda rezultat poprawnie wykonanego polecenia kompilacji (brak jakichkolwiek wiadomości i ponowne wyświetlenie znaku zachęty).

Slajd 25

B2E BUSINESS TO EDUCATION

SZCZECIŃSKI PARK
NAUKOWO-TECHNOLOGICZNY

JAVA
dla początkujących

Nazwa	Data modyfikacji	Typ	Rozmiar
InnaKlasa.class	2013-02-18 23:54	Plik CLASS	1 KB
InnaKlasa.java	2013-02-18 23:49	Plik JAVA	1 KB

InnaKlasa.java Data modyfikacji: 2013-02-18 23:49 Data utworzenia: 2013-02-18 23:16
Plik JAVA Rozmiar: 125 bajtów

DAILY GROUP Wstęp do języka i środowiska pracy


Jedyną widoczną zmianą jest pojawienie się pliku o takiej samej nazwie jaką wybraliśmy do pliku z rozszerzeniem „*.java”.

Nowy plik posiada rozszerzenie „*.class”. Zawiera on kod bajtowy przeznaczony do



uruchamiania na każdej maszynie wirtualnej.

Slajd 26





SZCZECIŃSKI PARK
NAUKOWO-TECHNOLOGICZNY


JAVA
dla początkujących



```

1  KtjInnaKlasa: InnaKlasa
2  InnaKlasa: InnaKlasa()
3  InnaKlasa: InnaKlasa(InnaKlasa)
4  InnaKlasa: InnaKlasa(InnaKlasa, String)
5  InnaKlasa: InnaKlasa(InnaKlasa, String, String)
6  InnaKlasa: InnaKlasa(InnaKlasa, String, String, String)
7  InnaKlasa: InnaKlasa(InnaKlasa, String, String, String, String)
8  InnaKlasa: InnaKlasa(InnaKlasa, String, String, String, String, String)

```



Wstęp do języka i środowiska pracy

Możemy wyświetlić zawartość nowego pliku w edytorze tekstowym.

Jak widać nie jest to kod maszynowy. Wiele części można odczytać, np. „SourceFile” jest to plik źródłowy z którego powstał dany kod bajtowy, „java/lang/String” jest to adres obiektu, który przechowuje ciąg znaków.

Slajd 27





SZCZECIŃSKI PARK
NAUKOWO-TECHNOLOGICZNY

JAVA
dla początkujących



```

C:\tmp\szkolenie\lekcja01\przyklad2>javac InnaKlasa.java
C:\tmp\szkolenie\lekcja01\przyklad2>java InnaKlasa
Witaj, to ja InnaKlasa
C:\tmp\szkolenie\lekcja01\przyklad2>

```

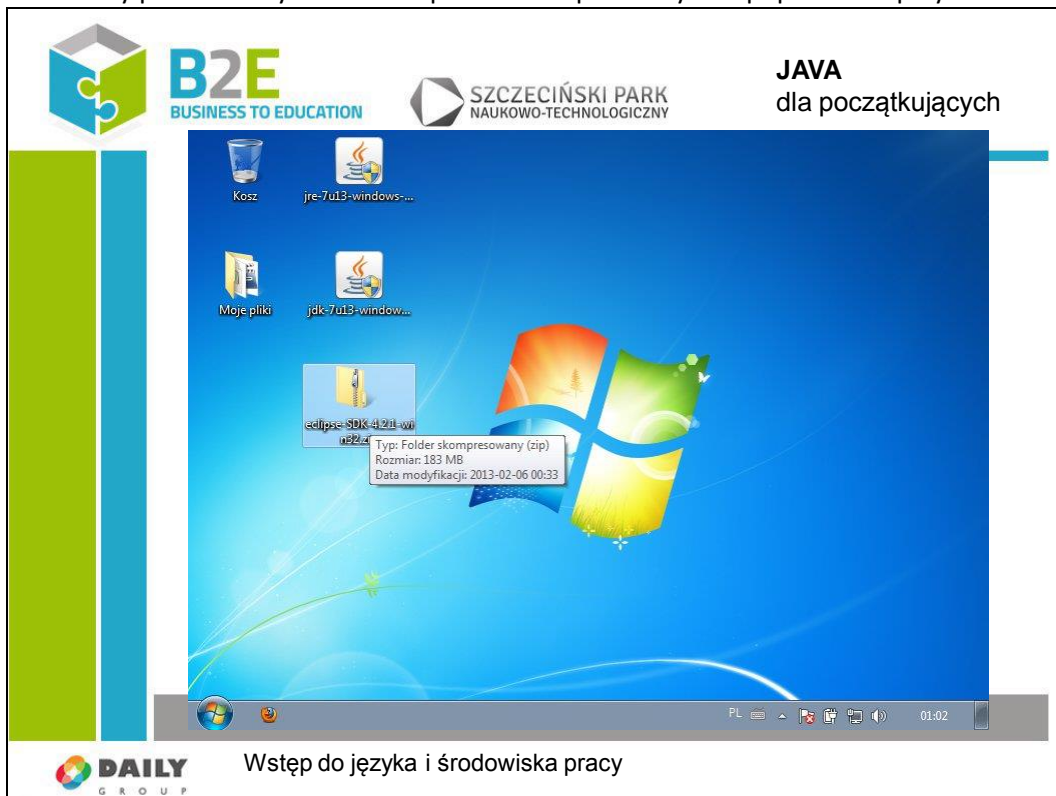


Wstęp do języka i środowiska pracy



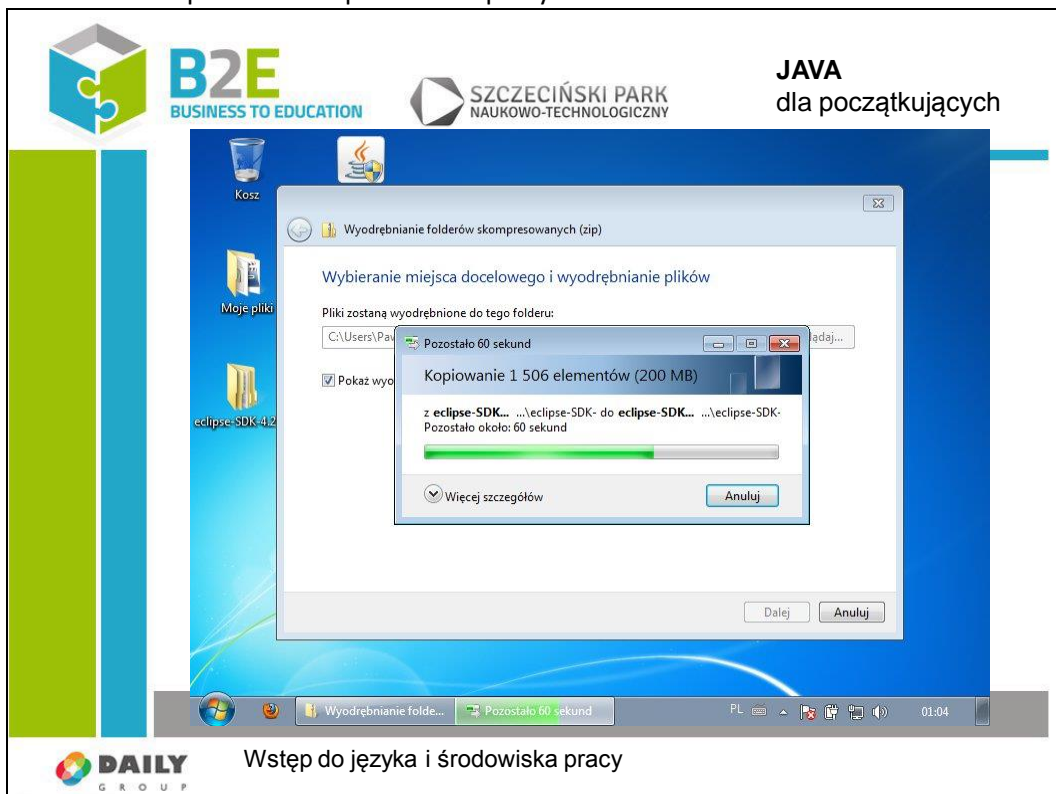
Slajd 28

Omawiany plik możemy uruchomić poleceniem pokazanym w poprzednim przykładzie.



Pisanie kodu w prostym edytorze tekstowym jest trudne. Będziemy korzystać ze środowiska Eclipse w celu usprawnienia pracy.

Slajd 29



Wypakowujemy pobrane archiwum.



Slajd 30

B2E BUSINESS TO EDUCATION **SZCZECIŃSKI PARK NAUKOWO-TECHNOLOGICZNY** **JAVA dla początkujących**

Nazwa	Data modyfikacji	Typ	Rozmiar
configuration	2013-02-06 01:05	Folder plików	
dropins	2012-09-14 20:48	Folder plików	
features	2013-02-06 01:03	Folder plików	
p2	2013-02-06 01:03	Folder plików	
plugins	2013-02-06 01:05	Folder plików	
readme	2013-02-06 01:03	Folder plików	
.eclipseproduct	2013-02-06 01:03	Plik ECLIPSEPROD...	1 KB
artifacts.xml	2013-02-06 01:03	Dokument XML	110 KB
eclipse.exe	2013-02-06 01:03	Aplikacja	312 KB
eclipse.ini	2013-02-06 01:03	Ustawienia konfig...	1 KB
eclipsesec.exe	2013-02-06 01:03	Aplikacja	24 KB
epl-v10.html	2013-02-06 01:03	Firefox HTML Doc...	17 KB
notice.html	2013-02-06 01:03	Firefox HTML Doc...	9 KB

DAILY GROUP Wstęp do języka i środowiska pracy

Środowisko uruchamiamy za pomocą pliku „eclipse.exe”.

Slajd 31

B2E BUSINESS TO EDUCATION **SZCZECIŃSKI PARK NAUKOWO-TECHNOLOGICZNY** **JAVA dla początkujących**

Workspace Launcher

Select a workspace

Eclipse SDK stores your projects in a folder called a workspace. Choose a workspace folder to use for this session.

Workspace: C:\tmp\workspace_szkolenie

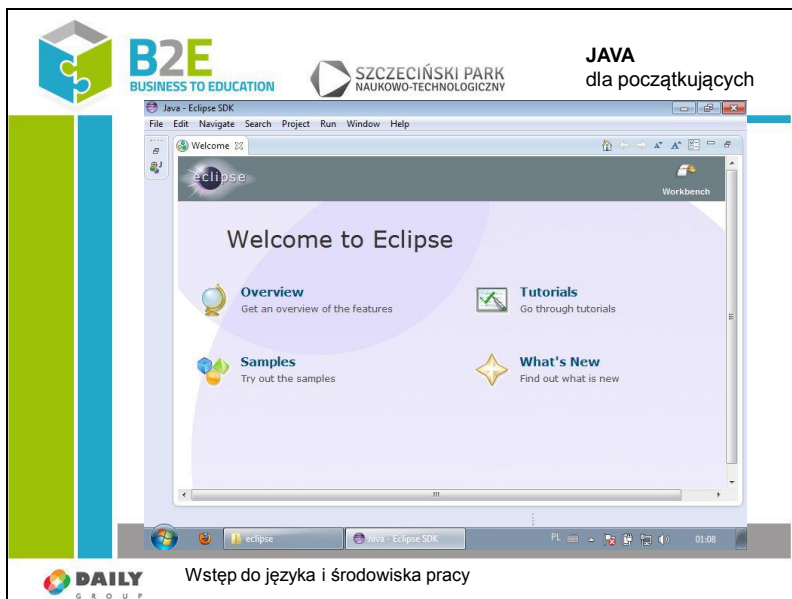
Use this as the default and do not ask again

DAILY GROUP Wstęp do języka i środowiska pracy

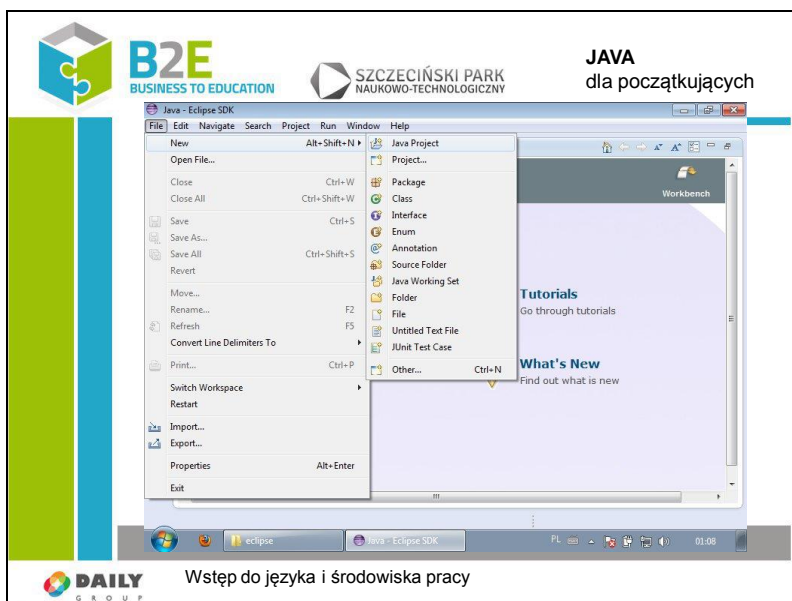
Przy pierwszym uruchomieniu trzeba wybrać folder pełniący funkcję „workspace”. Jest to lokalizacja w której znajdować się będą wszystkie nasze projekty



Slajd 32



Slajd 33



Zanim przystąpimy do pisania kodu, musimy stworzyć projekt.

W tym celu wybieramy:
File/New/Java Project



Slajd 34

B2E BUSINESS TO EDUCATION **SZCZECIŃSKI PARK** NAUKOWO-TECHNOLOGICZNY **JAVA** dla początkujących

DAILY GROUP Wstęp do języka i środowiska pracy

Nazwa projektu (Project name) może być dowolna.
Pozostałe parametry najlepiej pozostawić z ustawieniami domyślnymi.
Klikamy „Next”.

Slajd 35

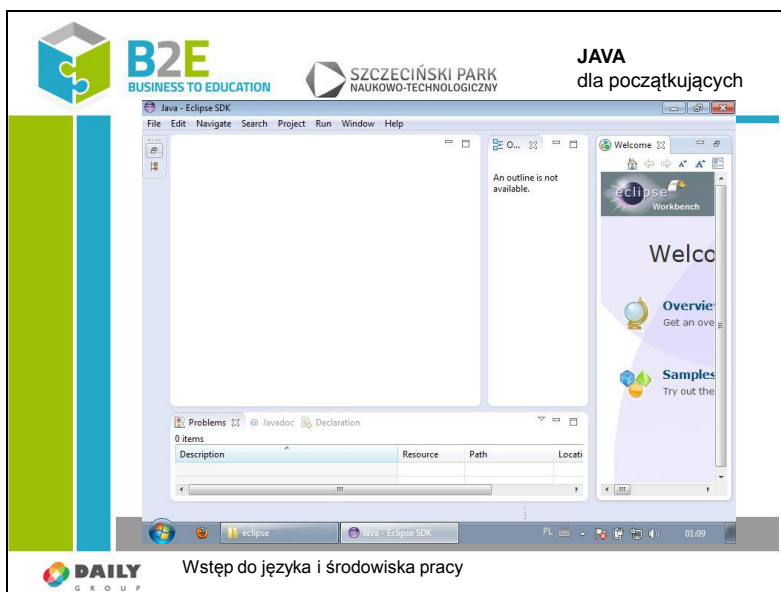
B2E BUSINESS TO EDUCATION **SZCZECIŃSKI PARK** NAUKOWO-TECHNOLOGICZNY **JAVA** dla początkujących

DAILY GROUP Wstęp do języka i środowiska pracy

Klikamy „Finish”.



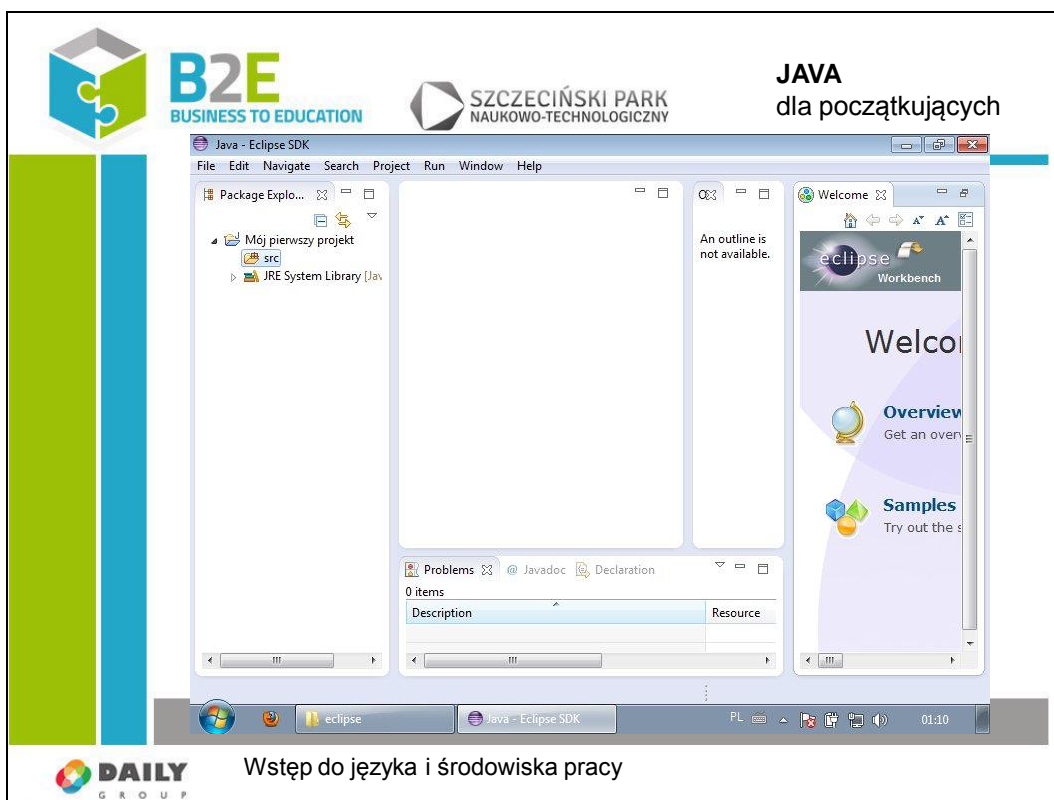
Slajd 36



Aby zobaczyć zawartość projektu klikamy na ikonę znajdującą się w lewym, górnym rogu (poniżej pozycji w menu o nazwie „File”).

Powinien wyświetlić się nam „Package Explorer”

Slajd 37

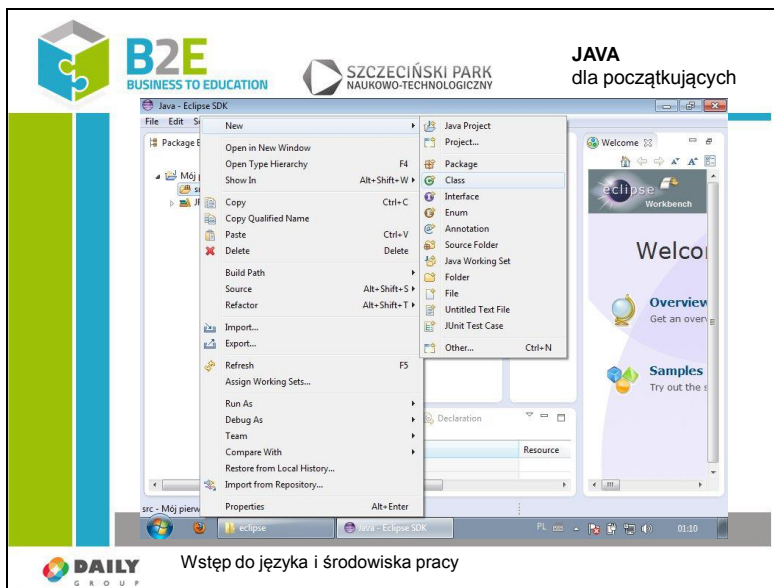


Pliki kodu źródłowego przechowywane są w folderze „src”.

Nowy projekt nie posiada żadnych plików. W celu stworzenia nowego należy kliknąć na folderze „scr” prawym przyciskiem myszy.

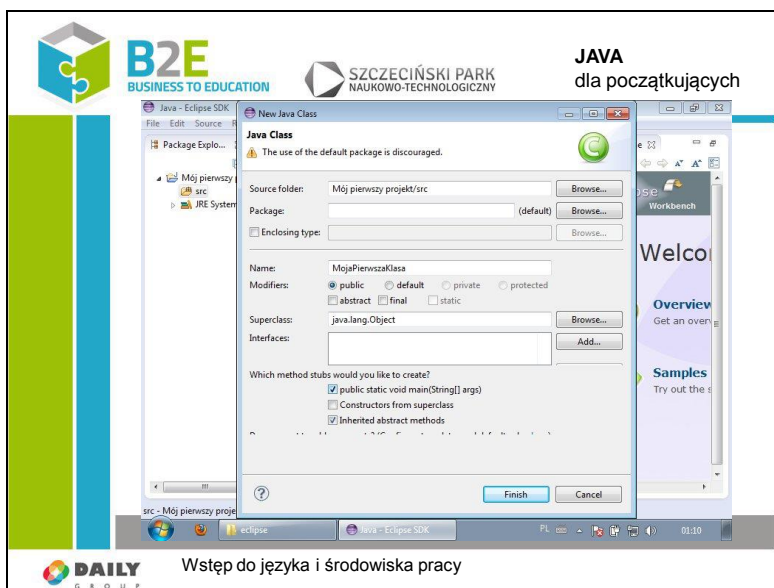


Slajd 38



Następnie wybieramy: New/Class.

Slajd 39



Podajmy nazwę klasy. Nie używamy spacji i zaczynamy dużą literą.

Zaznaczamy również opcję: „public static void main(String [] args)”.

Całość zatwierdzamy przyciskiem „Finish”.



Slajd 40

B2E BUSINESS TO EDUCATION **SZCZECIŃSKI PARK NAUKOWO-TECHNOLOGICZNY** **JAVA dla początkujących**

Wstęp do języka i środowiska pracy

Na środku ekranu wyświetliła się zawartość pliku. Znajduje się on w folderze „src” w (default package)”.

Slajd 41

B2E BUSINESS TO EDUCATION **SZCZECIŃSKI PARK NAUKOWO-TECHNOLOGICZNY** **JAVA dla początkujących**

Wstęp do języka i środowiska pracy

Jeżeli chcemy wyświetlić plik na całej szerokości ekranu, klikamy podwójnie na zakładce z zawartością pliku.



Slajd 42

B2E BUSINESS TO EDUCATION **SZCZECIŃSKI PARK NAUKOWO-TECHNOLOGICZNY** **JAVA dla początkujących**

```

public class MojaPierwszaKlasa {
    /**
     * @param args
     */
    public static void main(String[] args) {
        System.out.println("Hello world!");
    }
}
    
```

Wstęp do języka i środowiska pracy

Wers zaczynający się na „//” zastępujemy wyrażeniem:
System.out.println("Hello world!");
Skrótem klawiszowym „Ctrl” + „S” zapisujemy zmiany.

Slajd 43

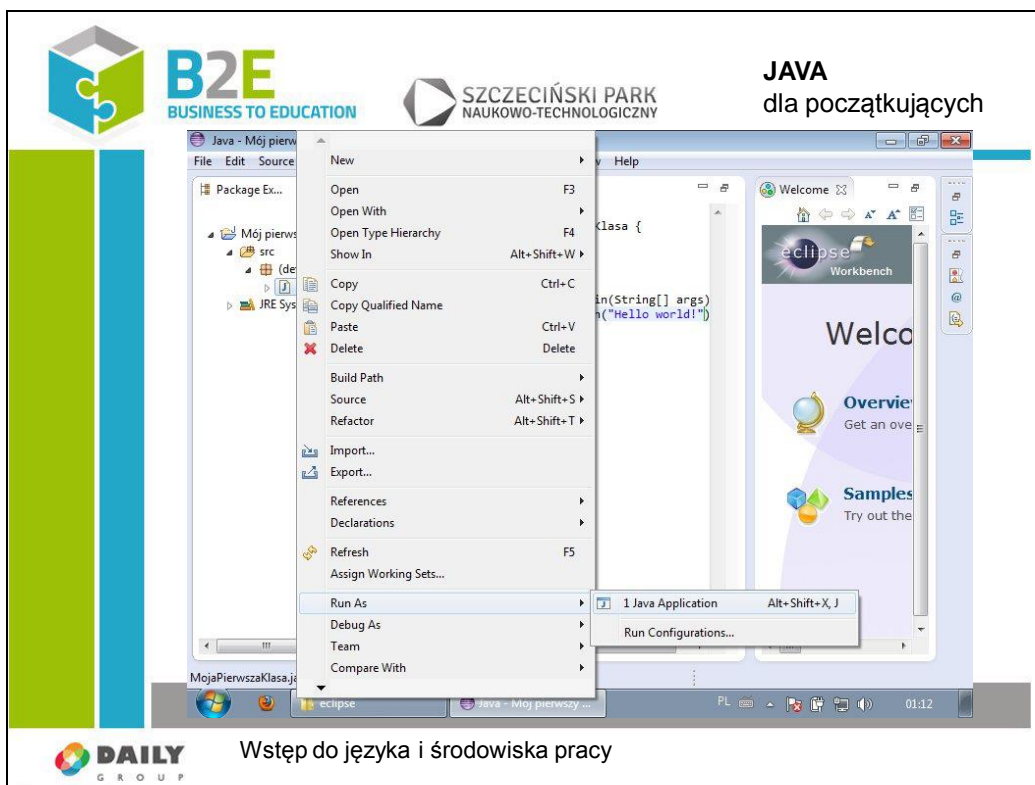
B2E BUSINESS TO EDUCATION **SZCZECIŃSKI PARK NAUKOWO-TECHNOLOGICZNY** **JAVA dla początkujących**

Wstęp do języka i środowiska pracy

Ponownie klikamy podwójnie na nazwie karty w celu jej zmniejszenia.
Dzięki temu ponownie widzimy strukturę plików w projekcie.
W celu uruchomienia pliku klikamy w obszarze „Package Explorer” na nim prawym klawiszem myszy.



Slajd 44



Wybieramy opcję:
„Run As/1 Java Application”



Slajd 45



Jako rezultat operacji powinniśmy zobaczyć konsolę z widocznym napisem „Hello world!”.



Slajd 46

JAVA
dla początkujących

Zadanie:


Karty w Eclipse można zamykać i przemieszczać w zależności od indywidualnych preferencji programisty.

Dostosuj swoje środowisko tak aby widzieć strukturę projektu, plik tekstowy i konsolę jednocześnie, bez potrzeby przełączania kart.

Jeżeli chcesz ponownie wyświetlić zamkniętą kartę, wejdź:



Window/Show View/Other...

I wpisz nazwę karty, którą potrzebujesz.



Wstęp do języka i środowiska pracy

Slajd 47


JAVA
dla początkujących

```

public class MojaPierwszaKlasa {

    public static void main(String[] args) {
        System.out.println("Hello world!");
    }

}
```



Wstęp do języka i środowiska pracy

Przeanalizujemy kod pierwszego programu.

Wszystko w Javie jest obiektem. Obiekt jest instancją danej klasy, czyli pewnym tworem przechowywanym w pamięci, który przechowuje dane i nie tylko. Pojęcie obiektu zostanie dokładnie wytłumaczone na lekcji 3.

W Javie w każdym pliku znajduje się jedna klasa. Później przedstawię jak definiować klasy wewnątrz innych klas. Jednak po procesie kompilacji zawsze klasa jest osobnym plikiem. Bardzo ważne jest aby klasa miała taką samą nazwę jak plik, w którym się znajduje. Pominięcie tej zasady traktowane jest jako błąd!

Wewnątrz klasy (pomiędzy nawiasami „{” i „}”) znajdują się pola i metody. W naszym




przykładzie nie ma pól, jest tylko jedna metoda o nazwie „main”.


W późniejszych lekcjach zostaną również wytłumaczone znaczenia słów „public”, „static” i „void”. Na razie wszystkie polecenia będziemy pisać we wnętrzu metody „main”. Jest to szczególna metoda, ponieważ jest zawsze wywoływana przy próbie uruchomienia obiektu.

Obiekt przeznaczony do, bezpośredniego uruchomienia musi posiadać metodę: „public static void main(String [] args)”.

Slajd 48



B2E
BUSINESS TO EDUCATION



SZCZECIŃSKI PARK
NAUKOWO-TECHNOLOGICZNY


JAVA
dla początkujących

```
public class MojaPierwszaKlasa {

    /**
     * Pierwszy komentarz
     */
    public static void main(String[] args) {
        // Drugi komentarz
        System.out.println("Hello world!");
    } // koniec metody

    /*
     Trzeci komentarz
     */
}

```



DAILY
GROUP

Wstęp do języka i środowiska pracy

W Javie, w prawie dowolnym miejscu, możemy pisać komentarze. Są one przeznaczone wyłącznie dla programistów.

Najczęściej spotykanymi komentarzami są te zaczynające się od symboli „//”. Możemy stosować je w tych samych liniach, w których są elementy kodu. Po wstawieniu symboli „//” kompilator pominie cały tekst, aż do końca linii.

Innym typem komentarzy są komentarze blokowe. Otwieramy je symbolem „/*” i zamykamy „*/”. Pomiędzy tymi znacznikami możemy wstawić dowolną liczbę wersów.

Dość szczególnym typem komentarzy blokowych są te zaczynające się symbolami „/**” i zamykające „*/”. Zostawia się je przed definicją klasy i jej metodami. Nazywane są „Javadoc”. Pełnią ważną funkcję w tworzeniu dokumentacji. W późniejszych lekcjach zostanie pokazane jak je tworzyć, oraz jak korzystać z tej dokumentacji.

Pisanie komentarzy jest bardzo ważne. W dużych projektach, nad którymi pracuje wiele osób, są one praktyczną formą zostawiania wiadomości. Jeżeli napiszemy fragment kodu, o bardzo skomplikowanej logice, warto go skomentować. W takim przypadku bardzo przydatne mogą okazać się nawet lakoniczne informacje o podstawowych zasadach działania kodu. Bardzo ułatwią one rozwijanie aplikacji osobom, które będą pracowały po nas.

Praca zespołowa nie jest jedynym powodem, dla którego pisze się komentarze. Jeżeli zechcemy po pewnym czasie powrócić do rozwijania swojej aplikacji, możemy nie pamiętać szczegółów dotyczących kodu. Oszczędzimy wiele czasu czytając komentarze,



zamiast szczegółowo analizować kod.

Dla kompilatora bieżący kod jest identyczny jak ten na poprzednim slajdzie.

Slajd 49

B2E
BUSINESS TO EDUCATION

SZCZECIŃSKI PARK
NAUKOWO-TECHNOLOGICZNY

JAVA
dla początkujących

```
System.out.println("Hello  
world!");  
  
System.out.print("Hello");  
System.out.println(" world!");  
  
System.out.println(2+2);
```

Wynik:
Hello world!
Hello world!
4

DAILY
GROUP

Wstęp do języka i środowiska pracy

Metoda:

„System.out.println”

Wyświetla tekst podany w nawiasach (i w znakach „”) w konsoli i wstawia znak nowej linii.

W instancji klasy PrintStream „System.out” są również inne metody, np. „print” wstawia tekst bez znaku nowej linii.



Metoda „println” wyświetla nie tylko tekst. W Javie wszystkie obiekty mogą być wyświetlone w postaci tekstu. Będziemy wykorzystywać ją na najbliższych lekcjach do sprawdzania wyników przebiegu programów.

Jest to jedna z najpowszechniej używanych metod. W środowisku Eclipse możemy wywołać ją automatycznie wpisując „sysou” i następnie wciskając „Ctrl” + „spacja”.

Skrót klawiszowy „Ctrl” + „spacja” jest powszechnie wykorzystywany (nie tylko w Eclipse!) do podpowiadania elementów składni. Będziemy często go wykorzystywać.



Slajd 50





JAVA
dla początkujących

```

public class MojaPierwszaKlasa {
    public static void main(String[] args) {
        System.out.println("Hello world!");
        ;
        System.out.
        print("Hello");
        System.
        out
        .println (" world!");

        System.out.println
        (2+2);
    }
}
    
```



Wstęp do języka i środowiska pracy

W Javie każde polecenie musi kończyć się średnikiem. Nie oznacza to jednak, że średnik musi znajdować się na końcu każdej linii. Jeżeli nasze polecenia są zbyt długie (nie zaleca się stosować wersów o długości 200 znaków, są nieczytelne), możemy umieścić je w kilku wersach, pamiętając o zakończeniu całości średnikiem

Na slajdzie znajduje się poprawny kod programu, wykonujący to co zostało przedstawione na slajdzie poprzednim. Należy zauważyć jednak, że polecenia zapisane w taki sposób nie są czytelne dla ludzi.

Średnika nie wstawiamy, np. po nawiasach klamrowych.

Slajd 51




JAVA
dla początkujących

Zadanie:

Napisz krótki program wypisujący na konsolę słowa:
„jeden dwa i trzy i cztery”;



Człowiek - najlepsza inwestycja



Projekt współfinansowany przez Unię Europejską w ramach Europejskiego Funduszu Społecznego



Wstęp do języka i środowiska pracy

public class Zolnierz {


```
public static void main(String[] args) {  
    System.out.println("jeden dwa i trzy i cztery");  
}  
  
}
```

Opis założonych osiągnięć ucznia

Po tej lekcji uczniowie poznają teoretyczną wiedzę na temat tego, czym jest Java. Będą umieli rozpocząć pracę w Javie w odpowiednim środowisku.

Lekcja 2 Typy podstawowe i instrukcje sterujące

Cel lekcji

Celem lekcji jest poznanie podstawowych typów danych, wytłumaczenie jak bezpiecznie przechowywać dane w zmiennych. Poznanie pętli sterujących i zarządzanie przebiegiem wykonania programu za ich pomocą.

Treść – slajdy z opisem

Slajd 1



Slajd 1 zawiera tytułowy slajd lekcji. W górnej części znajdują się loga: B2E BUSINESS TO EDUCATION, SZCZECIŃSKI PARK NAUKOWO-TECHNOLOGICZNY, KAPITAŁ LUDZKI NARODOWA STRATEGIA SPÓJNOŚCI, UNIA EUROPEJSKA EUROPEJSKI FUNDUSZ SPOŁECZNY oraz DAILY GROUP. Tytuł slajdu to "Java - lekcja 2" z podtytułem "Typy podstawowe i instrukcje sterujące". W prawym górnym rogu napisano "JAVA dla początkujących". W dolnej części slajdu znajduje się informacja o finansowaniu: "Człowiek - najlepsza inwestycja" i "Projekt współfinansowany przez Unię Europejską w ramach Europejskiego Funduszu Społecznego".

Slajd 2 Typy wbudowane są szczególnym typem danych. Występują ponieważ traktowanie prostych danych jako obiekty nie jest szczególnie wydajne. Zmienne te przechowują proste wartości

Typ „boolean” służy do przechowywania wartości logicznej: prawda lub fałsz.



Typ „char” służy do przechowywania znaku (liter). Posiada 16 bitów, więc możemy swobodnie używać polskie znaki.

Typ „byte” służy do przechowywania bajtów danych.

Typy „short”, „int” i „long” służą do przechowywania liczb całkowitych. W zależności od zakresu na jakim chcemy operować wybieramy odpowiedni typ.

Typy „float” i „double” przechowują liczby zmiennoprzecinkowe zgodnie ze standardem IEEE754.



JAVA

dla początkujących



Lista typów wbudowanych:

Nazwa typu	Rozmiar [bity]	Wartość minimalna	Wartość maksymalna	Odpowiednik obiektowy
boolean	-	-	-	Boolean
char	16	Unicode 0	Unicode 2 ¹⁶ -1	Character
byte	8	-128	+127	Byte
short	16	-2 ¹⁵	+2 ¹⁵ -1	Short
int	32	-2 ³¹	+2 ³¹ -1	Integer
long	64	-2 ⁶³	+2 ⁶³ -1	Long
float	32	IEEE754	IEEE754	Float
double	64	IEEE754	IEEE754	Double
void	-	-	-	Void

DAILY GROUP Typy podstawowe i instrukcje sterujące

Typ „void” oznacza zerową informację. Nie można stworzyć zmiennej typu „void”, ale może on być zwracany przez metody.

Slajd 3

JAVA

dla początkujących

```

boolean x1 = true;
char x2 = 'Z';
byte x3 = 102;
short x4 = 30000;
int x5 = 200000000;
long x6 = 100000000000000000L;
float x7 = 2.12f;
double x8 = 2.35;

System.out.println(x1);
System.out.println(x2);
System.out.println(x3);
System.out.println(x4);
System.out.println(x5);
System.out.println(x6);
System.out.println(x7);
System.out.println(x8);
    
```

Wynik:

```

true
Z
102
30000
200000000
100000000000000000
2.12
2.35
    
```

DAILY GROUP Typy podstawowe i instrukcje sterujące



Na slajdzie pokazane jest jak powinno się inicjalizować zmienne (nadawać wartości):

- boolean – używamy słów „true” lub „false”,
- char – podajemy pojedynczy znak w apostrofach,
- byte, short, int – liczba całkowita,
- long – liczba całkowita z literą „l” lub „L” na końcu,
- float – liczba ułamkowa z literą „f” lub „F” na końcu,
- double – liczba ułamkowa.

W celu wyświetlenia zainicjowanych wartości używamy wcześniej przedstawionej metody „println”. Jak widać w wyniku nasze zmienne przechowują dokładnie takie wartości jak podaliśmy.



Slajd 4

JAVA
dla początkujących


```

float f = 3.1415926535897932384626433f;
double d = 3.1415926535897932384626433f;
System.out.println("3.1415926535897932384626433");
System.out.println(f);
System.out.println(d);
    
```

Wynik:

```

3.1415926535897932384626433
3.1415927
3.1415927410125732
    
```





Typy podstawowe i instrukcje sterujące

Dokonując operacji na typach zmiennoprzecinkowych należy szczególnie pamiętać o dokładności wykonywanych obliczeń.

Przykład pokazuje dokładność z jaką typy „float” i „double” przechowują liczbę Pi.

Nie oznacza to jednak, że w Javie nie można przeprowadzać obliczeń o wyższej precyzji. W standardowych bibliotekach Javy znajdują się klasy takie jak „BigInteger” i „BigDecimal”, które mogą wykorzystać do zapamiętania liczby całą dostępną pamięć RAM komputera. Są to jednak obiekty. Jak korzystać z obiektów zostanie wytłumaczone w następnej lekcji.

Slajd 5

JAVA
dla początkujących

```


int a = 8, b = 2, c;
c = a + b;
System.out.println(c);
System.out.println(a+b);
System.out.println(a-b);
System.out.println(b-a);
System.out.println(a*b);
System.out.println(a/b);
System.out.println(c/a);
System.out.println(c%a);

a += a;
System.out.println(a);
a -= c;
System.out.println(a);
a *= a;
System.out.println(a);
a /= b;
System.out.println(a);
    
```

Wynik:

```

10
10
6
-6
16
4
1
2
16
6
36
18
    
```



Typy podstawowe i instrukcje sterujące

Na typach wbudowanych możemy wykonywać operacje matematyczne korzystając z dostępnych operatorów.

Operator „%” służy do otrzymywania reszty z dzielenia, pozostałe działają zgodnie z intuicją. Przypisanie sumy liczb a i b do c nastąpiło poprzez użycie operatora podstawiania („=”).



Operatory „+”, „-” i „*” działają zgodnie z intuicją (suma, różnica, mnożenie).

Należy zwrócić szczególną uwagę na operator „/” dzielenia. W przypadku gdy dzielimy a przez b dostajemy poprawny wynik ($8/2=4$). Kiedy jednak dzielimy c przez a ($10/8=1,25$) otrzymaliśmy wynik 1 zamiast 1,25. Dzieje się tak, ponieważ typy te nie przechowują części ułamkowej. Nie następuje również zaokrąglenie wartości. Cyfry znajdujące się po przecinku są „odcinane”, więc zamiast np. 1,8 też dostaniemy 1.

Operatory użyte poniżej biorą argumenty z obu stron, przypisując wartość do zmiennej znajdującej się z lewej strony. Wykonują analogicznie operacje sumy, różnicy, mnożenia i dzielenia.

Slajd 6

B2E BUSINESS TO EDUCATION

SZCZECIŃSKI PARK NAUKOWO-TECHNOLOGICZNY

JAVA
dla początkujących

```
int i = 1;
// preinkrementacja
System.out.println(++i);
// postinkrementacja
System.out.println(i++);
// predekrementacja
System.out.println(--i);
// postdekrementacja
System.out.println(i--);
```

Wynik:

```
2
2
2
2
```

DAILY GROUP Typy podstawowe i instrukcje sterujące

Operator preinkrementacji najpierw zwiększa zmienną o jedną jednostkę, następnie zwraca jej wartość. 1 zostało zwiększone, a następnie wyświetlone. Po operacji $i=2$.


Operacja postinkrementacji najpierw zwraca wartość, a następnie ją zwiększa 2 zostało najpierw wyświetlone, a następnie zwiększone. Po tej operacji $i=3$.

Operator predekrementacji najpierw zmniejsza zmienną o jednostkę, następnie zwraca jej wartość. 3 zostało zmniejszone, a następnie wyświetlone. Po operacji $i=2$.


Operacja postdekrementacji najpierw zwraca wartość, a następnie ją zmniejsza. 2 zostało najpierw wyświetlone, a następnie zmniejszone. Po tej operacji $i=1$.



Slajd 7



B2E
BUSINESS TO EDUCATION



SZCZECIŃSKI PARK
NAUKOWO-TECHNOLOGICZNY

JAVA
dla początkujących

```


int a = 8, b = 3;
System.out.println(a/b);
System.out.println((float)a/b);
System.out.println((double)a/b);

int x = 1500000000, y = 1500000000;
System.out.println(x+y);
    
```

Wynik:

```

2
2.6666667
2.6666666666666665
-1294967296
    
```



Typy podstawowe i instrukcje sterujące

Fakt, że dla typów „int”, „short”, „long” wartości mniejsze od 1, są obcinane, nie oznacza, że nie możemy ich używać jeżeli w przyszłości może zajść potrzeba dokonania dokładniejszych obliczeń. Istnieje możliwość rzutowania jednych typów na drugie.

Na slajdzie mamy trzy możliwe wyniki dzielenia 8 przez 3 w zależności od dokładności wyniku jaki chcemy otrzymać.

Poniżej mamy dość szczególny przypadek. Zastanówmy się, jak to jest możliwe, że dodając dwie liczby dodatnie, otrzymaliśmy ujemny wynik.

Proszę wyobrazić sobie, że typy dla liczb całkowitych mogą się „przekręcać”.

Dobłą analogią do tego zjawiska jest przykład licznika samochodowego. Jeżeli wartość na nim pokazywana będzie cały czas rosła, to przekroczy wartość maksymalną i ponownie przekroczy zero. Z omawianymi typami dzieje się dokładnie tak samo.

Wartość maksymalna dla „int” to $2^{31}-1$ (około 2 mld). Jeżeli zechcemy zapisać 3 mld, to „licznik się przekręci” i do minimalnej wartości -2^{31} (około -2 mld) dodana zostanie wartość będąca różnicą $3 \text{ mld} - 2 \text{ mld} = 1 \text{ mld}$. Otrzymaliśmy w ten sposób wynik około -1 mld.



Slajd 8

B2E
BUSINESS TO EDUCATION

SZCZECIŃSKI PARK
NAUKOWO-TECHNOLOGICZNY

JAVA
dla początkujących

Operatory logiczne (wartości mogą być):

- „==” - równe,
- „!=” - różne,
- „<” - mniejsze,
- „>” - większe,
- „<=” - mniejsze lub równe,
- „>=” - większe lub równe.

```
int a = 3, b = 5;  
System.out.println(a>b ? a : b);
```

Wynik:
5

DAILY
GROUP

Typy podstawowe i instrukcje sterujące


Przedstawione operatory służą do porównywania wartości. Działają zgodnie z tym, czego używaliśmy na lekcjach matematyki.

Poniżej mamy przykład jednego trójargumentowego operatora operatora w Javie. Składa się z dwóch znaków: „?” i „:”. W pierwszej kolejności podajemy wartość logiczną. Wynikiem każdej operacji logicznej jest wartość logiczna. Możemy w tym miejscu równie dobrze użyć zmiennej typu boolean. Następnie, po znaku „?” podajemy polecenie, które ma zostać wykonane w przypadku prawdy. Po znaku „:” podajemy polecenie dla fałszywej wartości logicznej.


Wynikiem użycia operatora „?” w przykładzie pokazanym na slajdzie jest drukowanie w konsoli wartości większej (a lub b).



Slajd 9



B2E
BUSINESS TO EDUCATION



SZCZECIŃSKI PARK
NAUKOWO-TECHNOLOGICZNY

JAVA
dla początkujących

Instrukcja warunkowa if-else:

```
int a = 3, b = 5, c = 5;


if (a < b) {
    System.out.println("Przykład 1, przypadek 1.");
}

if (a > b) {
    System.out.println("Przykład 2, przypadek 1.");
} else {
    System.out.println("Przykład 2, przypadek 2.");
}

if (a > b) {
    System.out.println("Przykład 3, przypadek 1.");
} else if (b > c) {
    System.out.println("Przykład 3, przypadek 2.");
} else {
    System.out.println("Przykład 3, przypadek 3.");
}
```

Wynik:


Przykład 1, przypadek 1.
Przykład 2, przypadek 2.
Przykład 3, przypadek 3.




Typy podstawowe i instrukcje sterujące

Na slajdzie przedstawione zostały różne warianty instrukcji warunkowej „if”. Korzystając z niej decydujemy, które polecenia mają być wykonane w zależności od zadanych kryteriów. Instrukcja warunkowa „if” działa w następujący sposób. Warunek w nawiasie po słowie „if” musi być wyrażeniem logicznym (dawać wartość prawda lub fałsz). W przypadku prawdy wykonają się instrukcje zawarte w nawiasach klamrowych, za wyrażeniem logicznym. Jeżeli później występuje słowo „else”, to w przypadku prawdy wykonają się instrukcje występujące po słowie else.

Slajd 10



B2E
BUSINESS TO EDUCATION



SZCZECIŃSKI PARK
NAUKOWO-TECHNOLOGICZNY

```
if (true && true && false) {
    System.out.println("Prawda!");
}
else {
    System.out.println("Fałsz!");
}


if (true && true) {
    System.out.println("Prawda!");
}

if (false || true || false) {
    System.out.println("Prawda!");
}

if (true && true) {
    System.out.println("Prawda!");
}
```

Wynik:

Fałsz!
Prawda!
Prawda!
Prawda!



Typy podstawowe i instrukcje sterujące

Dostępne mamy również inne operatory logiczne: koniunkcji („&&”) i alternatywy („||”).



Należy pamiętać, że kiedy użyjemy operatora „&&”, to kiedy przynajmniej jedna z wartości jest fałszywa, całe wyrażenie jest fałszywe. Dlatego wyrażenia sprawdzane są od lewej do prawej. Po napotkaniu pierwszej fałszywej wartości reszta nie jest już sprawdzana. Nie zachodzi taka potrzeba.

Slajd 11

The slide features a header with logos for B2E (Business to Education), Szczeciński Park Naukowo-Technologiczny, and the title 'JAVA dla początkujących'. The main content displays a Java code snippet: `int a = 5, b = 7, c = 0, y = 0; while (c < b) { y += a; c++; } System.out.println(y);`. Below the code, the output is shown as 'Wynik: 35'. At the bottom, there is a logo for DAILY GROUP and the text 'Typy podstawowe i instrukcje sterujące'.

Na slajdzie widzimy przykładową pętlę „while”. Wykonuje ona operację mnożenia a i b, zapisując wynik do c.

Po słowie „while” występuje wyrażenie logiczne, tak jak w instrukcji „if”. W przypadku prawdy instrukcje z nawiasów klamrowych zostaną wykonane. Następnie wyrażenie logiczne jest sprawdzane ponownie i w przypadku prawdy następuje kolejny krok pętli. Instrukcja „while” przestaje działać jeżeli warunek (wyrażenie logiczne w nawiasach) jest fałszywy. Jeżeli źle skonstruujemy warunek logiczny, pętla może się nie zatrzymać! W takim przypadku proces w systemie operacyjnym, który odpowiada naszemu programowi, powinien być „zabity”.



Slajd 12

Ćwiczenie:

Wykorzystując pętlę „while” napisz program obliczający 10 liczbę ciągu Fibonacciego.
Każda liczba ciągu powstaje poprzez dodanie dwóch poprzednich liczb, a dwie pierwsze liczby mają wartość 1.

Przykład:



Liczba 1 = 1
Liczba 2 = 1
Liczba 3 = 1 + 1 = 2
Liczba 4 = 1 + 2 = 3

Typy podstawowe i instrukcje sterujące

```
public class MojaPierwszaKlasa {  
    public static void main(String[] args) {  
        int krok = 0;  
        int n1=1, n2, wynik=0;  
        while (krok<10) {  
            n2 = wynik;  
            wynik += n1;  
            n1 = n2;  
            krok++;  
            System.out.println(krok);  
            System.out.println(wynik);  
            System.out.println();  
        }  
    }  
}
```




Slajd 13

JAVA

dla początkujących

```


int a = 1, b = 1;

do {
    a *= b++;
    System.out.println(a);
} while (a<1000);
    
```

Wynik:

```

1
2
6
24
120
720
5040
    
```





Typy podstawowe i instrukcje sterujące

Pętla „do-while” działa tak samo jak pętla „while”, z tą różnicą, że zawsze występuje pierwszy przebieg pętli. Dopiero po nim sprawdzany jest warunek. W przypadku prawdy następuje kolejny przebieg.

Pętla widoczna na slajdzie wypisuje kolejne wartości silni, aż do momentu, kiedy wartość będzie większa niż 1000.

Slajd 14





JAVA

dla początkujących

Ćwiczenie:

Wykorzystując pętlę „do-while” napisz program znajdujący najmniejszy dzielnik liczby 2629 (dzielnik musi być większy od 1).



Typy podstawowe i instrukcje sterujące

```

public class MojaPierwszaKlasa {
public static void main(String[] args) {
    
```



```
int x = 2639, i = 1;
do {
i++;
} while (x % i != 0);
System.out.println(i);
}
}
```

Slajd 15

B2E BUSINESS TO EDUCATION

SZCZECIŃSKI PARK NAUKOWO-TECHNOLOGICZNY

JAVA
dla początkujących

```
for (instrukcja początkowa; warunek; krok) {}

int a = 2, b = 9, c = 1;

for (int i=0; i<b; i++) {
    c *= a;
}
System.out.println(c);

Wynik:
512

// to działa tak samo
int i=0;
for (; i<b; ) {
    i++;
    c *= a;
}

// to jest poprawne
for (;;) {}
```

DAILY GROUP Typy podstawowe i instrukcje sterujące

Pętla „for” jest bardziej złożona od poprzednich przykładów. W nawiasie po słowie „for” nie występuje tylko warunek. Wyrażenie ma 3 części:

Pierwsza część, instrukcja początkowa, wykonywana tylko raz na samym początku. W naszym przypadku zadeklarowaliśmy i zainicjowaliśmy zmienną i.

Druga część, warunek, jest sprawdzana w każdym kroku. Musi to być wyrażenie logiczne. Dopóki wartość jest prawdziwa, dopóty wykonuje się pętla.

Trzecia część, to instrukcja, która wykonuje się po zakończeniu każdego kroku.

Przedstawiony przykład oblicza wartość liczby a podniesioną do potęgi b.

Każda z trzech instrukcji pętli „for” może być pusta. Z prawej strony slajdu pokazany jest przykład, który działa identycznie jak omawiana pętla. Jednak takie zapisywanie pętli jest mało czytelne dla ludzi.

Pętla „for”, przedstawiona w prawym dolnym rogu, również jest poprawna. Powoduje ona zawieszenie się wykonywanego programu, z powodu braku instrukcji warunkowej. Jest to przykład pętli nieskończonej.


Slajd 16 Przy operowaniu na pętlach możemy skorzystać z dwóch instrukcji: „break” i „continue”.

Przy napotkaniu słowa „continue”, wykonanie bieżącego kroku pętli zostaje zatrzymane i zaczyna się wykonywanie kroku następnego.


Po wystąpieniu słowa „break” następuje całkowite wyjście z pętli „for”.

Powyższy przykład sumuje wszystkie liczby nieparzyste od 1 do 10. Instrukcja warunkowa w pętli „for” w naszym przypadku nie ma żadnego znaczenia. Polecenie „break” przerywa działanie pętli kiedy tylko i będzie większe od 10.





B2E
BUSINESS TO EDUCATION




SZCZECIŃSKI PARK
NAUKOWO-TECHNOLOGICZNY

JAVA dla początkujących

```
for (int i=0, a=0 ; i<100 ; i++) {  
    if (i>10) {  
        break;  
    }  
    if (i%2 == 0) {  
        continue;  
    }  
    a += i;  
    System.out.println(a);  
}
```

Wynik:


```
1  
4  
9  
16  
25
```




DAILY GROUP Typy podstawowe i instrukcje sterujące

Kiedy reszta z dzielenia i przez 2 będzie równa 0 (liczba parzysta), nie nastąpi operacja sumy i wypisania na ekran.

Slajd 17



B2E
BUSINESS TO EDUCATION




SZCZECIŃSKI PARK
NAUKOWO-TECHNOLOGICZNY

JAVA dla początkujących

```
int a=2;  
switch (a) {  
case 1:  
    System.out.println("Wartość 1.");  
    break;  
case 2:  
    System.out.println("Wartość 2.");  
default:  
    System.out.println("Inna wartość.");  
}
```

Wynik:

```
Wartość 2.  
Inna wartość.
```



DAILY GROUP Typy podstawowe i instrukcje sterujące

Ostatnią instrukcją jest „switch”. Dzięki niej możemy decydować, które instrukcje wykonają się w danym przypadku.

Po słowie „switch” podajmy w nawiasie wartość typu „int”, „String” lub „Enum” (nie może to być wartość logiczna). Pojęcia „String” i „Enum” zostaną wytłumaczone w następnych lekcjach.



Następnie możemy zdefiniować dowolną ilość przypadków. Po słowie „case” podajmy wartość dla której dane instrukcje mają się wykonać. Nie otwieramy nawiasu klamrowego! Pamiętajmy, aby wstawić „break”, jeżeli nie chcemy, aby wykonały się instrukcje dla następnego przypadku. Po drugim „case” nie ma tego słowa, więc wykonały się również instrukcje dla przypadku „default”.

„default” jest przypadkiem szczególnym. Jeżeli wartość a nie występuje w żadnym z „case”, wówczas wykonają się instrukcje z tej części.

Slajd 18

Ćwiczenie:

Korzystając z dwóch pętli „for” (jedna wewnątrz drugiej), wypisz w konsoli całą tabliczkę mnożenia dla liczb od 1 do 10.
Do wyrównania tabeli możesz skorzystać z różnic w wykonaniu poleceń „print” i „println”.
Aby uzyskać łańcuchu znaków efektu tabulacji użyj wyrażenia „\t”.

```
public class MojaPierwszaKlasa {  
  
    public static void main(String[] args) {  
  
        for (int i=1 ; i<=10 ; i++) {  
            for (int j=1 ; j<=10 ; j++) {  
                System.out.print(i*j);  
                System.out.print("\t");  
            }  
            System.out.println();  
        }  
    }  
}
```



Slajd 19

Ćwiczenie:

Korzystając z typu „float” i „double” dodaj 10 razy 0,1 do wartości 0. Sprawdź jaki jest wynik sumowania dla każdego z typów.

Typy podstawowe i instrukcje sterujące

```
public class MojaPierwszaKlasa {  
  
    public static void main(String[] args) {  
  
        float af = 0f, bf = 0.1f;  
        double ad = 0, bd = 0.1;  
  
        for (int i=0 ; i<10 ; i++) {  
            af += bf;  
            ad += bd;  
        }  
  
        System.out.println(af);  
        System.out.println(ad);  
    }  
}
```



Slajd 20

JAVA
dla początkujących

Wnioski:

Typy „float” i „double” zawsze obarczone są błędem obliczeniowym!

Nie stosuj porównań typu:

```
If (liczba == 3.14) {...}
```

Aby robić to właściwie należy najpierw dobrze poznać arytmetykę zmiennopozycyjną.

Człowiek - najlepsza inwestycja

KAPITAŁ LUDZKI
NARODOWA STRATEGIA SPÓJNOŚCI

UNIA EUROPEJSKA
EUROPEJSKI
FUNDUSZ SPOŁECZNY

Projekt współfinansowany przez Unię Europejską w ramach Europejskiego Funduszu Społecznego

DAILY
GROUP

Typy podstawowe i instrukcje sterujące

Opis założonych osiągnięć ucznia

Po tej lekcji uczniowie będą wiedzieli jak poprawnie przechowywać dane w zmiennych, oraz jak programować pętle do wykonywania odpowiednich operacji.

Lekcja 3 Wstęp do programowania obiektowego

Cel lekcji

Celem lekcji jest wytłumaczenie, na czym polega idea programowania obiektowego. Opis budowy klasy w Javie, konstruktorów i metod.



Treść - slajdy z opisem

Slajd 1

Slide 1 is a title slide for a Java lesson. It features a green and blue vertical bar on the left. The top left contains logos for B2E (Business to Education) and Szczeciński Park Naukowo-Technologiczny. The top right says 'JAVA dla początkujących'. The main title is 'Java - lekcja 3' with the subtitle 'Wstęp do programowania obiektowego'. At the bottom, there are logos for 'KAPITAŁ LUDZKI' and 'UNIA EUROPEJSKA' with the text 'Człowiek - najlepsza inwestycja' and 'Projekt współfinansowany przez Unię Europejską w ramach Europejskiego Funduszu Społecznego'. The 'DAILY GROUP' logo is at the bottom left.

Slajd 2

Slide 2 is a content slide. It features the same green and blue vertical bar on the left. The top left contains logos for B2E and Szczeciński Park Naukowo-Technologiczny. The top right says 'JAVA dla początkujących'. The main title is 'Objekt jako element świata rzeczywistego'. Below the title is an image showing a silver pen, a blue pen, and a blue pen with its cap off. To the right of the image is a small diagram of a rectangular object with a pink border. At the bottom, there is the text 'Wstęp do programowania obiektowego' and the 'DAILY GROUP' logo.

Niemal każdy element świata rzeczywistego możemy potraktować jako obiekt, niezależnie od tego, czy jest niepodzielnym elementem (np. kamień), czy składa się z wielu innych części (np. długopis).

Jest to naturalny sposób postrzegania świata.



Slajd 3



O każdy obiekt ma typy właściwości

Informacje:

- kolor długopisu,
- cena,
- prędkość maksymalna samochodu,
- masa;

Operacje:

- pisanie,
- zdolność do poruszania się,
- wydawanie dźwięków.

Zauważmy, że ciężko jest wyobrazić sobie temperaturę jako obiekt. Nie utożsamiamy obiektu z przedmiotem! Obiektem może być np. stan techniczny części samochodowej.

Dobrym przykładem na to jest temperatura:

- ma swoją wartość,
- może maleć i wzrastać,
- nie może być niższa od zera absolutnego.

Jeżeli chcielibyśmy stworzyć model pomieszczenia, w którym się znajdujemy, obiektem powinno być przede wszystkim samo pomieszczenie, przedmioty znajdujące się w środku, jak i temperatura powietrza.

Byłby to model obiektowy. W jasny i czytelny sposób pozwoliłby zachować niezbędne informacje na temat stanu wszystkich przedmiotów, oraz przewidzieć możliwe zdarzenia. Jeżeli jest tablica i kreda, to możliwe, że któraś z osób coś na niej napisze. Jeżeli nie znalazłby się obiekt z operacją „pisz” (kreda lub długopis), to z obiektowego punktu widzenia niemożliwe jest, aby w tym pomieszczeniu zostało coś napisane.



Slajd 4

The slide features a central diagram illustrating generalization and specialization. At the top level is an orange box labeled "Pojazdy". Inside this box is a pink box labeled "Pojazdy spalinowe", which contains images of a blue car and a motorcycle. To the right of the pink box, still within the orange box, is an image of a green bicycle. The slide also includes logos for B2E (Business to Education), Szczeciński Park Naukowo-Technologiczny, and DAILY GROUP. The text "Wstęp do programowania obiektowego" is at the bottom.

Na obiektach możemy dokonywać generalizacji i specjalizacji. Są to kluczowe zjawiska, jeżeli chcemy skorzystać z mechanizmu dziedziczenia. Pojęcie dziedziczenia zostanie wyjaśnione na następnej lekcji.

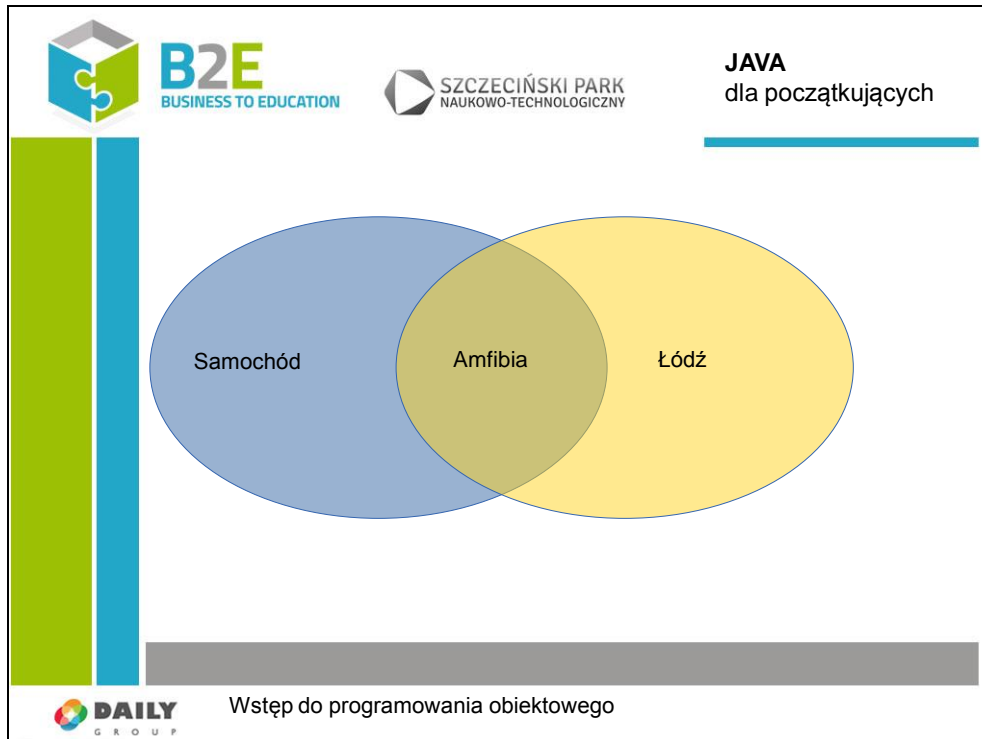
Samochód jest pojazdem spalinowym, jeżeli powiemy, że jest tylko pojazdem, wtedy następuje generalizacja.

Jeżeli powiemy, że samochód jest pojazdem, a ponadto jest pojazdem spalinowym, to następuje specjalizacja. Zjawisko to jest bardzo przydatne. Wyobraźmy sobie, że piszemy grę, w której występuje ruch uliczny. Na skrzyżowaniu, na czerwonym świetle stoi grupa pojazdów (samochody, rowery, motocykle). Kiedy zapali się zielone światło, wywołamy na każdym z pojazdów metodę „jeźdź”. Nie interesuje nas wtedy jakiego typu jest poszczególny pojazd. Wszystkie one potrafią jeździć, zatrzymywać się i skręcać.

W przypadku, kiedy ta sama grupa pojazdów dojedzie do wjazdu na autostradę, musimy zatrzymać wszystkie pojazdy, które nie są spalinowe. Jazda rowerem po autostradzie jest zakazana.



Slajd 5



Pojedynczy obiekt może posiadać cechy, które są zbiorem cech, należących do dwóch różnych pojazdów.

Np. amfibia posiada cechy zarówno łodzi, jak i samochodu.

Slajd 6

B2E BUSINESS TO EDUCATION

SZCZECIŃSKI PARK NAUKOWO-TECHNOLOGICZNY

JAVA dla początkujących

Ćwiczenie:
Mając do dyspozycji terminy: koń, zwierzę, osioł, muł
Rozmieść je na kartce i zakreśl odpowiednie okręgi tak aby uwzględnić relację relacje między nimi

DAILY GROUP Wstęp do programowania obiektowego



Slajd 7

B2E
BUSINESS TO EDUCATION

SZCZECIŃSKI PARK
NAUKOWO-TECHNOLOGICZNY

JAVA
dla początkujących

Typy widoczności:
- public,
- package.

```
public class MojZnak {  
  
}
```

DAILY
GROUP

Wstęp do programowania obiektowego

Klasę w Javie zapisujemy podając typ widoczności, słowo kluczowe „class”, nazwę klasy i nawiasy klamrowe, w których znajduje się ciało klasy.


Omówimy dwa typy widoczności: „public” i domyślny (Oficjalnie Oracle nazywa ten typ widoczności “package-private”, część programistów mówi o nim “package-protected” a jeszcze inni traktują go jako domyślny. W języku polskim mówi się czasem o “dostępie pakietowym”).

„public” oznacza, że nasza klasa będzie widoczna z każdego miejsca w kodzie.


Klasy w naszej aplikacji możemy trzymać w pakietach. Mechanizm działa tak, jak drzewo folderów w systemie operacyjnym. W Javie słowo kluczowe „package” nie opisuje typu widoczności, służy do deklarowania przynależności klasy do danego pakietu. Jeżeli chcemy skorzystać z typu widoczności domyślnej, to nie wpisujemy żadnego typu widoczności w definicji klasy. Wówczas nasza klasa będzie widoczna dla wszystkich klas znajdujących się w danym pakiecie.



Slajd 8



B2E
BUSINESS TO EDUCATION



SZCZECIŃSKI PARK
NAUKOWO-TECHNOLOGICZNY

JAVA
dla początkujących

```

public class MojZnak {

    private char znak;
    public int liczba;

}


public class MojaPierwszaKlasa {

    public static void main(String[] args) {

        MojZnak mojZnak = new MojZnak();
        mojZnak.liczba = 2;

    }

}
                
```



Wstęp do programowania obiektowego

Jeżeli chcemy przechowywać w klasie pewne informacje, tworzymy pola. W Javie nie można przechowywać danych, które nie należą do żadnego obiektu! (wyjątkiem od reguły są pola statyczne, które należą do definicji klasy a nie jej instancji).

Pola zawierają dodatkowo typ widoczności „private”. Oznacza to, że dane zmienna jest widoczna tylko w ciele danej klasy. Pola publiczne są widoczne w każdej klasie w aplikacji.

Na slajdzie w metodzie „main” po raz pierwszy świadomie tworzymy obiekt! Obiekt jest reprezentacją danej klasy (instancją) „powołaną do życia”. Przeanalizujemy jak odbywa się ten proces.

W pierwszej kolejności musimy stworzyć referencję. Referencja jest adresem na komórkę w pamięci, gdzie znajduje się dana instancja klasy. Referencja posiada również typ obiektu na który wskazuje.

Piszemy:

TypReferencji nazwaReferencji;

W tym przypadku jest to referencja o nazwie „mojZnak”, typu „MojZnak”.

Korzystając ze słowa kluczowego „new” alokujemy obszar pamięci operacyjnej komputera na przechowanie obiektu danego typu. Następnie znajduje się konstruktor danej klasy.


Należy pamiętać, że nazwy klas piszemy dużą literą, a nazwy pól z małej.0

Do pól danej klasy odwołujemy się pisząc referencję, wstawiając symbol kropki, następnie wstawiając nazwę pola (tak jak w przykładzie).


Zauważmy, że do pola „znak” nie możemy się odwołać (jest prywatne).



Slajd 9



B2E
BUSINESS TO EDUCATION




SZCZECIŃSKI PARK
NAUKOWO-TECHNOLOGICZNY

JAVA
dla początkujących

```

public class MojZnak {
    private char znak;
    public int liczba;
}

public class MojaPierwszaKlasa {
    public static void main(String[] args) {
        MojZnak mojZnak = new MojZnak();
        System.out.println(mojZnak.liczba);
    }
}
    
```



Wstęp do programowania obiektowego

W tym przykładzie chcemy wyświetlić pole „liczba” nie inicjalizując go. W tym przypadku otrzymamy 0, ale na ogół takie przypadki mogą być bardzo niebezpieczne i zagrażać stabilności naszego programu. Dlatego dobrym zwyczajem jest inicjalizacja zmiennych, mimo iż w Javie są dokładnie sprecyzowane reguły wartości domyślnych.

Data Type	Default Value (for fields)
-----------	----------------------------

byte	0
short	0
int	0
long	0L
float	0.0f
double	0.0d
char	'\u0000'


String (or any object)	null
------------------------	------

boolean	false
---------	-------


Jeżeli tworzymy zaawansowane obiekty, musimy mieć kontrolę nad wartościami domyślnymi ich pól.



Slajd 10



B2E
BUSINESS TO EDUCATION



SZCZECIŃSKI PARK
NAUKOWO-TECHNOLOGICZNY

JAVA
dla początkujących

```

public class MojZnak {

    private char znak;
    public int liczba;


    public MojZnak() {
        liczba = 7;
    }

}

public class MojaPierwszaKlasa {

    public static void main(String[] args) {
        MojZnak mojZnak = new MojZnak();
        System.out.println(mojZnak.liczba);
    }

}
    
```




Wstęp do programowania obiektowego


Konstruktor jest metodą specjalną. Jeżeli nie zdefiniujemy własnego, to kompilator dołącza domyślny konstruktor bezparametrowy. Dlatego w poprzednich przykładach mogliśmy pisać „new MojZnak()”.

Tworzenie własnych konstruktorów rozwiązuje problem inicjalizowania pól. W tym przypadku, po stworzeniu klasy pole „liczba” ma wartość 7.

Slajd 11



B2E
BUSINESS TO EDUCATION



SZCZECIŃSKI PARK
NAUKOWO-TECHNOLOGICZNY

JAVA
dla początkujących

```

public class MojZnak {

    private char znak;
    public int liczba;


    public MojZnak(int liczba) {
        this.liczba = liczba;
    }

}

public class MojaPierwszaKlasa {

    public static void main(String[] args) {
        // to już nie jest poprawne
        // MojZnak mojZnak = new MojZnak()
        MojZnak mojZnak = new MojZnak(5)
        System.out.println(mojZnak.liczba)
    }

}
    
```



Wstęp do programowania obiektowego

Na slajdzie widoczny jest przykład konstruktora z jednym parametrem. Tworząc obiekt, użytkownik musi podać wartość początkową dla pola „liczba”. Kompilator nie dołączył już



konstruktora domyślnego. Jest to jedyny sposób w jaki możemy stworzyć obiekt. Słowo kluczowe „this” jest referencją na obiekt, w którym aktualnie się znajdujemy. Polecenie „this.liczba = liczba” może wyglądać dziwnie, ale jest całkowicie bezpieczne. Ponieważ argument ma taką samą nazwę jak pole przysłania „ukrywa” nam właśnie to pole. Dlatego sam zapis „liczba = liczba”, będąc poprawnym syntaktycznie byłby bezsensownym przypisaniem wartości argumentu do tego samego argumentu. Używając słowa kluczowego „this” możemy dostać się do przysłoniętego pola. Powoduje przypisanie wartości argumentu „liczba” do pola obiektu.

W środowisku Eclipse konstruktory możemy zbudować automatycznie korzystając z kreatora dostępnego w:

„Source/Generate Konstruktor using Fields...”

Slajd 12

Ćwiczenie:

Stwórz obiekt zawierający dwa pola liczbowe: a i b (dowolnego typu).
Stwórz dwa konstruktory. Jeden bez parametrów, wtedy zainicjalizuj liczby wartością 0. Drugi z dwoma parametrami i przypisz ich wartości polom.
Klasę nazwij „DwieLiczby”.

```
public class DwieLiczby {
```

```
    private int a;
```

```
    private int b;
```

```
    public DwieLiczby(int a, int b) {
```

```
        this.a = a;
```

```
        this.b = b;
```

```
    }
```

```
    public DwieLiczby() {
```

```
        this.a = 0;
```


```
        this.b = 0;
```

```
    }
```


```
}
```



Slajd 13



B2E
BUSINESS TO EDUCATION



SZCZECIŃSKI PARK
NAUKOWO-TECHNOLOGICZNY

JAVA
dla początkujących

```

public class MojZnak {


    private char znak;
    public int liczba;

    public MojZnak(int liczba) {
        this.liczba = liczba;
    }

    public MojZnak() {

        public class MojaPierwszaKlasa {


            public static void main(String[] args) {
                // teraz już jest poprawne
                MojZnak mojInnyZnak = new MojZnak();
                MojZnak mojZnak = new MojZnak(5);
                System.out.println(mojZnak.liczba);
            }
        }
    }
}
    
```




Wstęp do programowania obiektowego

Możemy stworzyć dowolną liczbę konstruktorów. Muszą jednak różnić się listą argumentów, które przyjmują.

Slajd 14



B2E
BUSINESS TO EDUCATION



SZCZECIŃSKI PARK
NAUKOWO-TECHNOLOGICZNY

JAVA
dla początkujących

```

public class MojZnak {


    private char znak;
    private int liczba;

    public int getLiczba() {
        return liczba;
    }

    public void setLiczba(int liczba) {
        this.liczba = liczba;
    }

}

MojZnak mojZnak = new MojZnak();
mojZnak.setLiczba(13);
System.out.println(mojZnak.getLiczba());
    
```



Wstęp do programowania obiektowego

Pola nie powinny mieć widoczności publicznej. Dobrą praktyką jest enkapsulacja metod i pól, czyli ukrywania składowych obiektu przed światem zewnętrznym, dostępem z zewnątrz. Jest to dobra praktyka, dzięki której możemy zapewnić konsystentny stan naszego obiektu. Przykładowo mając definicję klasy pojazd i mając publiczne pole „przebieg”, narażamy się na to iż inny fragment kodu wprowadzi ujemną wartość. Jak więc na nich operować z poziomu innych klas?

Do tego używa się metod. Metoda posiada typ widoczności, typ który zwraca, nazwę oraz listę argumentów. Ogólnie przyjęta konwencja mówi że nazwa metody powinna być pisana małą literą.

Szczególnym typem zwracanym przez metodę jest „void”. Oznacza to, że metoda wykonuje jedynie pewne operacje, nie zwracając rezultatu. Tak działa metoda „setLiczba”.

W innych przypadkach, jeżeli zadeklarujemy zwracanie typu „int”, musimy w metodzie użyć słowa „return” wpisując za nim tę wartość. Tak działa metoda „getLiczba”.

Uwaga, słowo „return” kończy wykonywanie się metody. Instrukcje, które podamy po tym słowie, nie wykonają się.

Metody zaczynające się od fraz „set”, „get” oraz „is” są charakterystyczne, służą właśnie do obsługi pól. Przedrostki te zdefiniowane są w konwencji JavaBeans.

Nie ma potrzeby ich ręcznego pisania. Można skorzystać z kreatora w Eclipse:

„Source/Generate Getters and Setters...”

Zwłaszcza metoda z przedrostkiem „set” jest bardzo istotna. Daje nam kontrolę nad wartością, którą użytkownik będzie chciał przypisać do pola. Np. jeżeli dane pole będzie przechowywało wartość temperatury, nie może mieć wartości mniejszej niż -273. Przy próbie zapisania wartości -400 metoda „set” powinna wpisać -273.

Slajd 15



B2E
BUSINESS TO EDUCATION

SZCZECIŃSKI PARK
NAUKOWO-TECHNOLOGICZNY

JAVA
dla początkujących

```
public class MojZnak {

    public static String getNapis() {
        return "Napis";
    }

    private static String getInnyNapis() {
        return "InnyNapis";
    }

}

System.out.println(MojZnak.getNapis());
// to jest nieprawidłowe
// System.out.println(MojZnak.getInnyNapis());
```

DAILY
GROUP

Wstęp do programowania obiektowego


Metody, tak samo jak pola, mogą być statyczne. Oznacza to, że są wspólne dla wszystkich instancji danej klasy. Nie ma też potrzeby tworzenia klasy, jeżeli chcemy skorzystać z jej elementu statycznego.

W przykładzie mamy podane prawidłowe wywołanie metody statycznej. Zauważmy, że nie mamy żadnej referencji. Metody statyczne wywoływane są bezpośrednio dla nazwy klasy. Dlatego właśnie aplikacje w Javie uruchamia się za pomocą metody „main”. Metoda ta jest dostępna zanim zostanie stworzona instancja danego obiektu.


Oczywiście nigdy nie wywołamy metody prywatnej z poziomu innego obiektu, tak jak pokazane to jest w komentowanym kodzie.



Slajd 16



B2E
BUSINESS TO EDUCATION




SZCZECIŃSKI PARK
NAUKOWO-TECHNOLOGICZNY

JAVA
dla początkujących

```
public class MojZnak {  
  
    private static int liczba;  
    private static char znak;  
  
    public static void setValues(int mojaLiczba) {  
        liczba = mojaLiczba;  
    }  
  
    public static void setValues(int mojaLiczba, char mojZnak) {  
        liczba = mojaLiczba;  
        znak = mojZnak;  
    }  
  
}
```

MojZnak.setValues(10);
MojZnak.setValues(13, 'X');



DAILY
GROUP

Wstęp do programowania obiektowego

W tym przypadku nie ma sensu tworzenie instancji danej klasy, wszystko jest statyczne. Nie utworzymy również dwóch różnych obiektów typu „MojZnak”. Możemy zapisać:

```
MojZnak z1 = new MojZnak();
```


```
MojZnak z2 = new MojZnak();
```

W praktyce jest to bez sensu. I tak nie przechowamy w obiektach różnych wartości.


Oczywiście tak, jak w przypadku konstruktorów, może być kilka metod o tej samej nazwie, różniących się listą argumentów.



Slajd 17



B2E
BUSINESS TO EDUCATION



SZCZECIŃSKI PARK
NAUKOWO-TECHNOLOGICZNY

JAVA
dla początkujących

```

public class MojZnak {


    private static int liczba = 0;
    private char znak;

    public MojZnak(char znak) {
        this.znak = znak;
        liczba++;
    }

    public static int getLiczba() {
        return liczba;
    }

}

MojZnak z1 = new MojZnak('A');
MojZnak z2 = new MojZnak('B');
System.out.println(MojZnak.getLiczba());
    
```




Wstęp do programowania obiektowego

Jakie jest więc może być praktyczne zastosowanie pól statycznych? Możemy stworzyć klasę „StaleMatematyczne” i stworzyć publiczne, statyczne pole o nazwie pi i wartości „3,14”. We wszystkich klasach programu będziemy mogli korzystać z tej wartości wpisując „StaleMatematyczne.pi”.


Nieco ciekawszy jest przykład podany na slajdzie. Pole „liczba” jest statyczne, co oznacza, że jest wspólne dla wszystkich instancji danej klasy.

W każdym wywołaniu konstruktora następuje inkrementacja jego wartości. W prosty sposób możemy sprawdzić ile obiektów typu „MojZnak” zostało stworzonych.

Slajd 18



B2E
BUSINESS TO EDUCATION



SZCZECIŃSKI PARK
NAUKOWO-TECHNOLOGICZNY

JAVA
dla początkujących

```

public class MojZnak {

    private char znak1;
    private char znak2;


    public void setZnak1(char znak) {
        znak1 = znak;
        checkZnak(znak);
    }

    public void setZnak2(char znak) {
        znak2 = znak;
        checkZnak(znak);
    }

    private void checkZnak(char znak) {
        if (znak == 'X') {
            System.out.println("Wstawiłeś X!");
        }
    }

}

MojZnak mojZnak = new MojZnak();
mojZnak.setZnak1('A');
mojZnak.setZnak2('X');
    
```



Wstęp do programowania obiektowego



Z prywatnych metod korzystamy tylko w obrębie danej klasy. Można stosować je aby nie pisać dwa razy tego samego kodu.

W przykładzie, w każdej metodzie „set” następuje sprawdzenie, czy ustawiana wartość jest równa „X”.

Slajd 19

Ćwiczenie:

Stwórz obiekt zawierający dwie liczby: a i b (dowolnego typu). Nie pozwól aby, któraś z nich była ujemna (jeżeli użytkownik pod ujemną liczbę, zamień ją na 0). Nie pisz dwa razy tego samego kodu.

Udostępnij użytkownikowi dwie metody: suma i iloczyn. Mają zwracać sumę i iloczyn tych liczb.


Klasę nazwij „DwieLiczby”.

```
public class DwieLiczby {
    private int a;
    private int b;
    public DwieLiczby(int a, int b) {
        this.a = validate(a);
        this.b = validate(b);
    }
    public int suma() {
        return a+b;
    }
    public int iloczyn() {
        return a*b;
    }
    private int validate(int x) {
        return x<0 ? 0 : x;
    }
}


public class MojaPierwszaKlasa {
    public static void main(String[] args) {
        DwieLiczby dwieLiczby = new DwieLiczby(10, -1);
        System.out.println(dwieLiczby.suma());
    }
}
```



Slajd 20



B2E
BUSINESS TO EDUCATION



SZCZECIŃSKI PARK
NAUKOWO-TECHNOLOGICZNY


JAVA
dla początkujących

Ćwiczenie:

Stwórz obiekt (o dowolnej nazwie) z dwoma metodami statycznymi: suma i iloczyn.
Niech wynikiem metody suma będzie suma wartości, które zostały zwrócone przez metody suma, dla obiektów DwieLiczby.
Metoda iloczyn ma działać analogicznie.


Pamiętaj, że argumentem metody może być nie tylko typ wbudowany. Poniższy zapis jest poprawny:

```
void metoda(Obiekt a) {...}
```




KAPITAŁ LUDZKI
NARODOWA STRATEGIA SPÓJNOŚCI

Człowiek - najlepsza inwestycja



UNIA EUROPEJSKA
EUROPEJSKI
FUNDUSZ SPOŁECZNY

Projekt współfinansowany przez Unię Europejską w ramach Europejskiego Funduszu Społecznego



Wstęp do programowania obiektowego

```
public class InnaKlasa {
    public static int suma(DwieLiczby x1, DwieLiczby x2) {
        return x1.suma() + x2.suma();
    }
    public static int iloczyn(DwieLiczby x1, DwieLiczby x2) {
        return x1.iloczyn() * x2.iloczyn();
    }
}

public class MojaPierwszaKlasa {
    public static void main(String[] args) {
        DwieLiczby dwieLiczby1 = new DwieLiczby(10, -1);
        DwieLiczby dwieLiczby2 = new DwieLiczby(2, 3);
        System.out.println(InnaKlasa.suma(dwieLiczby1, dwieLiczby2));
        System.out.println(InnaKlasa.iloczyn(dwieLiczby1, dwieLiczby2));
    }
}
```

Opis założonych osiągnięć ucznia

Po tej lekcji uczniowie będą umieli tworzyć proste obiekty w Javie, odpowiednio inicjalizować pola klas w trakcie ich tworzenia.

Lekcja 4 Interfejsy, klasy abstrakcyjne i dziedziczenie

Cel lekcji

Celem lekcji jest wyjaśnienie pojęcia interfejsu i klasy abstrakcyjnej. Wytłumaczone będzie jak zwiększyć efektywność i bezpieczeństwo pracy poprzez stosowanie dziedziczenia.



Treść - slajdy z opisem

Slajd 1

Slide 1 is a title slide for a Java lesson. It features a green and blue vertical bar on the left. The top left contains the B2E logo (Business to Education) and the Szczeciński Park Naukowo-Technologiczny logo. The top right says 'JAVA dla początkujących'. The main title is 'Java - lekcja 4' with the subtitle 'Interfejsy, klasy abstrakcyjne i dziedziczenie'. At the bottom, there are logos for 'KAPITAŁ LUDZKI', 'UNIA EUROPEJSKA', and 'DAILY GROUP'. A small text at the bottom center reads 'Projekt współfinansowany przez Unię Europejską w ramach Europejskiego Funduszu Społecznego'.

Slajd 2

Slide 2 shows a code example for a Java class named 'Pojazd'. The code is as follows:

```
public class Pojazd {  
    private int droga;  
    public Pojazd(int droga) {  
        this.droga = droga;  
    }  
    public void doPrzodu(int odleglosc) {  
        droga += odleglosc;  
    }  
    public void doTylu(int odleglosc) {  
        droga -= odleglosc;  
    }  
}
```

The slide also includes the B2E logo, Szczeciński Park Naukowo-Technologiczny logo, 'JAVA dla początkujących' text, and the 'DAILY GROUP' logo at the bottom. The subtitle 'Interfejsy, klasy abstrakcyjne i dziedziczenie' is repeated at the bottom.

Klasa pokazana na przykładzie implementuj w prosty sposób niektóre właściwości pojazdu. Nowo stworzony pojazd ma licznik odległości ustawiony na zadaną wartość. Możemy poruszać się do przodu i do tyłu. Może to być rower, motocykl, lub samochód. Zastanówmy się co możemy zrobić, jeżeli chcemy stworzyć klasę „PojazdOswietlony”, który posiada te same właściwości co „Pojazd”. Ponadto chcemy, aby „PojazdOswietlony”, miał metody „włączSwiatla” i „wylączSwiatla”.



Pisanie całości nowej klasy, przepisując część funkcjonalności z pojazdu jest niepotrzebne. W Javie (w innych językach również) nie powinno się pisać dwa razy tego fragmentu kodu. Możemy korzystać z mechanizmu dziedziczenia.

Slajd 3

B2E BUSINESS TO EDUCATION

SZCZECIŃSKI PARK NAUKOWO-TECHNOLOGICZNY

JAVA
dla początkujących

```
public class PojazdOswietlony extends Pojazd {
    private boolean swiatla;

    public PojazdOswietlony(int droga, boolean swiatla) {
        super(droga);
        this.swiatla = swiatla;
    }

    public void włączSwiatla() {
        swiatla = true;
    }

    public void wyłączSwiatla() {
        swiatla = false;
    }
}
```

DAILY GROUP Interfejsy, klasy abstrakcyjne i dziedziczenie

Możemy stworzyć nową klasę która posiada wszystkie właściwości klasy „Pojazd”.

Po nazwie klasy, używamy słowa „extends” i podajemy nazwę klasy, po której będziemy dziedziczyć.

Klasa „Pojazd” posiadała tylko jeden konstruktor, z parametrem. Aby stworzyć obiekt „PojazdOswietlony” musimy najpierw wywołać konstruktor z klasy, którą dziedziczymy. Odpowiada za to instrukcja „super”. Jest to w pewnym sensie konstruktor klasy, z której dziedziczymy.

Slajd 4

B2E BUSINESS TO EDUCATION

SZCZECIŃSKI PARK NAUKOWO-TECHNOLOGICZNY

JAVA
dla początkujących



```
class MojaPierwszaKlasa {
    public static void main(String[] args) {
        PojazdOswietlony pojazdOswietlony =
            new PojazdOswietlony(0, false);
        pojazdOswietlony.włączSwiatla();
        pojazdOswietlony.doPrzodu(10);
    }
}
```

DAILY GROUP Interfejsy, klasy abstrakcyjne i dziedziczenie

Jak widać na slajdzie, na instancji obiektu „PojazdOswietlony” możemy wykonywać również metody z klasy „Pojazd”. W ten sposób uniknęliśmy ponownej implementacji niektórych funkcjonalności.



Slajd 5

JAVA
dla początkujących


```

public class Pojazd {
    private int droga;

    public Pojazd(int droga) {
        this.droga = droga;
    }

    public void doPrzodu(int odleglosc) {
        droga += odleglosc;
    }



    protected void zerojLicznik() {
        droga = 0;
    }
}
    
```



Interfejsy, klasy abstrakcyjne i dziedziczenie

Typ widoczności „protected” rozszerza domyślny typ widoczności. Metody i pola oznaczone w ten sposób widoczne są dla wszystkich klas znajdujących się w pakiecie. Rozszerzeniem jest to iż pola i metody są widoczne dla wszystkich klas, które dziedziczą z danej klasy.

Slajd 6

JAVA
dla początkujących


```

public class PojazdOswietlony extends Pojazd {
    private boolean swiatla;

    public PojazdOswietlony(int droga, boolean swiatla) {
        super(droga);
        this.swiatla = swiatla;
    }

    public void wlaczSwiatla() {
        swiatla = true;
    }

    public void zeroj() {
        zerojLicznik();
    }
}
    
```



Interfejsy, klasy abstrakcyjne i dziedziczenie

Możemy skorzystać metody „zerojLicznik” z dwóch powodów. Po pierwsze, dziedziczymy po klasie „Pojzd”. Po drugie, klasa „PojazdOswietlony” znajduj się w tym samym pakiecie.



Slajd 7





JAVA
dla początkujących

```

class MojaPierwszaKlasa {

    public static void main(String[] args) {
        Pojazd pojazd = new Pojazd(0);
        pojazd.zerujLicznik();



        PojazdOswietlony pojazdOswietlony =
            new PojazdOswietlony(0, false);
        pojazdOswietlony.zerujLicznik();
        pojazdOswietlony.zeruj();
    }
}
    
```



Interfejsy, klasy abstrakcyjne i dziedziczenie

Możemy skorzystać metod „zerujLicznik” tylko dlatego, że jesteśmy w tym samym pakiecie. W innych pakietach dostępna będzie tylko metoda „zeruj”.


Slajd 8

JAVA
dla początkujących

Ćwiczenie:

Stwórz hierarchię klas dla następujących terminów: zwierzę, kot, pers, jaguar. Klasy nie muszą posiadać żadnych metod.





Interfejsy, klasy abstrakcyjne i dziedziczenie

```

public class Zwierze {};
public class Kot extends Zwierze{};
public class Pers extends Kot{};
public class Jaguar extends Kot {};
    
```



Slajd 9

JAVA
dla początkujących

- └─ pakiet1
 - └─ ObiektA.java
- └─ pakiet2
 - └─ Main.java
 - └─ ObiektB.java


```

package pakiet1;

public class ObiektA {

    protected void pokazWiadomosc() {
        System.out.println("Wiadomość!");
    }

}
    
```



Interfejsy, klasy abstrakcyjne i dziedziczenie

Aby lepiej pokazać działanie typu widoczności stworzone zostały dwa pakiety.

Klasa „ObiektA” znajduje się w osobnym pakiecie i zawiera metodę o właściwości „protected”.

Slajd 10




JAVA
dla początkujących

```

package pakiet2;

import pakiet1.ObiektA;

public class ObiektB extends ObiektA {

    public void pokazOdziedziczonaWiadomosc() {
        pokazWiadomosc();
    }

}
    
```



Interfejsy, klasy abstrakcyjne i dziedziczenie



Pierwszy wers oznacza, że bieżąca klasa znajduje się w pakiecie „pakiet2”.

„ObiektA” nie jest widoczny. Musi zostać zaimportowany z innego pakietu.

„ObiektB” dziedziczy z „ObiektA”, więc ma dostęp do jego metody chronionej („protected”).



Slajd 11

JAVA
dla początkujących

```

package pakiet2;


import pakiet1.ObiektA;

public class Main {

    public static void main(String[] args) {
        ObiektA obiektA = new ObiektA();
        // nie ma odpowiedniej metody dla obiektA

        ObiektB obiektB = new ObiektB();
        obiektB.pokazOdziedziczonaWiadomosc();
    }
}

```



Interfejsy, klasy abstrakcyjne i dziedziczenie

Znajdujemy się cały czas w pakiecie „pakiet2”.



Nie mamy dostępu do metody chronionej obiektu „ObiektA”.

Możemy ją jednak wywołać korzystając z „ObiektB”.

Slajd 12

Każda klasa w Javie dziedziczy po obiekcie „Object”, zawsze! „Object” posiada kilka metod i są one dostępne dla każdego stworzonego obiektu w Javie. Jedną z nich jest „toString”. Reprezentuje ona obiekt w postaci tekstu.

W metodzie „main” stworzony został obiekt typu „Dimension”. Obiekt ten przechowuje dwa podstawowe wymiary: wysokość i szerokość. Kiedy wywołamy na nim metodę „toString” otrzymamy tekstową reprezentację obiektu.

JAVA
dla początkujących

```

import java.awt.Dimension;

class MojaPierwszaKlasa {


    public static void main(String[] args) {
        Dimension dimension = new Dimension(2, 3);
        System.out.println(dimension.toString());

        System.out.println(new MojaPierwszaKlasa());
    }

    @Override
    public String toString() {
        return super.toString() + " - dziedzicze!";
    }
}

```

Wynik: java.awt.Dimension[width=2,height=3]
MojaPierwszaKlasa@1e859c0 - dziedzicze!




Interfejsy, klasy abstrakcyjne i dziedziczenie

Klasa „MojPierwszaKlasa” zawiera już metodę „toString”. Chcę jednak aby zwracała wartość ustawioną przeze mnie. Wyrażenie „@Override” to adnotacja. Oznacza, że obiekt odziedziczył już taką metodę, ale chcę napisać ją we własny sposób. Używając słowa „super” wywołuję metodę „toString” dla obiektu, po którym dziedziczyłem.




Metoda „println” domyślnie wywołuje metodę „toString” dla każdego obiektu, dlatego ostatnia instrukcja metody „main” działa w ten sposób.

Slajd 13



B2E
BUSINESS TO EDUCATION



SZCZECIŃSKI PARK
NAUKOWO-TECHNOLOGICZNY

JAVA
dla początkujących

```

Dimension dim1 = new Dimension(2, 3);
Dimension dim2 = new Dimension(2, 3);


if (dim1 == dim2) {
    System.out.println("Tak dla 1");
}

if (dim1.equals(dim2)) {
    System.out.println("Tak dla 2");
}

    public boolean equals(Object obj) {
        if (obj instanceof Dimension) {
            Dimension d = (Dimension)obj;
            return (width == d.width) && (height == d.height);
        }
        return false;
    }

```

Wynik:
Tak dla 2



Interfejsy, klasy abstrakcyjne i dziedziczenie

Stworzone zostały dwa z pozoru takie same obiekty. Zastanówmy się jednak. Obiekty mają takie same dane, ale zostały utworzone w niezależny sposób.



W pierwszej instrukcji „if” nie sprawdzamy równości obiektów! Sprawdzamy równość referencji. Warunek byłby prawdziwy, jeżeli obie referencje wskazywałyby na ten sam obiekt, a tak nie jest.

Dla sprawdzenia równości obiektów wywołujemy na jednym z nich metodę „equals”. Metoda ta również jest dziedziczona po typie „Object”.

Jeżeli chcemy porównać dwie instancje stworzonej przez nas klasy, to pamiętajmy aby napisać dla niej własną wersję metody „equals”. Dla przykładu została podana Metoda equals zaimplementowana w klasie Dimension.




Slajd 14



JAVA dla początkujących

Ćwiczenie:

Stwórz klasę „Animal” z metodami „eat” i „toString”. Niech „toString” zwraca napis „animal”. Stwórz klasy „Cat” i „Dog”. Dla tych klas niech metoda „toString” zwraca nazwę klasy.
Wywołaj „println” podając jako argument każdy z obiektów.

 Interfejsy, klasy abstrakcyjne i dziedziczenie

```
public class Animal {
    public void eat() {System.out.println("eat");}
    @Override
    public String toString() {return "animal";}
}
public class Cat extends Animal {
    @Override
    public String toString() {return "cat";}
}
public class Dog extends Animal {
    @Override
    public String toString() {return "dog";}
}
class MojaPierwszaKlasa {
    public static void main(String[] args) {
        System.out.println(new Animal());
        System.out.println(new Cat());
        System.out.println(new Dog());
    }
}
```



Slajd 15



JAVA
dla początkujących



W Javie możemy dziedziczyć tylko po jednej klasie!

Mam metody dla motorówki i dla samochodu. Jak stworzyć obiekt amfibia?




Interfejsy, klasy abstrakcyjne i dziedziczenie

Slajd 16



JAVA
dla początkujących

```
public interface Boat {  
    public void floatOnWater();  
}  
  
public interface Car {  
    public void drive();  
}
```



Interfejsy, klasy abstrakcyjne i dziedziczenie



W tym celu stosuje się interfejsy. Interfejs zawiera jedynie metody bez ciał. Tak jak widzimy na slajdzie. Dodatkowo możemy wstawić listę argumentów.

Interfejs przedstawia listę operacji, które może wykonać dany obiekt.

W najnowszej wersji Javy 1.8 (8) zostały dodane metody domyślne do interfejsów. Są to pełnoprawne metody, które będą dostępne w klasach implementujących dany interfejs. Nie mniej jednak jest to temat bardziej zaawansowany więc tą informację należy potraktować jako zachętę do poszerzenia tematu we własnym zakresie.



Slajd 17

JAVA
dla początkujących

```

public class Amphibian implements Boat, Car {

    @Override
    public void drive() {System.out.println("Jadę!");}


    @Override
    public void floatOnWater() {System.out.println("Płynę!");}

}

class MojaPierwszaKlasa {

    public static void main(String[] args) {
        Amphibian amphibian = new Amphibian();
        amphibian.drive();
        amphibian.floatOnWater();
    }

}
    
```





Interfejsy, klasy abstrakcyjne i dziedziczenie

Interfejsów się nie dziedziczy. Interfejsy implementujemy, tak jak jest to pokazane w pierwszym wersie.

Dopiero teraz możemy nadać ciałom metody.

Slajd 18

JAVA
dla początkujących


```

class MojaPierwszaKlasa {

    public static void main(String[] args) {
        Amphibian amphibian = new Amphibian();
        printCar(amphibian);
    }

    private static void printCar(Car car) {
        System.out.println(car);
    }

}
    
```




Interfejsy, klasy abstrakcyjne i dziedziczenie

Bieżący przykład pokazuje jak jest sens nadawania interfejsów obiektom.


Metoda „printCar” przyjmuje jako argument obiekt typu „Car”. Obiekt „Amphibian” implementuje ten interfejs, dlatego wywołanie tej metody jest poprawne.



Slajd 19



B2E
BUSINESS TO EDUCATION



SZCZECIŃSKI PARK
NAUKOWO-TECHNOLOGICZNY

JAVA
dla początkujących

```

public abstract class Ptak {


    private int waga;

    public int getWaga() {
        return waga;
    }

    public void setWaga(int waga) {
        this.waga = waga;
    }

    public abstract boolean isMieszony();

}
    
```



Interfejsy, klasy abstrakcyjne i dziedziczenie

Często spotykamy się z sytuacją gdy klasa bazowa zawiera wspólne pola i metody dla klas bardziej wyspecjalizowanych, ale zarazem jest zbyt ogólna aby móc sensownie wykorzystać instancje tej klasy. W powyższym przykładzie każdy Ptak ma swoją wagę, więc klasy pochodne np. „Wróbel” czy „Sokół” rozszerzając klasę bazową otrzymają jakby za darmo obsługę wagi. Niemniej jednak sam Ptak jest zbyt ogólny aby mógł reprezentować konkretny obiekt. W takim przypadku przychodzą nam z pomocą klasy abstrakcyjne. Za ich pomocą możemy zdefiniować wspólne składowe klasy (pole waga i metody), ukształtować wspólne API wszystkich klas wyspecjalizowanych (metoda isMieszony) oraz uniemożliwić stworzenie instancji obiektu tej klasy.

Aby stworzyć klasę abstrakcyjną należy użyć słowa „abstract” definiując klasę. Dodatkowo klasa może , ale nie musi zawierać metody abstrakcyjne. Metoda abstrakcyjna jest tylko definicją sygnatury metody i nie zawiera ciała metody. Metody abstrakcyjne mogą być zadeklarowane tylko w klasie abstrakcyjnej. Klasa dziedzicząca z klasy abstrakcyjnej musi implementować wszystkie abstrakcyjne metody klasy nadrzędnej, chyba że sama jest zadeklarowana jako abstrakcyjna. Nie mniej jednak pierwsze „konkretna” klasa w drzewie dziedziczenia będzie musiała zaimplementować wszystkie metody abstrakcyjne z klas w górze hierarchii dziedziczenia.

Klasie abstrakcyjnej jest trochę podobna do interfejsu, z tym że dodatkowo może zawierać pola i metody.



Slajd 20




JAVA
dla początkujących

```

public class Bocian extends Ptak {
    public boolean isMieszony() {
        return true;
    }
}

// niepoprawne wywołanie
// Ptak ptak = new Ptak();
Ptak ptak = new Bocian();
Bocian bocian = new Bocian();
Object object = new Bocian();
ptak.setWaga(5);
bocian.setWaga(5);
// to już nie jest prawidłowe
// object.setWaga(5);
    
```



Interfejsy, klasy abstrakcyjne i dziedziczenie

Dziedziczenie po klasie abstrakcyjnej odbywa się tak samo jak po zwykłej klasie. Nie można jednak stworzyć instancji klasy abstrakcyjnej, tak samo jak nie można stworzyć instancji interfejsu.

Na obiekt typu „Bocian” może wskazywać referencja typu: „Object”, „Bocian” lub „Ptak”. Należy pamiętać, że na referencji typu „Object” nie wywołamy metody „setWaga”.

Slajd 21




JAVA
dla początkujących

Ćwiczenie:

Stwórz interfejs „WiFi” z metodą „connect”.


Stwórz interfejs „Bluetooth” z metodą „connect”.

Sprawdź czy możesz stworzyć interfejs „Wireless” z taką metodą.


Może nie ma potrzeby pisania metody „connect” w dwóch pierwszych interfejsach?

Stwórz klasę abstrakcyjną „Phone”, niech posiada pole z numerem telefonu i metodę „call”.

Stwórz klasę „MobilePhone”, która dziedziczy po tych obiektach.




Człowiek - najlepsza inwestycja
Projekt współfinansowany przez Unię Europejską w ramach Europejskiego Funduszu Społecznego



Interfejsy, klasy abstrakcyjne i dziedziczenie

```

public interface Wireless {
    public void connect();
}

public interface WiFi extends Wireless {}
public interface Bluetooth extends Wireless {}
    
```



```
public abstract class Phone {
    private long number = 123123123;
    public long getNumber() {return number;}
    public void setNumber(long number) {this.number = number;}
    public void call() {}
}
public class MobilePhone extends Phone implements WiFi, Bluetooth {
    // ta metoda wystąpi tylko raz
    // Wniosek:
    // nie mamy osobnych metod dla WiFi i Bluetooth
    @Override
    public void connect() {}
}
class MojaPierwszaKlasa {
    public static void main(String[] args) {
        MobilePhone mobilePhone = new MobilePhone();
        mobilePhone.call();
        mobilePhone.connect();
    }
}
```

Opis założonych osiągnięć ucznia

Uczeń będzie umiał wykorzystać w praktyce interfejs i klasę abstrakcyjną, tworzyć nowe klasy wykorzystując wcześniej zaimplementowany kod.


Lekcja 5 Polimorfizm, instrukcja „final”, klasy wewnętrzne i anonimowe

Cel lekcji

Celem lekcji jest wytłumaczenie pojęcia polimorfizmu, oraz jego praktycznego zastosowania w Javie. Pokazanie jak tworzyć klasy wewnętrzne, oraz efektywnie posługiwać się interfejsami.

Treść - slajdy z opisem

Slajd 1



Slajd 1

B2E BUSINESS TO EDUCATION

SZCZECIŃSKI PARK NAUKOWO-TECHNOLOGICZNY

JAVA dla początkujących

Java - lekcja 5

Polimorfizm, instrukcja „final”, klasy wewnętrzne i anonimowe

Człowiek - najlepsza inwestycja

KAPITAŁ LUDZKI NARODOWA STRATEGIA SPÓJNOŚCI

UNIA EUROPEJSKA EUROPEJSKI FUNDUSZ SPOŁECZNY

Projekt współfinansowany przez Unię Europejską w ramach Europejskiego Funduszu Społecznego

DAILY



Slajd 2





JAVA dla początkujących

```
public abstract class Ptak {  
  
    private int waga;  
  
    public int getWaga() {  
        return waga;  
    }  
  
    public void setWaga(int waga) {  
        this.waga = waga;  
    }  
  
    @Override  
    public String toString() {  
        return "Jestem Ptakiem!";  
    }  
}
```

 Polimorfizm, instrukcja „final”, klasy wewnętrzne i anonimowe


Mamy prostą klasę "Ptak" z własną implementacją metody "toString".

Slajd 3



JAVA dla początkujących

```
public interface MaImie {  
  
    public String getName();  
}
```

 Polimorfizm, instrukcja „final”, klasy wewnętrzne i anonimowe

Istnieje również interfejs "MaImie" z metodą "getName".



Slajd 4

B2E
BUSINESS TO EDUCATION

SZCZECIŃSKI PARK
NAUKOWO-TECHNOLOGICZNY

JAVA
dla początkujących

```

public class Bocian extends Ptak implements MaImie {

    private String imie = "Kajtek";

    @Override
    public String getName() {
        return imie;
    }

    @Override
    public String toString() {
        return "Jestem bocianem o imieniu " + imie;
    }

}
    
```

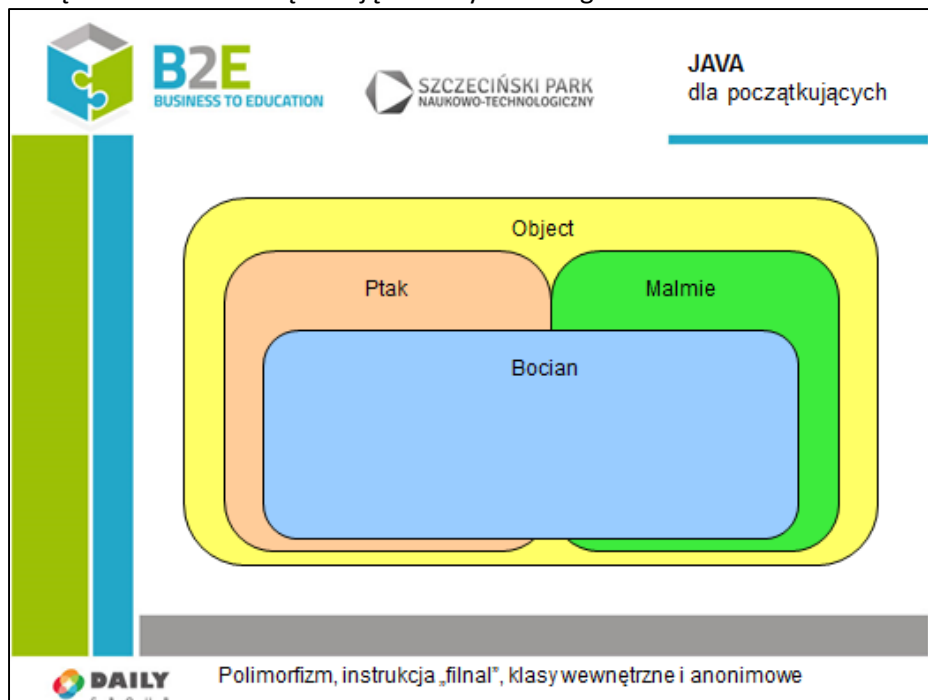
Polimorfizm, instrukcja „final”, klasy wewnętrzne i anonimowe

Stworzona jest klasa "Bocian", która rozszerza klasę "Ptak" i implementuje interfejs "MaImie".

Oczywiście w przypadku implementacji interfejsu, musimy zaimplementować wszystkie metody, które do tego interfejsu należą.


Bieżąca klasa ma własną wersję metody "toString".


Slajd 5





Slajd 6





JAVA
dla początkujących


```

class MojaPierwszaKlasa {

    public static void main(String[] args) {
        Bocian bocian = new Bocian();
        Ptak ptak = (Ptak) bocian;
        MaImie maImie = (MaImie) bocian;
        Object object = (Object) bocian;

        System.out.println(bocian);
        System.out.println(ptak);
        System.out.println(maImie);
        System.out.println(object);

        if (bocian == ptak) {
            System.out.println("Zachodzi równość!");
        }
    }
}
    
```



Polimorfizm, instrukcja_filna1, klasy wewnętrzne i anonimowe

W wyniku działania programu dostaniemy cztery razy napis "Jestem bocianem o imieniu Kajtek". I napis "Zachodzi równość".

Jak widać możemy przypisać obiekt "bocian" do Referencji czterech różnych typów: "Bocian", "Ptak", "Malmie", "Object". Zjawisko to nazywamy polimorfizmem (inaczej wielopostaciowość).

Obiekt możemy traktować na różne sposoby w zależności od potrzeb.



Zastanówmy się dlaczego dla referencji typu "Ptak" nie pokazał się napis "Jestem ptakiem!"?

Odpowiedź może zasugerować nam porównanie. Pamiętajmy, że nie sprawdzamy równości obiektów, na które wskazują referencje! Sprawdzamy, czy wskazują na ten sam obiekt. Wszystkie referencje w w metodzie "main" wskazują na obiekt stworzony przez konstruktor "Bocian()".

To, że metoda "toString" z klasy "Ptak" zwraca napis "Jestem ptakiem!", nie ma żadnego znaczenia. W klasie "Bocian" mamy zupełnie inną implementację tej metody. Klasa nie może posiadać dwóch metod o takiej samej nazwie i takiej samej liście argumentów.



Slajd 7





JAVA
dla początkujących

```

public class PolskiBocian extends Bocian {
    private String imie = "Polski bocian Bonifacy.";

    public static void main(String[] args) {
        Ptak ptak = new PolskiBocian();
        System.out.println(ptak.toString());
    }
}
    
```



Polimorfizm, instrukcja „final”, klasy wewnętrzne i anonimowe

Proszę zgadnąć co wyświetli powyższy program.

Wynik: Jestem bocianem o imieniu Kajtek

„Nadpisanie” pola imie nie skutkuje zmianą tekstu wypisywanego przez program.

Przykład ten demonstruje iż polimorfizm dotyczy tylko metod i nie ma analogicznego wpływu na pola.

Slajd 8




JAVA
dla początkujących

```

class MojaPierwszaKlasa {

    public static void main(String[] args) {
        Bocian bocian = new Bocian();

        if (bocian instanceof Ptak) {
            System.out.println("Bocian jest ptakiem!");
        }
    }
}
    
```





Polimorfizm, instrukcja „final”, klasy wewnętrzne i anonimowe

Aby móc rozpoznawać typ obiektu możemy skorzystać z operatora "instanceof". W powyższym przykładzie widzimy jego użycie i jak można się domyślić wynikiem działania operatora jest wartość logiczna.



Slajd 9





JAVA
dla początkujących


Ćwiczenie:

Stwórz dwie klasy:

Liczba, niech przechowuje dowolną liczbę i posiada metodę „podajWartosc”.

Napis, niech przechowuje dowolny tekst i posiada metodę „podajTekst”.

W dowolnym obiekcie, który posiada metodę „main” stwórz metodę „wyswietl”, która będzie przyjmowała jako argument obiekt typu „Object”. W przypadku „Liczba” wywołaj metodę „podajWartosc” i analogicznie w przypadku „Napis”.





Polimorfizm, instrukcja „final”, klasy wewnętrzne i anonimowe

```
public class Liczba {
    private int liczba = 123;
    public int podajWartosc() {return liczba;}
}
public class Napis {
    private String napis = "Bla, bla, bla...";
    public String podajTekst() {return napis;}
}
class MojaPierwszaKlasa {
    public static void main(String[] args) {
        wyswietl(new Liczba());
        wyswietl(new Napis());
        wyswietl(new Object());
    }
    private static void wyswietl(Object object) {
        if (object instanceof Liczba) {
            System.out.println(((Liczba)object).podajWartosc());
        }
        else if (object instanceof Napis) {
            System.out.println(((Napis)object).podajTekst());
        }
        else {
            System.out.println("Nie rozpoznano obiektu!");
        }
    }
}
```



Slajd 10






JAVA

dla początkujących

Ćwiczenie:

Stwórz klasę „Temperatura” z metodą „podajTemperature”. Wykorzystaj klasę „Liczba” z poprzedniego ćwiczenia do przechowywania wartości poprzez mechanizm dziedziczenia. Popraw metodę „wyswietl”, aby poprawnie wywoływała metodę w zależności od podanego typu obiektu.



Polimorfizm, instrukcja „final”, klasy wewnętrzne i anonimowe

```
public class Temperatura extends Liczba {
    private char jednostka = 'C';
    public String podajTemperature() {
        return Integer.toString(podajWartosc())
            + Character.toString(jednostka);
    }
}

class MojaPierwszaKlasa {

    public static void main(String[] args) {
        wyswietl(new Liczba());
        wyswietl(new Napis());
        wyswietl(new Object());
        wyswietl(new Temperatura());
    }

    private static void wyswietl(Object object) {
        if (object instanceof Liczba) {
            if (object instanceof Temperatura) {
                System.out.println(((Temperatura)object).podajTemperature());
            }
            else {
                System.out.println(((Liczba)object).podajWartosc());
            }
        }
        else if (object instanceof Napis) {
            System.out.println(((Napis)object).podajTekst());
        }
        else {
```



```
System.out.println("Nie rozpoznano obiektu!");
```

```
}
```

```
}
```

```
}
```

Slajd 11



Slide 11 content includes logos for B2E (Business to Education), Szczeciński Park Naukowo-Technologiczny, and DAILY GROUP. The title is 'JAVA dla początkujących'. The exercise text is: 'Ćwiczenie: Stwórz taką klasę: public class BocianBialy extends Bocian { } Następnie zmodyfikuj klasę „Bocian” dodając słowo „final” jak w przykładzie poniżej. public final class Bocian extends Ptak implements MaImie { } Jaki problem wystąpił w aplikacji?'. The footer text is 'Polimorfizm, instrukcja „final”, klasy wewnętrzne i anonimowe'.

Słowo "final" użyte w definicji klasy oznacza, że nie możemy po niej dziedziczyć.

Slajd 12




Slide 12 content includes logos for B2E (Business to Education), Szczeciński Park Naukowo-Technologiczny, and DAILY GROUP. The title is 'JAVA dla początkujących'. The exercise text is: 'Ćwiczenie: Dodaj „final” do metody „toString” w klasie „Bocian”. Czy „BocianBialy” może posiadać własną implementację tej metody?'. The footer text is 'Polimorfizm, instrukcja „final”, klasy wewnętrzne i anonimowe'.

Pisanie własnych wersji metod nazywamy przesłanianiem. Nie można przesłonić metody oznaczonej jako final. Jest to ostateczna implementacja.



Slajd 13


JAVA
dla początkujących

```

public class Drzewo {
    private String gatunek = "Klon";

    public class Lisc {
        @Override
        public String toString() {
            return "Jestem liściem drzewa " + gatunek;
        }
    }

    public Lisc getLisc() {
        return new Lisc();
    }
}
    
```





Polimorfizm, instrukcja „final”, klasy wewnętrzne i anonimowe

Java umożliwia tworzenie klas wewnętrznych. Klasa wewnętrzna ma dostęp do elementów prywatnych klasy, w której się znajduje.

Klasa wewnętrzna może posiadać typy widoczności: prywatny, chroniony, publiczny i domyślny (pakietowy).

Slajd 14





JAVA
dla początkujących

```

class MojaPierwszaKlasa {
    public static void main(String[] args) {
        Drzewo drzewo = new Drzewo();
        Drzewo.Lisc lisc = drzewo.new Lisc();
        System.out.println(lisc);
    }
}
    
```

Wynik:
Jestem liściem drzewa Klon



Polimorfizm, instrukcja „final”, klasy wewnętrzne i anonimowe

Tak wygląda prawidłowe wywołanie klasy wewnętrznej. Zwróćmy uwagę jak wygląda użycie operatora "new" dla obiektu "Lisc".

Dlaczego nie można stworzyć obiektu typu "Lisc" bez użycia obiektu "Drzewo"?

Obiekt typu "Lisc" może mieć dostęp do pól obiektu typu "Drzewo". Dlatego aby stworzyć "lisc" musimy stworzyć najpierw obiekt "drzewo".



Slajd 15




JAVA
dla początkujących

```

public class Miasto {
    public class Ulica {
        public class Dom {
            @Override
            public String toString() {
                return Ulica.this.toString() + " Dom";
            }
        }
        @Override
        public String toString() {
            return Miasto.this.toString() + " Ulica";
        }
    }
    @Override
    public String toString() {
        return "Miasto";
    }
}
    
```



Polimorfizm, instrukcja „final”, klasy wewnętrzne i anonimowe

Widoczny przypadek jest nieco bardziej skomplikowany. Jak widać każda z klas może mieć własną metodę "toString".

Zwróć uwagę jak następuję wywołanie metody z klas zewnętrznych,

Slajd 16




JAVA
dla początkujących

```

Miasto miasto = new Miasto();
Miasto.Ulica ulica = miasto.new Ulica();
Miasto.Ulica.Dom dom = ulica.new Dom();
System.out.println(dom);
    
```

Wynik:

Miasto Ulica Dom





Polimorfizm, instrukcja „final”, klasy wewnętrzne i anonimowe

Obiekty tworzymy dokładnie tak jak w poprzednim przykładzie.

Aby zbudować wynik wywołane zostały metody z wszystkich trzech obiektów.



Slajd 17

JAVA
dla początkujących

```

public abstract class Ptak {


    private int waga;

    public int getWaga() {
        return waga;
    }

    public void setWaga(int waga) {
        this.waga = waga;
    }

    @Override
    public String toString() {
        return "Jestem Ptakiem!";
    }
}

public interface MaImie {
    public String getName();
}
    
```




Polimorfizm, instrukcja „final”, klasy wewnętrzne i anonimowe

Skorzystajmy z klasy abstrakcyjnej i interfejsu, których używaliśmy wcześniej.

W poprzedniej lekcji podane było, że nie możemy stworzyć instancji danej klasy abstrakcyjnej lub interfejsu. Od tej reguły nie ma wyjątku, ale nie musimy deklarować wprost nowego typu klasy aby stworzyć instancję obiektu dziedziczącego z klasy abstrakcyjnej. W praktyce możemy stworzyć klasę anonimową, co zostanie pokazane na następnym slajdzie. Podobnie możemy postąpić z interfejsem.

Slajd 18




JAVA
dla początkujących

```

class MojaPierwszaKlasa {

    public static void main(String[] args) {
        Ptak ptak = new Ptak() {
        };
        ptak.setWaga(10);

        MaImie maImie = new MaImie() {
            @Override
            public String getName() {
                return "Moje imie";
            }
        };
        System.out.println(maImie.getName());
    }
}
    
```



Polimorfizm, instrukcja „final”, klasy wewnętrzne i anonimowe

"Ptak" jest klasą abstrakcyjną. W nawiasach klamrowych możemy dodać dodatkowe metody i pola do klasy. W ten sposób stworzony został obiekt anonimowy. Nie możemy odwołać się do klasy, której obiekt jest instancją. Klasa ta nie ma nazwy, ale wiemy, że




zgodnie z polimorfizmem, jest typu "Ptak".


W przypadku interfejsu sprawa jest nieco bardziej skomplikowana. Nie ma on ciał swoich metod, musimy więc je zaimplementować.

Klasy anonimowe są często wykorzystywane gdy istnieje potrzeba jednorazowego wykorzystania ich funkcjonalności. Interfejs graficzny użytkownika oraz obsługa zdarzeń zostaną omówione w późniejszej lekcji. Niemniej jednak jest to dobry przykład wykorzystania klas anonimowych. Powiedzmy że mamy dwa przyciski „Ok.” i „Cancel” obydwa z nich reagują na kliknięcie, ale ich zachowanie jest zupełnie odmienne. Wobec czego interfejs obsługujący kliknięcie jest ten sam (metoda „onClick”), ale implementacja będzie różna. Dodatkowo nie ma innych elementów GUI które powinny obsłużyć zdarzenie dokładnie w ten sam sposób, więc nie ma sensu tworzenia nowych klas np. OnClickListener i CancelClickListener. Dlatego lepiej jest użyć dwóch klas anonimowych, aby implementując ten sam interfejs wykonały różne zadania.

Slajd 19



B2E
BUSINESS TO EDUCATION



SZCZECIŃSKI PARK
NAUKOWO-TECHNOLOGICZNY


JAVA
dla początkujących

Ćwiczenie:


Korzystając z klasy anonimowej stworzonej z interfejsu zaprojektuj mechanizm sprawdzający ile czasu trwa wykonanie dowolnej metody. Czas w postaci long (milisekundy) zwróci Ci wywołanie:

```
System.currentTimeMillis();
```

Człowiek - najlepsza inwestycja




KAPITAŁ LUDZKI
NARODOWA STRATEGIA SPÓJNOŚCI



UNIA EUROPEJSKA
EUROPEJSKI
FUNDUSZ SPOŁECZNY

Projekt współfinansowany przez Unię Europejską w ramach Europejskiego Funduszu Społecznego



DAILY
GROUP

Polimorfizm, instrukcja „final”, klasy wewnętrzne i anonimowe

```
public interface CustomOperation {
    public void execute();
}

class MojaPierwszaKlasa {
    public static void main(String[] args) {
        System.out.println(checkTime(new CustomOperation() {
            @Override
            public void execute() {
                myOperation();
            }
        }));
    }

    private static long checkTime(CustomOperation customOperation) {
        long l = System.currentTimeMillis();
        customOperation.execute();
    }
}
```



```
        return System.currentTimeMillis() - l;  
    }  
    private static void myOperation() {  
        int i = 1;  
        while (i>0) {  
            i++;  
        }  
    }  
}
```

Opis założonych osiągnięć ucznia

Po tej lekcji uczniowie będą wiedzieć jak wykorzystuje się w praktyce polimorfizm i interfejsy.

Lekcja 6 Typ wyliczeniowy, JavaDocs

Cel lekcji

Celem lekcji jest zapoznanie uczniów z typem wyliczeniowym i prostą metodą dokumentowania kodu za pomocą JavaDocs.

Treść - slajdy z opisem

Slajd 1



Slajd 1 zawiera następujące elementy:

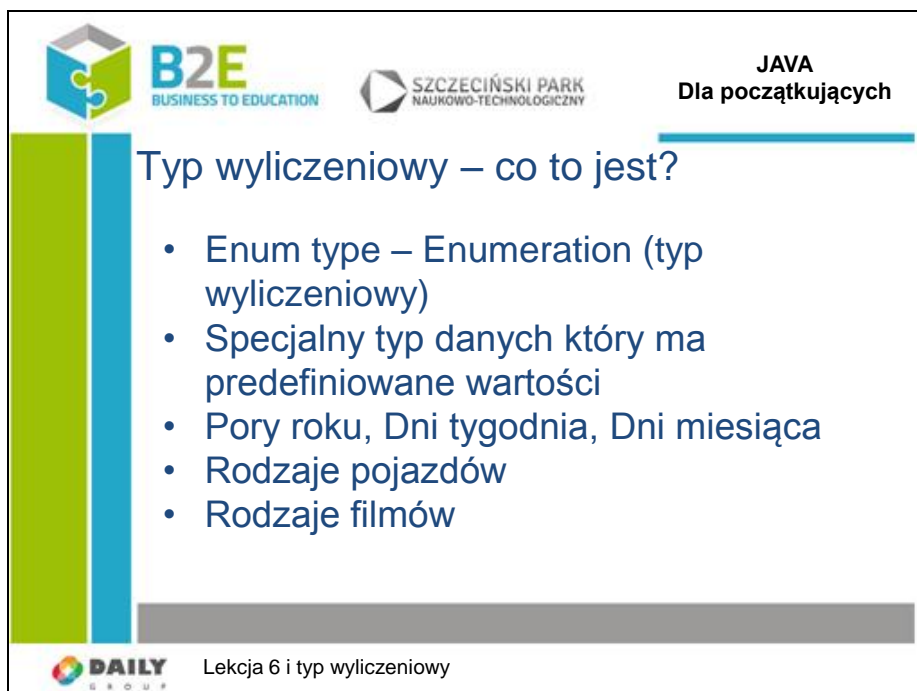
- Logotypy: B2E BUSINESS TO EDUCATION, SZCZECIŃSKI PARK NAUKOWO-TECHNOLOGICZNY, KAPITAŁ LUDZKI NARODOWA STRATEGIA SPÓJNOŚCI, UNIA EUROPEJSKA EUROPEJSKI FUNDUSZ SPOŁECZNY.
- Tytuł: **Java - lekcja 6**
- Podtytuły: Typ wyliczeniowy, JavaDocs
- Podtytuł: JAVA dla początkujących
- Logo DAILY GROUP
- Informacja: Człowiek - najlepsza inwestycja
- Informacja: Projekt współfinansowany przez Unię Europejską w ramach Europejskiego Funduszu Społecznego

Na lekcji poznamy co to jest typ wyliczeniowy, do czego znajduje zastosowanie i jak można go wykorzystać w programowaniu.

Poznamy narzędzia do dokumentowania kodu



Slajd 2

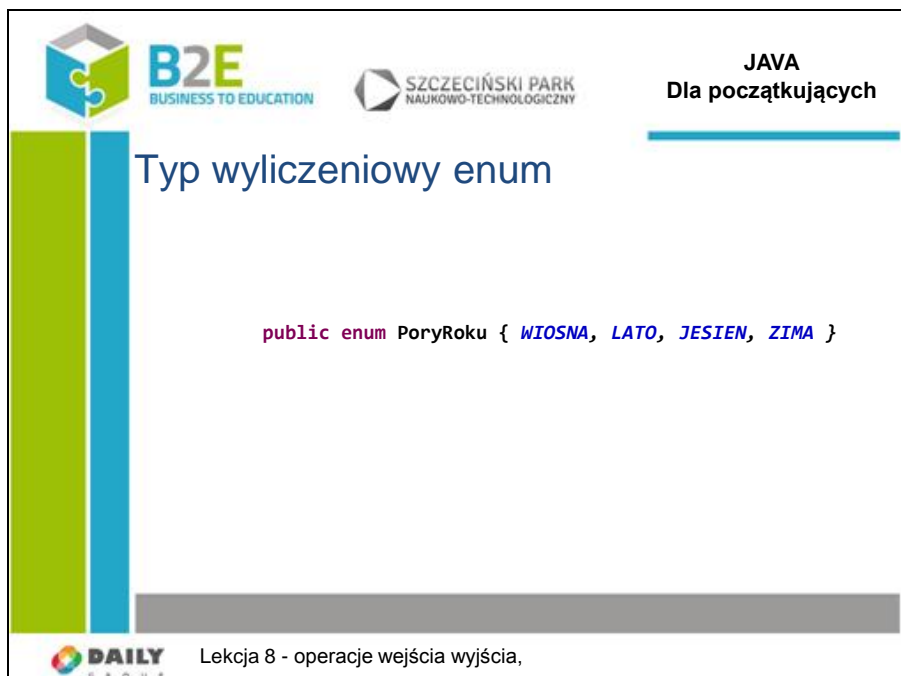


Logo B2E BUSINESS TO EDUCATION and SZCZECIŃSKI PARK NAUKOWO-TECHNOLOGICZNY are visible in the top left. The title 'JAVA Dla początkujących' is in the top right. The main text asks 'Typ wyliczeniowy – co to jest?' and lists examples: Enum type – Enumeration (typ wyliczeniowy), Specjalny typ danych który ma predefiniowane wartości, Pory roku, Dni tygodnia, Dni miesiąca, Rodzaje pojazdów, and Rodzaje filmów. The footer includes the DAILY GROUP logo and 'Lekcja 6 i typ wyliczeniowy'.

Typ wyliczeniowym jest specjalny typ danych który może przyjmować jedną z zadeklarowanych stałych.

Przykładem może być pora roku. Załóżmy że chcemy aby zmienna poraRoku przyjmowała tylko ustalone przez nas wartości tj. wiosna, lato, jesien i zima.

Slajd 3



Logo B2E BUSINESS TO EDUCATION and SZCZECIŃSKI PARK NAUKOWO-TECHNOLOGICZNY are visible in the top left. The title 'JAVA Dla początkujących' is in the top right. The main text asks 'Typ wyliczeniowy enum' and shows the code: `public enum PoryRoku { WIOSNA, LATO, JESIEN, ZIMA }`. The footer includes the DAILY GROUP logo and 'Lekcja 8 - operacje wejścia wyjścia,'.

Przygotujemy nową uruchamialną klasę i zadeklarujemy następujący typ zmiennej: PoryRoku który będzie przyjmować następujące wartości **public enum PoryRoku { WIOSNA, LATO, JESIEN, ZIMA }**



Slajd 4

B2E
BUSINESS TO EDUCATION

SZCZECIŃSKI PARK
NAUKOWO-TECHNOLOGICZNY

JAVA
Dla początkujących

Typ wyliczeniowy enum

```

public enum PoryRoku { WIOSNA, LATO, JESIEN, ZIMA }

public static void main(String[] args) {
// TODO Auto-generated method stub

PoryRoku poryRoku = PoryRoku.JESIEN;
                
```

Lekcja 8 - operacje wejścia wyjścia,

Przygotujemy nową uruchamialną klasę i zadeklarujemy następujący typ zmiennej:

PoryRoku który będzie przyjmować następujące wartości

public enum PoryRoku { WIOSNA, LATO, JESIEN, ZIMA }

W metodzie main zadeklarujemy zmienną typu PoryRoku i przypiszmy jej wartość.

Zauważmy, że wpisując wartość możemy posługiwać się zadeklarowanym typem PoryRoku.

Slajd 5

B2E
BUSINESS TO EDUCATION

SZCZECIŃSKI PARK
NAUKOWO-TECHNOLOGICZNY

JAVA
Dla początkujących

Typ wyliczeniowy enum

```

public enum PoryRoku { WIOSNA, LATO, JESIEN, ZIMA }

public static void main(String[] args) {
    PoryRoku.
}
                
```

class : Class<Ptak.PoryRoku>

JESIEN : Ptak.PoryRoku - Ptak.PoryRoku

LATO : Ptak.PoryRoku - Ptak.PoryRoku

WIOSNA : Ptak.PoryRoku - Ptak.PoryRoku

ZIMA : Ptak.PoryRoku - Ptak.PoryRoku

valueOf(String arg0) : PoryRoku - PoryRoku

values() : PoryRoku[] - PoryRoku

valueOf(Class<T> enumType, String name) : T - Enum

Lekcja 8 - operacje wejścia wyjścia,

Zauważmy, że wpisując wartość możemy posługiwać się zadeklarowanym typem PoryRoku. Eclipse podpowiada nam wartości mechanizmem intellisense.



Slajd 6

B2E
BUSINESS TO EDUCATION

SZCZECIŃSKI PARK
NAUKOWO-TECHNOLOGICZNY

JAVA
Dla początkujących

Typ wyliczeniowy enum - zadanie

- Proszę napisać klasę która na podstawie przekazanej konstruktorowi pory roku ustali czy lubisz porę roku.
- Metoda czyLubie() niech wydrukuje / wyświetli informację czy zadana pora roku jest lubiana czy nie.

Lekcja 8 - operacje wejścia wyjścia,

Proszę wykorzystać instrukcję switch/case

Dla uproszczenia klasę można zadeklarować jak zagnieżdżoną klasę programu zamiast nowego pliku dla klasy.

Inicjacja obiektu powinna następować wg przykładu:

LubiePoryRoku w = LubiePoryRoku(PoryRoku.JESIEN);

Slajd 7

B2E
BUSINESS TO EDUCATION

SZCZECIŃSKI PARK
NAUKOWO-TECHNOLOGICZNY

JAVA
Dla początkujących

Typ wyliczeniowy enum - rozwiązanie

```
public class LubiePoryRoku {
    PoryRoku poryRoku;

    LubiePoryRoku(PoryRoku p) {
        this.poryRoku = p;
    }

    public void czyLubie(){
        System.out.print("Cześć czy lubię porę roku "+
            poryRoku.toString()+"? ");
        switch (poryRoku) {
            case WIOSNA: System.out.println("Lubię wiosnę");break;
            case LATO: System.out.println("Bardzo lubię lato");break;
            case JESIEN: System.out.println("eech tak sobie");break;
            default: System.out.println("byłoby padał śnieg");break;}
    }
}
```

Lekcja 8 - operacje wejścia wyjścia,

Klasa nazywa się LubiePoryRoku

Jej konstruktor zapamiętuje zadany przy tworzeniu obiektu klasy zmienną typu PoraRoku w polu klasy.

W metodzie czyLubie zastosowano instrukcję switch / case

Proszę zauważyć jak łatwo można wykorzystać typ wyliczeniowy do obsługi poszczególnych przypadków tego typu



Dodatkowo, proszę zwrócić uwagę na konwersję wartości nazw stałych w instrukcji wyświetlającej napis „Cześć czy lubię porę roku "+poryRoku.toString()+"? „

Slajd 8

B2E BUSINESS TO EDUCATION

SZCZECIŃSKI PARK NAUKOWO-TECHNOLOGICZNY

JAVA
Dla początkujących

Typ wyliczeniowy enum - rozwiązanie

```
public static void main(String[] args) {
    // TODO Auto-generated method stub

    LubiePoryRoku w = new LubiePoryRoku(PoryRoku.ZIMA);
    w.czyLubie();
}
```

DAILY GROUP Lekcja 8 - operacje wejścia wyjścia.

Wywołanie klasy zadeklarowanej jako klasę wewnętrzną można wykonać w zaprezentowany sposób.

Proszę zauważyć że konstruktor wołany jest ze stałą ustawianą typem PoryRoku.

Slajd 9

B2E BUSINESS TO EDUCATION

SZCZECIŃSKI PARK NAUKOWO-TECHNOLOGICZNY

JAVA
dla początkujących

JavaDocs – dokumentacja kodu

- Czy dokumentacja jest potrzebna?
- Jak dokumentować?
- JavaDocs – narzędzie do dokumentowania
- Korzyści

DAILY GROUP Lekcja 6 i typ wyliczeniowy

Czy dokumentacja kodu jest potrzebna?

Odp: Jest potrzebna. Wyobraźmy sobie że piszemy kod od dłuższego czasu. Powrót do napisanego kodu sprzed pół roku może okazać się trudny.

Krótki komentarz, opis zaraz przybliży nam do czego służy metoda, kto jaką wersję klasy i



kiedy napisał

Jak dokumentować?

Oczywiście nie chodzi o pisanie wielostronicowych elaboratów. Opis ma być krótki, zrozumiały, zawierać tylko to co chcielibyśmy wiedzieć wykorzystując np. klasę. Najlepiej wg określonego szablonu

W lekcji pierwszej omówiliśmy komentarze w kodzie źródłowym:

// do końca linii

/* */ wieloliniowy

Ich zadaniem jest dokumentacja danego fragmentu kodu, ułatwienie zrozumienia algorytmu. Odbiorcami są developerzy czytający kod lub go modyfikujący.

Zachodzi jednak czasem potrzeba udokumentowania API biblioteki, programu który stworzyliśmy. Często użytkownik, również developer może nie mieć dostępu do kodu źródłowego, lub po prostu chce użyć naszych narzędzi bez zagłębiania się w szczegóły. Do tworzenia tego rodzaju dokumentacji używa się JavaDocs.

JavaDocs jest specjalnym rodzajem komentarza umieszczanym w kodzie źródłowym. Pozwala on na wygenerowanie dokumentacji technicznej naszego kodu. Dzięki tego rodzaju komentarzom czytelnik dokumentacji może dowiedzieć się jakie klasy, metody i pola ma dostępne, opis każdego z nich. Ponieważ jest to ogólnoprzyjęty sposób dokumentacji środowiska dewelopersie wspierają go. I tak np. Eclipse pomaga nam w tworzeniu JavaDoc w kodzie, w popup'ach wyświetla informacje po najechaniu na interesujący element. Źródłem tych informacji może być nasz własny udokumentowany kod lub dokumentacja osobno ściągnięta do używanej biblioteki (nawet bez konieczności posiadania kodu źródłowego biblioteki).

Slajd 10

B2E BUSINESS TO EDUCATION

SZCZECIŃSKI PARK NAUKOWO-TECHNOLOGICZNY

JAVA
Dla początkujących

JavaDocs – dokumentacja kodu

```
System.out.println("Kwadrat liczby to: "+a*a+".");
try {
    // zdefiniuj Buffer
    for(int i=0; i<a; i++)
        System.out.println("Kwadrat liczby to: "+i*i+".");
}
```

void java.io.PrintStream.println(String x)
Prints a String and then terminate the line. This method behaves as though it invokes `print(String)` and then `println()`.

Parameters:
x The String to be printed.

@ Javadoc

DAILY Lekcja 6 i typ wyliczeniowy

Wystarczy że najedziemy na metodę (tutaj println) i otrzymujemy szybką informację jak dana metoda działa, jakie ma parametry. Dzięki temu nie musimy szukać informacji w dokumentacji.



Slajd 11

B2E BUSINESS TO EDUCATION

SZCZECIŃSKI PARK NAUKOWO-TECHNOLOGICZNY

JAVA
Dla początkujących

JavaDocs – dokumentacja kodu

```
public static void main(String[] args) {  
    LubiePoryRoku lpr = new LubiePoryRoku();  
}
```

LubiePoryRoku

Press 'F2' for focus

@ javadoc: XX Declaration Search

DAILY GROUP Lekcja 6 i typ wyliczeniowy

A oto nasz ostatni kod (klasa LubiePoryRoku).

Po najechnaniu na nazwę klasy nie wyświetlają się żadne podręczne informacje.

Slajd 12

B2E BUSINESS TO EDUCATION

SZCZECIŃSKI PARK NAUKOWO-TECHNOLOGICZNY

JAVA
Dla początkujących

JavaDocs – jak dokumentować

- Komentarz
- Standard

```
/**  
 * Funkcja obliczająca kwadrat liczby  
 * @author karol.kowalski  
 * @version 1.1  
 */
```

- Predefiniowane tagi @

DAILY GROUP Lekcja 6 i typ wyliczeniowy

Dokumentacja kodu jest wykonywana w formie komentarza przed typem zmiennej, klasą, metodą.

Aby odróżnić dokumentację od zwykłego komentarza stosuje się znaczniki



```
/**
```

```
*/
```

Ustalono standard opisu i tagi



Slajd 13





JAVA
Dla początkujących

JavaDocs – przykład

```

/**
 * Lista pór roku w naszej szerokości geograficznej
 *
 *
 * @author karol.kowalski
 * @version 1.0
 * @see PoryRokuExt
 *
 */
public enum PoryRoku { WIOSNA, LATO, JESIEN, ZIMA }
    
```





Lekcja 6 i typ wyliczeniowy

Proszę przyjrzeć się przykładowi który dokumentuje ostatnio zdefiniowany typ wyliczeniowy PoryRoku.

Dzięki dołączonej dokumentacji każde użycie typu PoryRoku powoduje podręczne wyświetlenie informacji.

Slajd 14

JAVA
Dla początkujących


JavaDocs – przykład

```

/**
 *
 * @author manfred.poznanski
 * @
 *
 */
public
{
    
```

@author - author name

- @author
- @category
- @deprecated
- @see
- @serial
- @since
- @version
- @{@code}
- @{@docRoot}
- @{@linkplain}
- @{@link}
- @{@literal}
- @{@value}
- @ejb.bean
- @ejb.ejb-external-ref



Lekcja 6 i typ wyliczeniowy

Narzędzie Eclipse wspiera przy pisaniu dokumentacji – lista dostępnych tagów



Slajd 15

B2E BUSINESS TO EDUCATION

SZCZECIŃSKI PARK
NAUKOWO-TECHNOLOGICZNY

JAVA
Dla początkujących

JavaDocs – zasady

```
/**  
 * Krótki opis klasy LubiePoryRoku  
 * <p>  
 * Użyj {@link #LubiePoryRoku\(PoryRoku\)} aby ustawić porę roku.  
 *  
 * @param PoryRoku zmienna typu PoryRoku  
 * @return konstruktor nic nie zwraca  
 *  
 * @author manfred.poznanski  
 * @version 1.1  
 * @since 2013-02-02  
 */
```

DAILY GROUP Lekcja 6 i typ wyliczeniowy

Przyjęto że pierwszą linią dokumentującą będzie krótki opis co dany element wykonuje
Następnie można dodać drugą linię (po znaczniku nowej linii <p>) zawierający opis metody i jej parametrów, oraz co zwraca metoda
Potem, zależnie od uznania tagi dot. autora, wersji daty wprowadzenia.

Slajd 16

B2E BUSINESS TO EDUCATION

SZCZECIŃSKI PARK
NAUKOWO-TECHNOLOGICZNY

JAVA
Dla początkujących

JavaDocs – podgląd

```
13     * @see PoryRokuExt  
14     *  
15     *  
16     */  
17     public enum PoryRoku { WIOSNA, LATO, JESIEN, ZIMA }  
--
```

Problems @ Javadoc Declaration Search Console Progress

LubiePoryRoku.PoryRoku

Lista pór roku w naszej szerokości geograficznej

Version: 1.0
Author: karol.kowalski
See Also: [PoryRokuExt](#)

DAILY GROUP Lekcja 6 i typ wyliczeniowy

Podgląd pisanej dokumentacji można śledzić na zakładce Javadoc



Slajd 17

The screenshot shows an IDE with a Java code editor and a tooltip for the `LubiePoryRoku.PoryRoku` class. The code in the editor is:

```
public static void main(String[] args) {  
    PoryRoku a = null;  
}
```

The tooltip displays the following information:

- LubiePoryRoku.PoryRoku**
- Lista pór roku w naszej szerokości geograficznej
- Version:** 1.0
- Author:** karol.kowalski
- See Also:** [PoryRokuExt](#)

Logos for B2E (Business to Education) and Szczeciński Park Naukowo-Technologiczny are visible at the top. The slide title is "JAVA Dla początkujących". At the bottom, it says "Lekcja 6 i typ wyliczeniowy".

Oto widok po najechnaniu na nazwę typu PoryRoku

Slajd 18

The screenshot shows a slide titled "JAVA Dla początkujących" with the subtitle "JavaDocs – zadanie". The task is described as follows:

- Proszę napisać dokumentację do ostatnio dodanej klasy `LubiePoryRoku`
- Proszę opisać konstruktor
- Oraz metodę `czyLubie()`
- Proszę sprawdzić czy przy pisaniu oprogramowania wyświetla się dokumentacja

Logos for B2E (Business to Education) and Szczeciński Park Naukowo-Technologiczny are visible at the top. At the bottom, it says "Lekcja 6 i typ wyliczeniowy".

Oto widok po najechnaniu na nazwę typu PoryRoku



Slajd 19




JAVA
Dla początkujących

JavaDocs – rozwiązanie zadania

```


/**
 * Klasa LubiePoryRoku
 * <p>
 * Implementuje zaawansowany algorytm ekspercki do ustalania
 * preferencji pór roku użytkownika.<p>
 * Konstruktor {@link #LubiePoryRoku}
 * ustawia wybraną porę roku.<p>
 * Metoda {@link #czyLubie\(\)} zwraca preferencje
 *
 *
 * @author manfred.poznanski
 * @version 1.1
 * @since 2013-02-02
 *
 */
public class LubiePoryRoku
    
```



Lekcja 6 i typ wyliczeniowy

Oto przykładowa dokumentacja klasy
Proszę zwrócić uwagę na linki do poszczególnych metod klasy

Slajd 20

JAVA
Dla początkujących

JavaDocs – rozwiązanie zadania

```

public static void main(String[] args) {
    LubiePoryRoku lpr = new LubiePoryRoku(PoryRoku.JESIEN);
}
    
```

LubiePoryRoku

Klasa [LubiePoryRoku](#)


Implementuje zaawansowany algorytm ekspercki do ustalania preferencji pór roku uz

Problemy @J Konstruktor [Lubie_pory_roku](#) ustawia wybraną porę roku.

Metoda [czyLubie\(\)](#) zwraca preferencje

Since:
2013-02-02

Version:





Lekcja 6 i typ wyliczeniowy

Oto przykładowa dokumentacja klasy
Proszę zwrócić uwagę na linki do poszczególnych metod klasy




Slajd 21



JAVA
Dla początkujących

JavaDocs – rozwiązanie zadania

```
/**
 * Krótki opis konstruktora LubiePoryRoku
 * <p>
 * Użyj {@link #LubiePoryRoku\(PoryRoku\)}
 * aby ustawić porę roku.
 *
 * @param PoryRoku zmienna typu PoryRoku
 * @return konstruktor nic nie zwraca
 *
 * @author manfred.poznanski
 * @version 1.1
 * @since 2013-02-02
 */
LubiePoryRoku(PoryRoku p) {
```



Lekcja 6 i typ wyliczeniowy

Slajd 22



JAVA
Dla początkujących

JavaDocs – rozwiązanie zadania



```
/**
 * Metoda wyświetlająca preferencje pór roku użytkownika
 * <p>
 * Metoda jest bezparametrowa \(wykorzystuje pole klasy
PoryRoku
 *
 * @return przesyła na standardowe urządzenie wyjściowe
\(console\) raport z preferencjami
 *
 * @author manfred.poznanski
 * @version 1.1
 * @since 2013-02-02
 */
public void czyLubie\(\){
```



Lekcja 6 i typ wyliczeniowy



Slajd 23





JAVA
Dla początkujących

JavaDocs – rozwiązanie zadania

```



2+ * Klasa LubiePoryRoku
14 public class LubiePoryRoku {
15
16     private PoryRoku poraRoku;
17
19+ * Lista pór roku w naszej szerokości geograficznej
28 public enum PoryRoku { WIOSNA, LATO, JESIEN, ZIMA }
29
31+ * Krótki opis konstruktora LubiePoryRoku
44 public LubiePoryRoku(PoryRoku poraRoku) {
45     this.poraRoku = poraRoku;
46 }
47
48 public static void main(String[] args) {
49     LubiePoryRoku lpr = new LubiePoryRoku(PoryRoku.JESIEN);
50 }
51
52 }
    
```



Lekcja 6 i typ wyliczeniowy

Dokumentacja nie musi zaśmiecać kodu – można ją zwinąć żeby nie przeszkadzała do jednej linii.

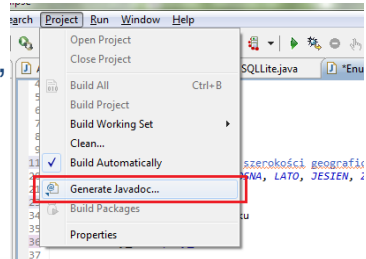
Slajd 24





JAVA
Dla początkujących

JavaDocs – dokumentacja html

- Cały projekt wraz z dokumentacją można zapisać w HTML-u
- W tym celu należy wybrać opcję „Generate Javadocs”





Lekcja 6 i typ wyliczeniowy

Po uruchomieniu kreatora wybrać folder docelowy gdzie będzie zapisana strona z dokumentacją (Destination) i nacisnąć Finish



Slajd 25

JAVA
Dla początkujących

JavaDocs – dokumentacja html

Enum EnumTest.Pory_roku

```

public static enum EnumTest.Pory_roku
extends java.lang.Enum<EnumTest.Pory_roku>
{
    Version:
    1.0
    Author:
    karol.kowalski
}

```

Lista pór roku w naszej szerokości geograficznej

DAILY GROUP Lekcja 6 i typ wyliczeniowy

Oto rezultat generowania

Automatycznie powstaje dokument HTML który przedstawia hierarchię klas, dla każdej klasy opis, listę metod, drzewo relacji i dokumentację którą wpisaliśmy

Slajd 26

JAVA
Dla początkujących

Pytania

1. Co to jest typ wyliczeniowy?
2. Do czego stosujemy typ enum?
3. Jakie zalety w programowaniu ma typ enum?
4. Co to jest JavaDoc?
5. Jak należy dokumentować kod w JavaDoc?
6. Dlaczego dokumentowanie jest potrzebne?

DAILY GROUP Lekcja 6 i typ wyliczeniowy

Pytania podsumowujące temat.

Opis złożonych osiągnięć ucznia

Po tej lekcji uczniowie będą umieli poznawać informacje na temat wykorzystywanych obiektów i ich metod z dokumentacji. W łatwy sposób operować typami wyliczeniowymi.



Lekcja 7 Tablice jednowymiarowe i wielowymiarowe, kolekcje

Cel lekcji

Celem lekcji jest przedstawienie podstawowych możliwości przechowywania zbiorów obiektów w tablicach, lub kolekcjach. Wy tłumaczone będzie pojęcie listy.

Treść - slajdy z opisem



Slajd 1

Slajd 2

Tablicą nazywamy obiekt zawierający zestaw elementu określonego typu. Długość i typ tablicy są określone przy jej deklarowaniu
Struktura taka pozwala na zarządzanie w programie zbiorem elementu tego samego typu.
Np. zestaw ostatnich 10 pomiarów temperatury.



Slajd 3

JAVA
Dla początkujących


Deklaracja tablicy jednowymiarowej

```

int[] tablica;// deklaracja - tablica będzie zawierać typu int
tablica = new int[8];// inicjacja - długość tablicy jest 8

tablica[0] = 100;// pierwszy element tablicy
tablica[1] = 220;
tablica[2] = 250;
tablica[3] = 280;
tablica[4] = 300;
tablica[5] = 310;
tablica[6] = 400;
tablica[7] = 405;// ostatni element tablicy

System.out.println("Element tablicy o indeksie 3="+tablica[3]);
    
```



Lekcja 7 – tablice, kolekcje

int[] tablica;

W pierwszej linii wykonywana jest deklaracja tablicy. Mówi ona że:

- tablica będzie zawierać typ całkowity (int),
- nazwa obiektu tablicy to ... tablica

tablica = new int[8];

Inicjowana jest tablica i ustawiana jest jej długość

Linie zawierające „tablica[1] = 220;” ustawiają kolejne wartości tablicy

Co robi ostatnia linia kodu?

Odp. Wyświetla napis „Element tablicy o indeksie 3=280”

Co zrobić aby wyświetlić pierwszą wartość tablicy?

Odp. Ustawić indeks tablica[0]

Slajd 4




JAVA
Dla początkujących

Tablica jednowymiarowa

Jaki będzie efekt wywołania takiej linii dla kodu z poprzedniego slajdu?

```

System.out.println("Element tablicy o indeksie 8="+tablica[8]);

Exception in thread "main"
java.lang.ArrayIndexOutOfBoundsException: 8
at ArrayTest.main(ArrayTest.java:22)
    
```



Lekcja 7 – tablice, kolekcje




Zgłoszony zostanie błąd


Array Index Out Of Bounds Exception – Wyjątek: Indeks tablicy poza zakresem

Oznacza to, że nie wolno wychodzić poza zakres tablicy bo spowoduje to wyjątek i przerwanie działania programu.

Slajd 5



B2E
BUSINESS TO EDUCATION




SZCZECIŃSKI PARK
NAUKOWO-TECHNOLOGICZNY

JAVA
Dla początkujących

Tablica jednowymiarowa - zadanie

Proszę napisać program który:

- zadeklaruje tablicę 10 elementową typu int
- wypełni ją dowolnymi wartościami
- wyświetli wszystkie wartości w konsoli




Lekcja 7 – tablice, kolekcje

Sugestia: wykorzystać pętlę for

Slajd 6



B2E
BUSINESS TO EDUCATION



SZCZECIŃSKI PARK
NAUKOWO-TECHNOLOGICZNY

JAVA
Dla początkujących


Rozwiązanie zadania

```
int[] tablica;// deklaracja
tablica = new int[10];// inicjacja

for (int i=0;i<10;i++)// iteruj od 0 do 9
tablica[i] = i*100;

for (int i=0;i<10;i++)// iteruj od 0 do 9
System.out.println("Element tablicy o indeksie
"+i+"="+tablica[i]);

    Element tablicy o indeksie 0=0
    Element tablicy o indeksie 1=100
    Element tablicy o indeksie 2=200
    Element tablicy o indeksie 3=300
    Element tablicy o indeksie 4=400
    ...
```



Lekcja 7 – tablice, kolekcje

Zastosowanie pętli for i zmiennej „i” do iterowania po elementach tablicy

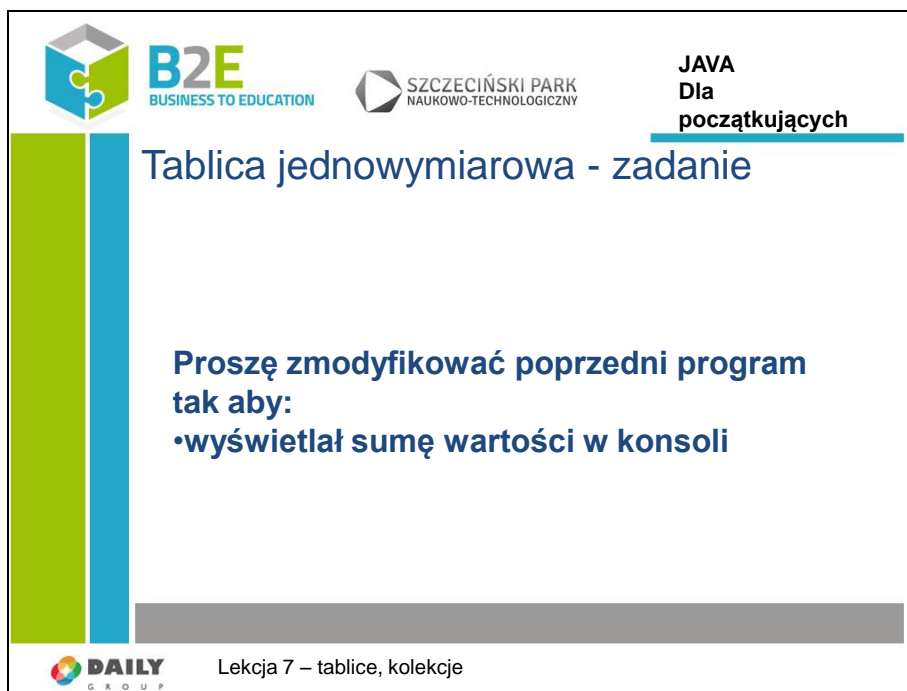
Proszę zwrócić uwagę na:

Zmienna „i” jest wykorzystana do ustalenia wartości tablicy = i*100;

Zmienna „i” jest wykorzystana do wyświetlenia indeksu przy wyświetlaniu wyników



Slajd 7



Slide 7 content: B2E BUSINESS TO EDUCATION logo, SZCZECIŃSKI PARK NAUKOWO-TECHNOLOGICZNY logo, JAVA Dla początkujących title, Tablica jednowymiarowa - zadanie title, Proszę zmodyfikować poprzedni program tak aby: •wyświetlał sumę wartości w konsoli instruction, DAILY GROUP logo, Lekcja 7 – tablice, kolekcje footer.

Sugestia: wykorzystać pętle for dla wszystkich elementów tablicy
for (int x:tablica)

Slajd 8



Slide 8 content: B2E BUSINESS TO EDUCATION logo, SZCZECIŃSKI PARK NAUKOWO-TECHNOLOGICZNY logo, JAVA Dla początkujących title, Rozwiązanie zadania title, Java code snippet: int suma =0; for (int x:tablica) suma+=x; System.out.println("Suma elementów tablicy wynosi "+suma);, Output: Suma elementów tablicy wynosi 4500, DAILY GROUP logo, Lekcja 7 – tablice, kolekcje footer.

Zastosowanie pętli for dla wszystkich elementów tablicy nie wymaga iteratora (zmiennej „i”).

Proszę zwrócić uwagę na formułę suma+=x;

Co ona oznacza?

Odp: suma = suma + x;



Slajd 9



JAVA
Dla
początkujących

Tablica – inny sposób deklaracji

```
int[] tablica = {12,23,45,67,89,0,1,2,3,4};
```

Element tablicy o indeksie 0=12
Element tablicy o indeksie 1=23
Element tablicy o indeksie 2=45
Element tablicy o indeksie 3=67
Element tablicy o indeksie 4=89
Element tablicy o indeksie 5=0
Element tablicy o indeksie 6=1
Element tablicy o indeksie 7=2
Element tablicy o indeksie 8=3
Element tablicy o indeksie 9=4
Suma elementów tablicy wynosi 246



Lekcja 7 – tablice, kolekcje

Inny sposób deklarowania i inicjowania tablicy – przyda się tylko wówczas gdy wartości tablicy będą w programie stałe.


Slajd 10



JAVA
Dla
początkujących

Łańcuch znaków - string

```
String nazwisko = "Kowalski";  
System.out.println("Nazwisko =" + nazwisko);
```



Lekcja 7 – tablice, kolekcje

Łańcuch znaków String jest tablicą jednowymiarową zawierającą elementy typu znak (char).

Proszę sprawdzić jak będzie działał powyższy program.

Odp. Jak się można spodziewać program wyświetli nazwisko „Kowalski”.



Slajd 11

B2E
BUSINESS TO EDUCATION

SZCZECIŃSKI PARK
NAUKOWO-TECHNOLOGICZNY

JAVA
Dla początkujących

Rozmiar tablicy

```
System.out.println("Liczba
elementów tablicy wynosi "+tablica.Length);

System.out.println("Długość nazwiska =" +nazwisko.Length());
```

Lekcja 7 – tablice, kolekcje

Typ String jest specyficznym typem tablicowym i posiada metody do wykonywania operacji na znakach. Metoda length() zwraca długość napisu zapamiętanego w zmiennej. Typ tablicowy posiada licznik .length który zwraca liczbę elementów w tablicy

Slajd 12

B2E
BUSINESS TO EDUCATION

SZCZECIŃSKI PARK
NAUKOWO-TECHNOLOGICZNY

JAVA
Dla początkujących

Tablica dwuwymiarowa

Rozmiar = 8


0	1	2	3	4	5	6	7
0							
1							
2							
3							

Lekcja 7 – tablice, kolekcje

Rozważmy tablicę dwuwymiarową 8x4
Jak będzie wyglądała deklaracja takiej tablicy?



Slajd 13



JAVA
Dla początkujących

Deklaracja tablicy dwuwymiarowej



```
int[][] tablica2;  
tablica2 = new int[8][4];  
  
int[][] tablica3;  
tablica3 = new int[8][4][3];
```



Lekcja 7 – tablice, kolekcje

Jak zadeklarować tablicę trójwymiarową 8x4x3?

Slajd 14




JAVA
Dla początkujących

Tablica dwuwymiarowa - zadanie

Proszę program który:

- Zadeklaruje tablicę dwuwymiarową 8x4
- Ustawi jako wartość tablicy iloczyn indeksów np. dla indeksu 2,2 wartość będzie 4
- Wyświetli wszystkie elementy tablicy dwuwymiarowej
- Obliczy i wyświetli sumę wszystkich elementów tablicy





Lekcja 7 – tablice, kolekcje

Sugestia: wykorzystać 2x pętle for dla wszystkich elementów tablicy w wymiarze x i y (8,4)
Kto poda pierwszy wynik sumy?

Odp: 168



Slajd 15

JAVA
Dla początkujących


Rozwiązanie zadania

```

int[][] tablica2;
tablica2 = new int[8][4];
int suma2 = 0;

for (int i=0;i<8;i++)
    for (int j=0;j<4;j++)
        tablica2[i][j] = i*j;

for (int i=0;i<8;i++)
for (int j=0;j<4;j++)
{
suma2+=tablica2[j][i];
System.out.println("Element tablicy o indeksie "+i+
    ", "+j+"="+tablica2[i][j]);
}
System.out.println("Suma elementów tablicy
    dwuwymiarowej wynosi "+suma2);
    
```



Lekcja 7 – tablice, kolekcje

W rozwiązaniu zadania jest błąd.

Kto wskaże w którym miejscu, na czym błąd polega i co stałoby się przy uruchomieniu tego programu.

Należy bardzo uważać żeby nie pomylić kolejności iteratorów

Slajd 16




JAVA
Dla początkujących

Inne sposoby zarządzania tablicami

Predefiniowane struktury:

- Collection
- LinkedList
- ArrayList
- HashMap



Lekcja 7 – tablice, kolekcje

Zarządzanie tablicami z przykładów jest na dłuższą metę dość uciążliwe.

Wady:

- Musimy pilnować zakresów,
- Musimy deklarować typy obiektów,
- Aby wyszukać dany element musimy napisać parę linii kodu,
- ...

W celu wyeliminowania tych ale również innych wad środowisko programistów



przygotowało sobie specjalne narzędzia. Są to struktury pozwalające na łatwe zarządzanie danymi.

Slajd 17

Zarządzanie tablicami z przykładów jest na dłuższą metę dość uciążliwe.

Wady:

- Musimy pilnować zakresów,
- Musimy deklarować typy obiektów,
- Aby wyszukać dany element musimy napisać parę linii kodu,
- ...

W celu wyeliminowania tych ale również innych wad środowisko programistów przygotowało sobie specjalne narzędzia. Są to struktury pozwalające na łatwe zarządzanie danymi.

Slajd 18

Pierwsza uwaga do przykładu – należy wykorzystać bibliotekę narzędzi java.util. Dla uproszczenia dodamy wszystkie narzędzia w domenie util poprzez instrukcję

```
import java.util.*; //potrzebna biblioteka narzędziowa
...
List l = new ArrayList(); // deklaracja i inicjacja listy

for (int i=0;i<10;i++)
    l.add(new Integer(i)); // dodawanie obiektów do listy

Iterator i = l.iterator(); // powołanie iteratora

while (i.hasNext()) // petla po wszystkich elementach
    System.out.println("Element listy =" + i.next());
// wyświetlenie kolejnych elementów
}
```



import java.util.*; - gwiazdka oznacza że wszystkie biblioteki w danej bibliotece będą dołączone do programu.

Alternatywnie można dodać następujące linie:

import java.util.ArrayList;

import java.util.List;

import java.util.Iterator;

Deklaracja listy o nazwie „l” jest prosta: **List l = new ArrayList();**

Warto zauważyć że nie deklarowaliśmy typu tablicy.

Następnie dodawane są **obiekty** do listy (10 elementów). Aby powołać obiekt stosujemy instrukcję `new Integer(<wartość>)` – nowy obiekt typu Integer.

Aby wyświetlić wszystkie elementy z listy należy powołać iterator. Jest to obiekt pozwalający na dostęp do kolejnych elementów listy.

Zastosowano pętlę while której sprawdza czy jest na iteratorze kolejny element. Jeśli tak instrukcja `i.next()` zwraca obiekt i jednocześnie przechodzi do kolejnego elementu listy.

Slajd 19

The slide features a header with logos for B2E (Business to Education), Szczeciński Park Naukowo-Technologiczny, and the text 'JAVA Dla początkujących'. The main title is 'Lista - zadanie'. The central text asks the user to modify a previous program to manage strings instead of numbers, increase the number of elements from 10 to 20, and display all elements. The footer includes the DAILY GROUP logo and the text 'Lekcja 7 – tablice, kolekcje'.

Sugestia: wykorzystać obiekt String analogicznie jak Integer



Slajd 20



Slide 20 content: B2E BUSINESS TO EDUCATION logo, SZCZECIŃSKI PARK NAUKOWO-TECHNOLOGICZNY logo, JAVA dla początkujących title, Rozwiązanie title, two code snippets showing list operations, and output: Element listy = test 19, Element listy = 21. Footer: DAILY GROUP Lekcja 7 – tablice, kolekcje

Wystarczy w jednym miejscu zmienić instrukcję powołującą nowe obiekty z `new Integer(i)` na np. `new String(„tekst”+i)`

Co warto zauważyć:

1. Przerobienie programu jest dużo prostsze niż w przypadku tablic [] []
1. Jedno miejsce
2. Nie musimy zajmować się kontrolą liczby elementów
3. Nic nie stoi na przeszkodzie aby lista zarządzała różnymi obiektami np. `String` i `Integer`
4. Mamy do dyspozycji całą paletę narzędzi do zarządzania listą – następny slajd

Slajd 21



Slide 21 content: B2E BUSINESS TO EDUCATION logo, SZCZECIŃSKI PARK NAUKOWO-TECHNOLOGICZNY logo, JAVA Dla początkujących title, Lista – dodatkowe funkcje title. Footer: DAILY GROUP Lekcja 7 – tablice, kolekcje


Są wśród nich:

- Dodawanie i wstawianie obiektów (`add`), na pozycję, dodawanie całej kolekcji
- Czyszczenie listy
- Wyszukiwanie obiektów




- Pobieranie obiektów wg nr
- Usuwanie wg indeksów, wg obiektów, wszystkich
- Tworzenie podlist
- Ustawianie obiektu na pozycji
- Sprawdzanie rozmiaru
- Iterowanie

Slajd 22



B2E
BUSINESS TO EDUCATION



SZCZECIŃSKI PARK
NAUKOWO-TECHNOLOGICZNY

JAVA
dla początkujących


Klasy generyczne

```

//Do tej pory:
List imiona = new ArrayList();
for(int i = 0; i < imiona.size(); i++) {
    String imie = (String) imiona.get(i) ;
}

//Generycznie:
List<String> imiona = new ArrayList<String>();
for(int i = 0; i < imiona.size(); i++) {
    String imie = imiona.get(i) ;
}

```



Lekcja 7 – tablice, kolekcje

Klasy generyczne zostały dodane do Javy 1.5. Generyczność pozwala na tworzenie klas posiadających pełną implementację, jednakże bez deklarowania typów wykorzystywanych przez implementację.

Na poprzednich slajdach nie deklarowaliśmy jaki typ danych przetrzymuje kolekcja. Przez co możliwe było dodanie dowolnego typu do tej samej kolekcji. Dodatkowo podczas wyciągania elementów z kolekcji trzeba było rzutować je na odpowiedni typ danych. Jeśli przez pomyłkę umieściliśmy przypadkowy obiekt w kolekcji to o błędzie dowiadywalismy się dopiero podczas działania programu w momencie rzutowania.

Generics – tak określane w języku angielskim nie dotyczą tylko i wyłącznie kolekcji, można je stosować w własnych klasach. Jednak ich wykorzystanie wykracza poza ramy tego programu nauczania. Niemniej jednak kolekcje są bardzo dobrym przykładem ich użycia. Kolekcja przetrzymuje dane, zazwyczaj tego samego typu. Wobec czego logika zarządzania kolekcją (dodawanie, usuwanie, pobieranie elementów) nie zależy od typu jaki kolekcja przechowuje. Dopiero podczas użycia kolekcji programista deklaruje iż zamierza przetrzymywać dany typ w kolekcji. Na naszym przykładzie String. Taka deklaracja pozwala kompilatorowi na zgłaszanie prób wstawienia do kolekcji nie kompatybilnego typu i wczesne zauważenie błędu. Dodatkowo nie musimy już rzutować pobranych elementów, gdyż z góry wiadomo jaki typ danych przetrzymuje kolekcja.

Jest to dość pobieżne omówienie typów generycznych, więc zachęcam do wnauki we własnym zakresie. W dalszej części kursu będziemy czasem używać generics'ów jednak nie zakładamy umiejętności tworzenia klas generycznych przez uczniów.



Slajd 23

B2E
BUSINESS TO EDUCATION

SZCZECIŃSKI PARK
NAUKOWO-TECHNOLOGICZNY

JAVA
Dla początkujących

Pytania

1. Jak zadeklarować listę 1-dno wymiarową?
2. Jak wypełnić listę wartościami – zaproponuj krótki program?
3. Co się stanie przy przekroczeniu zakresu tablicy?
4. Jak sprawdzić długość tablicy?
5. Jak zadeklarować listę trójwymiarową?
6. Co to jest ArrayList i do czego służy?
7. Do czego służy obiekt Iterator?
8. W jakiej domenie znajdują się narzędzia do obsługi list?

DAILY
GROUP

Lekcja 7 – tablice, kolekcje

Pytania podsumowujące temat.

Opis założonych osiągnięć ucznia

Po tej lekcji uczniowie będą umieli gromadzić obiekty w tablicach i kolekcjach. Posiadają również umiejętność przeglądanie tych struktur za pomocą pętli.

Lekcja 8 Operacje wejścia – wyjścia i wyjątki

Cel lekcji

Celem lekcji prezentacja możliwości zabezpieczenia aplikacji przed błędami, oraz umiejętność wymiany danych z zasobami z poza programu (sieć, pliki lokalne).

Treść - slajdy z opisem

Slajd 1

B2E
BUSINESS TO EDUCATION

SZCZECIŃSKI PARK
NAUKOWO-TECHNOLOGICZNY

JAVA
dla początkujących

Java - lekcja 8

Operacje wejścia – wyjścia (IO) i wyjątki

KAPITAŁ LUDZKI
NARODOWA STRATEGIA SPÓJNOŚCI

Człowiek - najlepsza inwestycja

UNIA EUROPEJSKA
EUROPEJSKI
FUNDUSZ SPOŁECZNY

Projekt współfinansowany przez Unię Europejską w ramach Europejskiego Funduszu Społecznego

DAILY
GROUP



Slajd 2

The slide features a header with four logos: a stylized cube logo, B2E BUSINESS TO EDUCATION, SZCZECIŃSKI PARK NAUKOWO-TECHNOLOGICZNY, and JAVA dla początkujących. The main content area contains the text 'IO', 'Co to jest?', and 'Zdarzenia losowe?'. A vertical bar on the left is divided into green and blue sections. A grey horizontal bar is at the bottom. The DAILY GROUP logo is in the bottom left corner.

Operacje wejścia/wyjścia - w informatyce taką nazwę nadano zadaniom związanym z komunikacją ze światem zewnętrznym. Światem zewnętrznym z punktu widzenia pisanej właśnie aplikacji. I tak oczekiwanie na interakcję użytkownika (wpisanie czegoś w konsoli, kliknięcie myszką), odczyt z lokalnego dysku twardego, zapis do sieciowego dysku, pobranie zasobu z Internetu, wydrukowanie czegoś, użycie kamery, odczyt z bazy danych itp. – to wszystko można zaklasyfikować jako komunikację ze światem zewnętrznym.

Ponieważ operacje IO odbywają się pomiędzy różnymi systemami (światami) mogą wystąpić różne problemy komunikacyjne. Zdalny serwer się zrestartuje lub po prostu przerwie połączenie, kabel od drukarki się odłączy (użytkownik akurat przesunął nogę i zahaczył o niego), dysk twardy będzie miał uszkodzoną powierzchnię i nie da się odczytać pliku, lub po prostu się przepełni i nie damy rady nic więcej zapisać. Ogólnie wiele nieprzewidzianych sytuacji może się wydarzyć.

Slajd 3

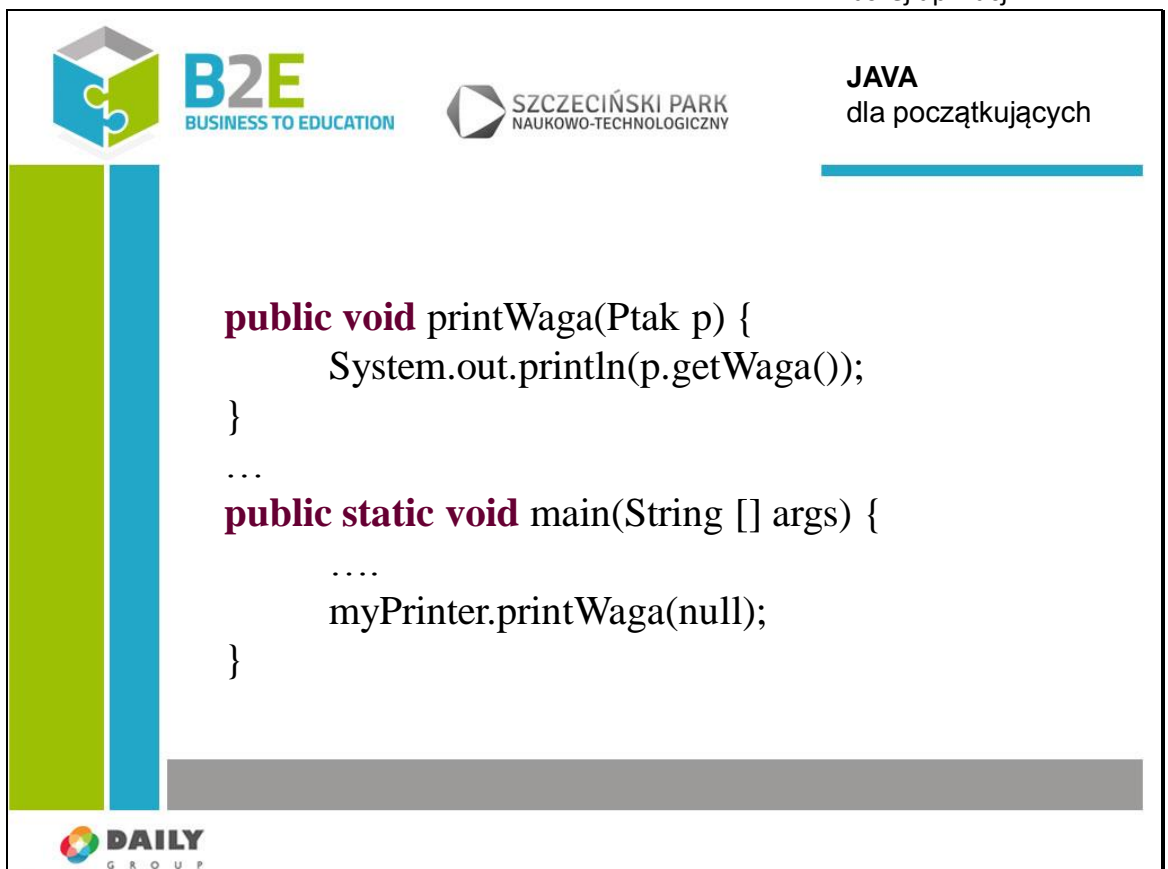


Slide 3 content: The slide features a header with logos for B2E (BUSINESS TO EDUCATION), SZCZECIŃSKI PARK NAUKOWO-TECHNOLOGICZNY, and 'JAVA dla początkujących'. The main title is 'wyjątki' (exceptions). Below the title, it says 'Sytuacje wyjątków!' and 'Potrzebne jest ubezpieczenie (zabezpieczenie)'. The slide is branded with 'DAILY GROUP' at the bottom left.

W związku z czym musi istnieć jakiś mechanizm obsługi takich sytuacji. Z pomocą przychodzą nam wyjątki (exceptions).

Wyjątki:
Podczas obsługi IO w Javie często korzysta się z wyjątków, w celu naprawy stanu programu po nieoczekiwanym zdarzeniu. Jednak wyjątki w Javie nie ograniczają się tylko do obsługi błędów IO. Wyjątki są uniwersalnym mechanizmem obsługi nieprzewidzianych zdarzeń, również pochodzących z wnętrza naszej aplikacji.

Slajd 4



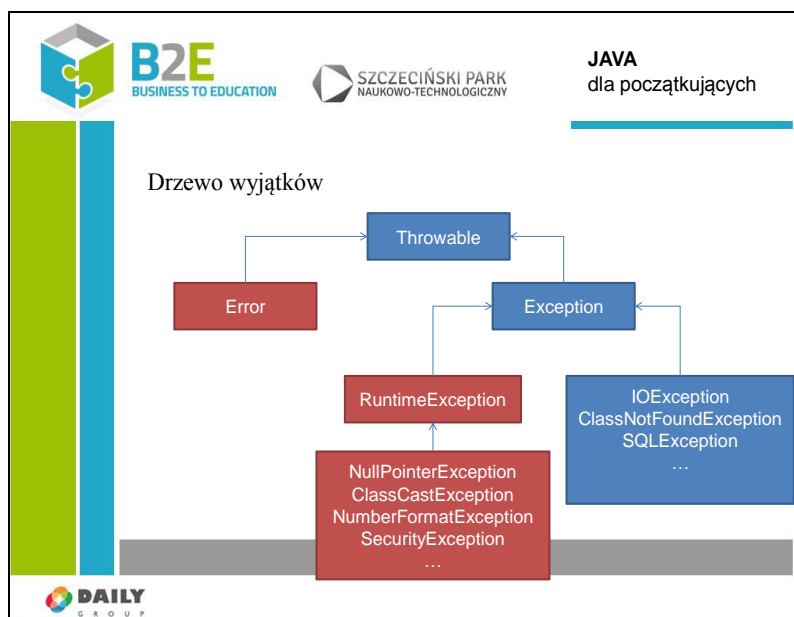
Slide 4 content: The slide features the same header as slide 3. The main content is a code snippet in Java. The code shows a method `printWaga` and a `main` method. In the `main` method, `myPrinter.printWaga(null)` is called, which is the example of a NullPointerException. The slide is branded with 'DAILY GROUP' at the bottom left.

Przykładowo próba odwołania się do metody na referencji której wartość jest równa NULL; JVM chętnie wykona nasze rozkazy, ale jak ma rozumieć to wywołanie? Maszyna wirtualna nie jest w stanie wywołać metody dla nie zdefiniowanego obiektu, dlatego napotykając na taką sytuację wyrzuca wyjątek `NullPointerException`. Programista został poinformowany, że wydarzyła się sytuacja, której nikt się nie spodziewał, z której



nie ma prostego logicznego i domyślnego wyjścia. Co niby ma zrobić program, wypisać losową wagę? Udać, że nic się nie stało? A co, jeśli ktoś czeka, aż na ekranie pojawi się waga ptaka? No właśnie, wyjątki muszą zostać obsłużone w należyty sposób. Brak obsługi wyjątku spowoduje zakończenie programu, aby tego uniknąć to programista musi zdecydować, co w takim przypadku zrobić. Każdy problem powinien mieć jakieś rozsądne rozwiązanie, dla naszego przykładu można przechwycić wyjątek i wypisać na konsolę „Niestety ptak niedostępny, wobec czego nie mogę podać wagi”.

Slajd 5




Wyjątki w Javie tworzą drzewo dziedziczenia. Rozróżniamy dwa typy wyjątków. Checked i Unchecked exceptions. Na slajdzie na czerwono zaznaczone są wyjątki nieweryfikowalne, a na zielono weryfikowalne.

Checked exceptions – ich obsługa jest obowiązkowa, wiążą się z sytuacjami, które ze swojej natury zazwyczaj można jeszcze naprawić i zmuszają programistę do podjęcia takiej próby.


Unchecked Exceptions – ich obsługa nie jest wymuszona, gdyż z swej natury sytuacja jest na tyle krytyczna, że zazwyczaj nie ma łatwego sposobu jej naprawy.

Oczywiście sposób naprawy sytuacji zależy od decyzji programisty. I dlatego każdy z tych wyjątków można przechwycić i spróbować poradzić sobie z sytuacją.

Slajd 6



B2E
BUSINESS TO EDUCATION



SZCZECIŃSKI PARK
NAUKOWO-TECHNOLOGICZNY


JAVA

dla początkujących

```
import java.io.BufferedReader;
import java.io.BufferedInputStream;
import java.io.FileInputStream;
import java.io.InputStreamReader;
import java.util.zip.GZIPInputStream;

...

FileInputStream fis = new FileInputStream("/ksiazka.txt.gz");
BufferedInputStream bis = new BufferedInputStream(fis);
GZIPInputStream gis = new GZIPInputStream(fis);
InputStreamReader isr = new InputStreamReader(gis);
BufferedReader br = new BufferedReader(isr);
br.readLine();
```



Operacje wejścia – wyjścia i wyjątki

Operacje wejścia-wyjścia w Javie polegają na obsłudze strumieni. Maszyna wirtualna udostępnia obiekty, które, np. odczytują znaki z klawiatury.

W architekturze obsługi strumieni w Javie zastosowano wzorec dekoratora. Podstawowym obiektem jest strumień, który to w zależności od potrzeb możemy opakować w bardziej specjalizowane klasy dekorujące. Klasa dekoratora dostarcza dodatkowych bardziej wyspecjalizowanych operacji.

W przykładzie odczytujemy pierwszą linię książki przechowywanej w spakowanym pliku tekstowym. Najpierw otwieramy plik, później aby przyspieszyć odczyt buforujemy go w pamięci, później dekompresujemy, następnie zamieniamy strumień bajtów na strumień znaków, na koniec opakowujemy cały czas ten sam strumień w klasę dającą łatwiejszy dostęp do operacji na ciągach znaków. W tym przypadku zależało nam na metodzie `readLine()`.


Dzięki strumieniowi nie musimy w pamięci przechowywać osobnego obiektu dla każdego z kroków. Przetwarzanie odbywa się płynnie i odczytywane są kolejne bajty na żądanie, w związku z czym możliwe jest przetwarzanie dużych plików.



Slajd 7



B2E
BUSINESS TO EDUCATION




SZCZECIŃSKI PARK
NAUKOWO-TECHNOLOGICZNY

JAVA
dla początkujących

```

import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;

class MojaPierwszaKlasa {
    public static void main(String[] args) {
        InputStreamReader isr = new InputStreamReader(System.in);
        BufferedReader br = new BufferedReader(isr);
        String line = "";
        try {
            line = br.readLine();
            System.out.println(„no exception was thrown”);
        } catch (IOException e) {
            e.printStackTrace();
        } finally {
            br.close();
        }
        System.out.println(„Wczytałem: \” + line + \””);
    }
}
    
```



Operacje wejścia – wyjścia i wyjątki

Przykład podany na slajdzie odczytuje cały wers wpisany do konsoli.

Pojawił się nowy element: bloki "try-catch". Jest to mechanizm obsługi błędów. W przypadku gdybyśmy utracili połączenie z urządzeniem z którego czytamy, program jest w stanie obsłużyć błąd i zareagować w odpowiedni sposób. Uchroni to naszą aplikację od nieprzewidzianego zachowania. Zasada działania bloku try-catch jest intuicyjna. Wszystko co znajduje się pomiędzy try i catch jest podejrzane o możliwość wyrzucenia wyjątku. Jako argument do instrukcji catch podajemy klasę wyjątku jaką chcemy złapać. Istnieje możliwość dodania więcej niż jednej instrukcji catch (przykład będzie na dalszych slajdach), w związku z czym można mieć oddzielną obsługę różnych typów nieprzewidzianych sytuacji. Jeśli w bloku try mamy kilka linii kodu i w pierwszej zostanie rzucony wyjątek to pozostałe linie nie zostaną wykonane i program wznowi działanie od zawartości bloku Catch, który złapał zadany wyjątek. W naszym przykładzie w przypadku wyjątku rzuconego z metody readLine() na ekranie nie pojawi się napis „no exception was thrown”. Jeśli żadne z bloków catch nie przechwyci wyjątku, zostanie on przekazany w górę stosu wywołań. Jeśli nic go nie przechwyci program się zatrzyma. Opcjonalną częścią bloku try-catch jest blok finally – blok ten gwarantuje że kod zawarty w nim zawsze się wykona, niezależnie czy był wyjątek czy nie i czy został obsłużony. Jeśli wystąpi wyjątek wewnątrz bloku finalny, to pozostałe linie kodu w tym bloku nie będą wykonane.



Slajd 8



B2E
BUSINESS TO EDUCATION



SZCZECIŃSKI PARK
NAUKOWO-TECHNOLOGICZNY

JAVA dla początkujących

Ćwiczenie:

Sprawdź co wyświetli się na konsoli po wykonaniu poniższego kodu. Sprawdź na własnych przykładach działanie wyrażeń: „\n”, „\t”, „\\”, „\”.

```
class MojaPierwszaKlasa {  
  
    public static void main(String[] args) {  
        System.out.println("\");  
        System.out.println("\ntekst");  
        System.out.println("\\");  
        System.out.println("\ttekst");  
    }  
}
```




Operacje wejścia – wyjścia i wyjątki

\ " - cudzysłów
\n – znak nowej linii
\\ - znak \
\t – tabulator


Więcej o formatowaniu tekstu można dowiedzieć się w dokumentacji:
<http://docs.oracle.com/javase/8/docs/api/java/util/Formatter.html#syntax>



Slajd 9



B2E
BUSINESS TO EDUCATION




SZCZECIŃSKI PARK
NAUKOWO-TECHNOLOGICZNY

JAVA

dla początkujących

```
class MojaPierwszaKlasa {  
  
    public static void main(String[] args) {  
        String str = "sto dwa";  
  
        try {  
            int i = Integer.parseInt(str);  
        } catch (NumberFormatException e) {  
            e.printStackTrace();  
        }  
        System.out.println("Jestem tutaj!");  
    }  
}
```



DAILY
GROUP

Operacje wejścia – wyjścia i wyjątki


W przypadku próby stworzenia instancji klasy "Integer", korzystając z nieprawidłowego łańcucha znaków, program zakończy się niepowodzeniem.

Możemy tego uniknąć stosując blok "try-catch", tak jak na przykładzie. W bloku "try" wykonują się polecenia potencjalnie "niebezpieczne". Oczywiście łańcuch znaków "sto dwa" jest nieprawidłowy i przejdziemy do bloku "catch". Problem zostanie przechwycony i wykonają się polecenie z tego bloku. Na ekranie zobaczymy cały stack-trace programu. W przyszłości, w dużych aplikacjach, będziemy mogli łatwo odszukać klasę, metodę, a nawet wers, w którym pojawił się błąd, wraz z opisem błędu.


Fakt, że w konsoli pokazał się raport o błędzie, nie wpływa na niestabilność programu. Napis "Jestem tutaj!" również będzie widoczny. Uniknęliśmy niespodziewanego przerwania programu.



Slajd 10



B2E
BUSINESS TO EDUCATION



SZCZECIŃSKI PARK
NAUKOWO-TECHNOLOGICZNY

JAVA
dla początkujących

Ćwiczenie:


Wykorzystując przykład z poprzedniego slajdu wczytaj z konsoli dwie liczby dowolnego typu. Przechwycić ewentualne wyjątki. Program powinien zakończyć swoje działanie dopiero po wczytaniu dwóch liczb.

Wykorzystaj fakt, że każda referencja może wskazywać na pusty obiekt „null”, to znaczy, że możesz zapisać:

```
Integer i = null;
```

Następnie możesz wykonać porównanie, aby sprawdzić, czy nadal jest „pusty”:

```
i == null
```



Operacje wejścia – wyjścia i wyjątki

```
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
```

```
class MojaPierwszaKlasa {
```


```
    public static void main(String[] args) {
        int a = read();
        int b = read();
        System.out.println("Wczytałem: "
            + a + " i " + b + ".");
    }
```


```
    private static int read() {
        InputStreamReader isr = new InputStreamReader(System.in);
        BufferedReader br = new BufferedReader(isr);
        Integer i = null;
        while (i == null) {
            try {
                i = Integer.parseInt(br.readLine());
            } catch (IOException e) {
                System.err.println("Błąd odczytu z konsoli. Prawdopodobnie nie da
                się z tym już nic zrobić, propaguję wyjątek wyżej!");
                throw new RuntimeException(e);
            } catch (NumberFormatException e) {
```



```
ponownie.");  
        }  
    }  
    return i;  
}  
}
```


Slajd 11

**B2E**
BUSINESS TO EDUCATION

SZCZECIŃSKI PARK
NAUKOWO-TECHNOLOGICZNY



JAVA
dla początkujących

```
public class ZeroParameterException extends Exception {  
    public ZeroParameterException(String message) {  
        super(message);  
    }  
}
```

Operacje wejścia – wyjścia i wyjątki

Stworzone przez nas klasy mogą powodować charakterystyczne tylko dla nich błędy. Warto, aby w takim przypadku rzucały wyjątek stworzony specjalnie dla nich. Stworzenie takiego wyjątku jest bardzo proste. Wystarczy dziedziczyć po klasie "Exception" i zaimplementować przynajmniej jeden konstruktor.


Slajd 12

JAVA
dla początkujących

```

class MojaPierwszaKlasa {
    public static void main(String[] args) {
        int a = 2, b = 0;
        try {calculate(a, b);} catch
        (ZeroParameterException e) {
            e.printStackTrace();
        }
    }
    private static int calculate(int a, int b)
        throws ZeroParameterException {
        if (a == 0) {throw new ZeroParameterException(
            "Argument a jest zerem!");}
        if (b == 0) {throw new ZeroParameterException(
            "Argument b jest zerem!");}
        return a + b;
    }
}
    
```





Operacje wejścia – wyjścia i wyjątki

Jeżeli metoda może wyjątek wyrzucić, należy zadeklarować to za listą argumentów, tak jak na slajdzie.

Kiedy znajdzie niepożądana sytuacja (w tym przypadku jeden z argumentów będzie zerem), wyjątek powinien być wyrzucony. Użycie "throw" kończy wywołanie metody. Nie zwróci ona żadnej wartości.


Slajd 13

JAVA
dla początkujących

Ćwiczenie:

Napisz program liczący pierwiastki trójmianu kwadratowego. W przypadku parametru a równego 0 rzuć wyjątkiem. W przypadku ujemnej delty również. Pierwiastek jest metodą statyczną „sqrt” w klasie „Math”.



Operacje wejścia – wyjścia i wyjątki

```

public class AZeroweException extends Exception {
    public AZeroweException() {
        super("A nie może być zerem!");
    }
}

public class DeltaUjemnaException extends Exception {
    
```



```

public DeltaUjemnaException() {
    super("Delta jest ujemna");
}

}

public class Trojmian {

    public static void licz(int a, int b, int c)
        throws AZeroweException, DeltaUjemnaException {
        if (a == 0)
            throw new AZeroweException();
        int delta = b * b - 4 * a * c;


        if (delta < 0)
            throw new DeltaUjemnaException();
        // ...
    }
}

class MojaPierwszaKlasa {


    public static void main(String[] args) {
        try {
            Trojmian.licz(2, 20, -1000);
        } catch (AZeroweException | DeltaUjemnaException e) {
            e.printStackTrace();
        }
    }
}

```

Slajd 14



B2E
BUSINESS TO EDUCATION



SZCZECIŃSKI PARK
NAUKOWO-TECHNOLOGICZNY

JAVA
dla początkujących


```

import java.io.IOException;
import java.io.InputStream;
import java.net.MalformedURLException;
import java.net.URL;
import java.net.URLConnection;

class MojaPierwszaKlasa {
    public static void main(String[] args) {
        try {
            URL url = new URL("http://google.com");
            URLConnection con = url.openConnection();
            InputStream is = con.getInputStream();

            is.close();
        } catch (MalformedURLException e) {
            System.out.println("Niepoprawny URL");
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}

```



DAILY
GROUP

Operacje wejścia – wyjścia i wyjątki

Strumienie są bardzo uniwersalne. Java umożliwia w łatwy sposób obsługę sieci za ich pomocą.

Spójrz na wyjątki i spróbuj odgadnąć jakie problemy mogą wystąpić w trakcie połączenia z Internetem.

Nieprawidłowy adres URL i błąd odczytu (zerwane połączenie).

Odpowiednie nazewnictwo czyni pracę programisty łatwiejszą.

Slajd 15



B2E
BUSINESS TO EDUCATION



SZCZECIŃSKI PARK
NAUKOWO-TECHNOLOGICZNY

JAVA
dla początkujących

Ćwiczenie:

Wykorzystując kod z poprzedniego przykładu połącz się z dowolną stroną internetową i wyświetl w konsoli jej kod źródłowy.

Sprawdź jakimi metodami możesz operować na obiekcie typu „InputStream”. Każdy bajt odczytany ze strumienia możesz rzutować na typ „char” i wyświetlić w konsoli jako znak.



Operacje wejścia – wyjścia i wyjątki

```
import java.io.IOException;
import java.io.InputStream;
import java.net.MalformedURLException;
import java.net.URL;
import java.net.URLConnection;

class MojaPierwszaKlasa {

    public static void main(String[] args) {
        try {
            URL url = new URL("http://google.com");
            URLConnection con = url.openConnection();
            InputStream is = con.getInputStream();

            byte [] b = new byte[512];
            while (is.read(b) > -1) {
                for (int i=0 ; i<512 ; i++)
                    System.out.print((char)b[i]);
            }

            is.close();
        } catch (MalformedURLException e) {
            e.printStackTrace();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```



Slajd 16

```
}  
}  
  
}  
  
}
```

B2E
BUSINESS TO EDUCATION

SZCZECIŃSKI PARK
NAUKOWO-TECHNOLOGICZNY

JAVA
dla początkujących

```
import java.io.BufferedReader;  
import java.io.DataInputStream;  
import java.io.FileInputStream;  
import java.io.InputStreamReader;  
  
class MojaPierwszaKlasa {  
    public static void main(String[] args) {  
        try(BufferedReader br = new BufferedReader(  
            new InputStreamReader(  
                new DataInputStream(new FileInputStream("plik.txt"))  
            ))) {  
            String strLine;  
            while ((strLine = br.readLine()) != null) {  
                System.out.println(strLine);  
            }  
        } catch (Exception e) {  
            e.printStackTrace();  
        }  
    }  
}
```

DAILY
GROUP

Operacje wejścia – wyjścia i wyjątki

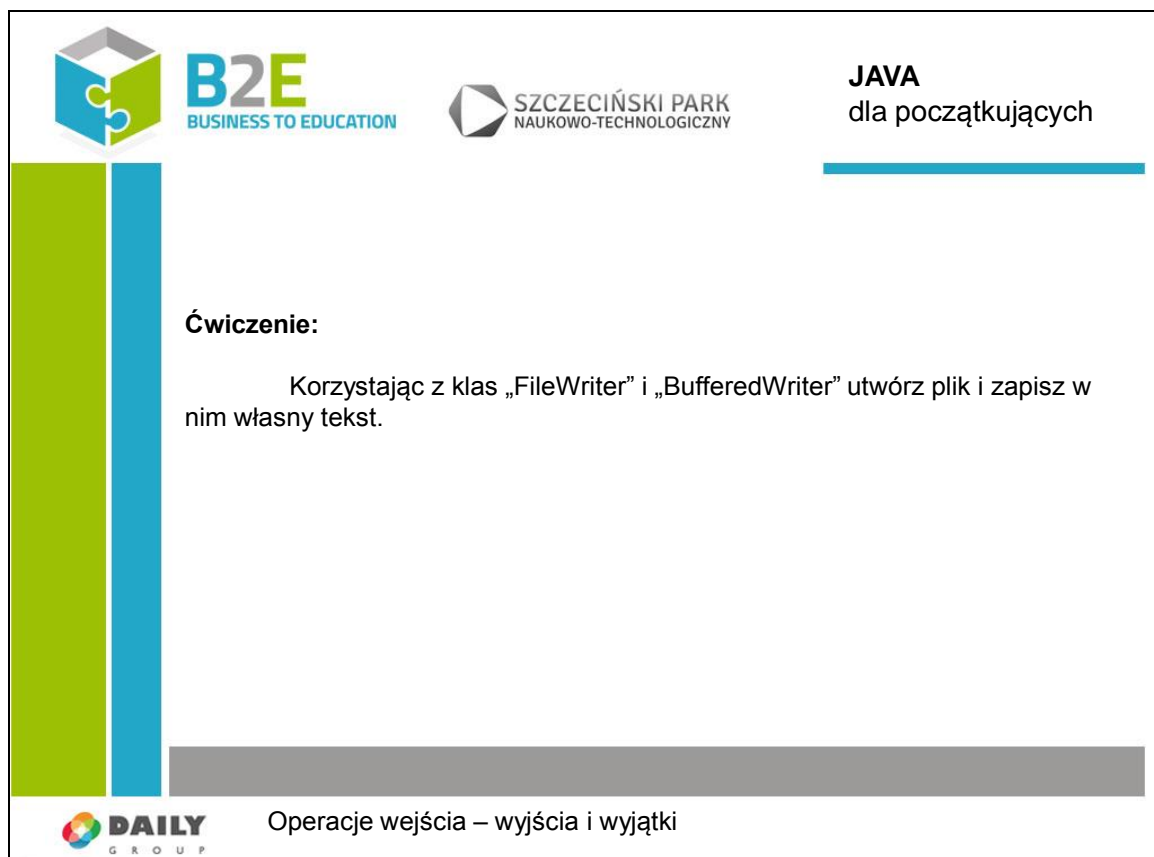
Podstawową operacją na strumieniach jest obsługa plików. Przykład pokazuje, jak odczytać plik.

Domyślna lokalizacja pliku jest w folderze naszego projektu.

Dodatkowo proszę zwrócić uwagę na nową konstrukcję bloku try. Od Javy 1.7 mamy możliwość skorzystania z automatycznego zamykania zasobów. Konstrukcja jest dość prosta w nawiasach półokrągłych następujących zaraz po try inicjalizujemy zasoby, które mają być automatycznie obsługiwane przez maszynę wirtualną Javy. Warunkiem jest to aby zasoby otwierane implementowały interfejs AutoClosesable. Po zakończeniu wykonywania kodu z bloku try wszystkie zasoby zostaną zwolnione.



Slajd 17



The slide features a header with logos for B2E (Business to Education), Szczeciński Park Naukowo-Technologiczny, and the title 'JAVA dla początkujących'. Below the logos is a vertical bar with green and blue segments. The main text is an exercise instruction: 'Ćwiczenie: Korzystając z klas „FileWriter” i „BufferedWriter” utwórz plik i zapisz w nim własny tekst.' At the bottom, there is a logo for 'DAILY GROUP' and the text 'Operacje wejścia – wyjścia i wyjątki'.

```
import java.io.BufferedWriter;  
import java.io.FileWriter;
```

```
class MojaPierwszaKlasa {  
    public static void main(String[] args) {  
        try {  
            FileWriter fstream = new FileWriter("out.txt");  
            BufferedWriter out = new BufferedWriter(fstream);  
            out.write("Mój tekst");  
            out.close();  
        } catch (Exception e) {  
            e.printStackTrace();  
        }  
    }  
}
```



Slajd 18

B2E
BUSINESS TO EDUCATION

SZCZECIŃSKI PARK
NAUKOWO-TECHNOLOGICZNY

JAVA
dla początkujących

Ćwiczenie:

Zapisz do pliku źródło dowolnej strony internetowej. Niech plik ma rozszerzenie „*.html”. Co mogło spowodować taki efekt?

Człowiek - najlepsza inwestycja

KAPITAŁ LUDZKI
NARODOWA STRATEGIA SPÓJNOŚCI

UNIA EUROPEJSKA
EUROPEJSKI
FUNDUSZ SPOŁECZNY

Projekt współfinansowany przez Unię Europejską w ramach Europejskiego Funduszu Społecznego

DAILY
GROUP

Operacje wejścia – wyjścia i wyjątki

```
import java.io.BufferedWriter;
import java.io.FileWriter;
import java.io.IOException;
import java.io.InputStream;
import java.net.MalformedURLException;
import java.net.URL;
import java.net.URLConnection;

class MojaPierwszaKlasa {
    public static void main(String[] args) {
        try {
            URL url = new URL("http://google.com");
            URLConnection con = url.openConnection();
            InputStream is = con.getInputStream();

            FileWriter fstream = new FileWriter("strona.html");
            BufferedWriter out = new BufferedWriter(fstream);
            out.write("strona.txt");
            // byte zajmuje 1 bajt, a char 2 bajty (można błędnie zapisać znaki)
            byte [] b = new byte[512];
            while (is.read(b) > -1) {
                for (int i=0 ; i<512 ; i++)
                    out.write((char)b[i]);
            }
        }
    }
}
```



```
        out.close();
        is.close();
    } catch (MalformedURLException e) {
        e.printStackTrace();
    } catch (IOException e) {
        e.printStackTrace();
    }
}
}
```

Opis założonych osiągnięć ucznia

Po tej lekcji uczniowie będą umieli skorzystać z połączenia sieciowego i plików lokalnych w komputerze. Ponadto będą umieli zabezpieczyć swój program przed potencjalnymi błędami.

Lekcja 9 Graficzny interfejs użytkownika - Swing

1. Cel lekcji

Celem lekcji jest pokazanie możliwości graficznego pakietu Swing. Wytlumaczenie jak poprawnie korzystać z tych elementów, oraz jak zarządzać ich rozkładem w oknie.

2. Treść - slajdy z opisem


Slajd 1




Slajd 1 zawiera tytułową stronę prezentacji. W lewym górnym rogu znajdują się loga: B2E BUSINESS TO EDUCATION, SZCZECIŃSKI PARK NAUKOWO-TECHNOLOGICZNY oraz KAPITAŁ LUDZKI. W prawym górnym rogu napisano: JAVA dla początkujących. W centrum znajduje się tytuł: Java - lekcja 9 oraz podtytuł: Graficzny interfejs użytkownika - Swing. W dolnej części slajdu znajdują się loga: KAPITAŁ LUDZKI, UNIA EUROPEJSKA EUROPEJSKI FUNDUSZ SPOŁECZNY oraz DAILY GROUP. W dolnej części slajdu znajduje się również napis: Projekt współfinansowany przez Unię Europejską w ramach Europejskiego Funduszu Społecznego.



Slajd 2



B2E
BUSINESS TO EDUCATION



SZCZECIŃSKI PARK
NAUKOWO-TECHNOLOGICZNY

JAVA
dla początkujących

```

import javax.swing.JFrame;

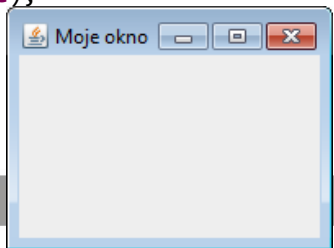
public class Start {


    public static void main(String [] args) {
        JFrame mainFrame = new JFrame("Moje okno");
        mainFrame.setBounds(200, 200, 500, 300);

        mainFrame.setDefaultCloseOperation(JFrame.EXIT_ON_C
OSE);

        mainFrame.setVisible(true);
    }
}

```





Graficzny interfejs użytkownika - Swing

Okno możemy stworzyć korzystając z klasy "Jframe".

Łącuch znaków podany w konstruktorze jest tytułem okna.



Metoda "setBounds" posiada 4 argumenty:

- odległość w poziomie od lewego, górnego rogu,
- odległość w pionie od lewego, górnego rogu,
- szerokość okna,
- wysokość okna.

Jeżeli nie ustawimy metody "setDefaultCloseOperation", to kliknięcie przycisku zamknięcia w prawym, górnym rogu nie spowoduje wyłączenia programu. Okno zniknie, ale aplikacja nadal będzie chodziła w tle!

Ostatnie polecenie powoduje wyświetlenie się okna.

Slajd 3

JAVA
dla początkujących

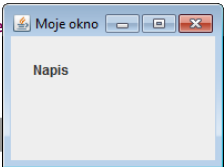
```


public class Start {
    public static void main(String [] args) {
        JFrame mainWindow = new JFrame("Moje okno");
        mainWindow.setBounds(200, 200, 200, 150);

        mainWindow.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        mainWindow.setLayout(null);

        JLabel label = new JLabel("Napis");
        label.setBounds(20, 20, 50, 20);

        mainWindow.add(label);
        mainWindow.setVisible(true);
    }
}
    
```





Graficzny interfejs użytkownika - Swing

Do dyspozycji mamy wiele gotowych komponentów. Jeżeli chcemy wyświetlić tekst możemy skorzystać z klasy "JLabel".

Skorzystajmy z metody "setLayout" z argumentem "null". Będziemy wtedy mogli skorzystać z absolutnego pozycjonowania komponentów widoku. W późniejszych slajdach zostanie wytłumaczone, jak układać elementy korzystając z zarządców rozkładu (layoutów).

Nie zapomnijmy dodać obiektu "label" do okna.

Slajd 4




JAVA
dla początkujących

```

public class Start {
    public static void main(String [] args) {
        JFrame mainWindow = new JFrame("Moje okno");
        mainWindow.setBounds(200, 200, 200, 150);

        mainWindow.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        mainWindow.setLayout(null);

        JTextField jTextField = new JTextField("Napis");
        jTextField.setBounds(20, 20, 100, 40);

        mainWindow.add(jTextField);
        mainWindow.setVisible(true);
    }
}
    
```







Graficzny interfejs użytkownika - Swing

Klasa "JtextField" umożliwi wyświetlenie pola tekstowego z możliwością edycji. Ma ono jednak ograniczenie. Tekst może mieć tylko jeden wers.



Slajd 5

JAVA
dla początkujących

```

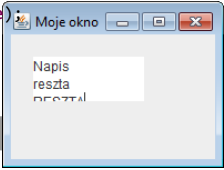
public class Start {


    public static void main(String [] args) {
        JFrame mainWindow = new JFrame("Moje okno");
        mainWindow.setBounds(200, 200, 200, 150);

        mainWindow.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        mainWindow.setLayout(null);

        JTextArea jTextArea = new JTextArea("Napis");
        jTextArea.setBounds(20, 20, 100, 40);

        mainWindow.add(jTextArea);
        mainWindow.setVisible(true);
    }
}
    
```







Graficzny interfejs użytkownika - Swing

"JTextArea" umożliwia wyświetlenie tekstu składającego się z wielu wersów. Zauważmy jednak, że tekst nie mieści się w polu.

Slajd 6

JAVA
dla początkujących

```

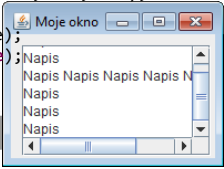
public class Start {


    public static void main(String [] args) {
        JFrame mainWindow = new JFrame("Moje okno");
        mainWindow.setBounds(200, 200, 200, 150);

        mainWindow.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        mainWindow.setLayout(null);

        JTextArea jTextArea = new JTextArea("Napis");
        JScrollPane jScrollPane = new JScrollPane(jTextArea);
        jScrollPane.setBounds(10, 10, 170, 100);

        mainWindow.add(jScrollPane);
        mainWindow.setVisible(true);
    }
}
    
```







Graficzny interfejs użytkownika - Swing

Problem z widocznością tekstu możemy rozwiązać za pomocą "JScrollPane". Obiekt "JTextArea" może teraz zmieniać swój rozmiar dynamicznie. Przeglądanie ułatwiają paski przewijania.

Slajd 7

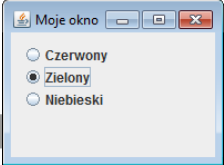




JAVA
dla początkujących

```

JRadioButton button1 = new JRadioButton("Czerwony");
button1.setBounds(10, 10, 100, 20);
JRadioButton button2 = new JRadioButton("Zielony");
button2.setBounds(10, 30, 100, 20);
JRadioButton button3 = new JRadioButton("Niebieski");
button3.setBounds(10, 50, 100, 20);
ButtonGroup colorButtonGroup = new ButtonGroup();
colorButtonGroup.add(button1);
colorButtonGroup.add(button2);
colorButtonGroup.add(button3);

mainFrame.add(button1);
mainFrame.add(button2);
mainFrame.add(button3);
    
```





Graficzny interfejs użytkownika - Swing

Obiekty "JRadioButton" umożliwiają wybór pojedynczej opcji. Zaznaczony może być tylko jeden element dlatego wszystkie muszą tworzyć grupę. Obiekt "ButtonGroup" kontroluje wszystkie komponenty "JRadioButton". Jeżeli nie użylibyśmy go, opisywana zasada nie zachodziłaby.

Slajd 8




JAVA
dla początkujących

```

String [] dane = {"Czerwony", "Zielony", "Niebieski"};
SpinnerModel model = new SpinnerListModel(dane);
JSpinner spinner = new JSpinner(model);
spinner.setBounds(10, 10, 100, 20);

mainFrame.add(spinner);
    
```






Graficzny interfejs użytkownika - Swing

Obiekt typu "JSpinner" również umożliwia nam wybór tylko jednej opcji, ale w nieco inny sposób. Zwróćmy uwagę na sposób, w jaki wstawiamy dane do obiektu. Korzystamy z modelu. Wynika to z zastosowania Wzorca projektowego MVC przez twórców Swingu. Wzorzec ten rozdziela warstwę widoku od warstwy danych.



Slajd 9

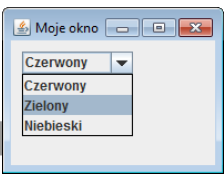




JAVA
dla początkujących

```

JComboBox<String> jComboBox = new JComboBox<String>();
jComboBox.addItem("Czerwony");
jComboBox.addItem("Zielony");
jComboBox.addItem("Niebieski");
jComboBox.setBounds(10, 10, 100, 20);

mainFrame.add(jComboBox);
    
```







Graficzny interfejs użytkownika - Swing

"JComboBox", też pozwala wybrać tylko jedną opcję. Jest on typem generycznym. W nawiasach "<>" podajemy typ obiektu jaki będzie przechowywany w "JComboBox".

Slajd 10

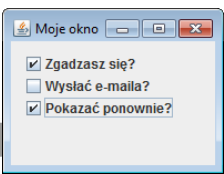




JAVA
dla początkujących

```

JCheckBox jCheckBox1 = new JCheckBox("Zgadzasz się?");
jCheckBox1.setBounds(10, 10, 150, 20);
JCheckBox jCheckBox2 = new JCheckBox("Wysłać e-maila?");
jCheckBox2.setBounds(10, 30, 150, 20);
JCheckBox jCheckBox3 = new JCheckBox("Pokazać ponownie?");
jCheckBox3.setBounds(10, 50, 150, 20);

mainFrame.add(jCheckBox1);
mainFrame.add(jCheckBox2);
mainFrame.add(jCheckBox3);
    
```







Graficzny interfejs użytkownika - Swing

"JCheckBox" umożliwia wybór kilku opcji. Elementy te są całkowicie niezależne. Nie dodajemy ich do żadnej grupy, tak jak to było w przypadku "JRadioButton".

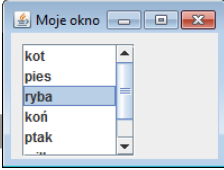



Slajd 11

JAVA
dla początkujących

```
String[] selections = { "kot", "pies", "ryba", "koń", "ptak",
"wilk", "lis"};
JList<String> list = new JList<String>(selections);
JScrollPane jScrollPane = new JScrollPane(list);
jScrollPane.setBounds(10, 10, 100, 100);
mainFrame.add(jScrollPane);
```







Graficzny interfejs użytkownika - Swing

"JList" jest również typem generycznym. Pozwala on wyświetlić dane w bardziej czytelny sposób. Można ustawiać możliwość zaznaczania pojedynczego lub wielokrotnego.

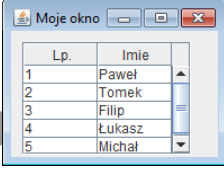
Slajd 12





JAVA
dla początkujących

```
String rowData[][] = { { "1", "Paweł"},
{ "2", "Tomek"},
{ "3", "Filip"},
{ "4", "Łukasz"},
{ "5", "Michał" } };
String columnNames[] = { "Lp.", "Imie"};
JTable table = new JTable(rowData, columnNames);
JScrollPane jScrollPane = new JScrollPane(table);
jScrollPane.setBounds(10, 10, 150, 100);

mainFrame.add(jScrollPane);
```






Graficzny interfejs użytkownika - Swing

"Jtable" działa podobnie jak "Jlist", z tą różnicą, że umożliwia wyświetlenie dodatkowych kolumn.



Slajd 13

  **JAVA**
dla początkujących

Ćwiczenie:

Zbuduj formularz do wpisania adresu zamieszkania.

Imię:

Nazwisko:


Ulica:

Numer domu:



Numer Lokalu:

Kod pocztowy: -

Miasto:

 Graficzny interfejs użytkownika - Swing

Slajd 14


  **JAVA**
dla początkujących

Ćwiczenie:

Stwórz plik o zawartości:


Jan
Kowalski
Wiejska
12
4
12-345
Warszawa

Plik może znajdować się w dowolnej lokalizacji.


 Graficzny interfejs użytkownika - Swing

Zapisując dane aplikacji w plikach można stworzyć coś w rodzaju prymitywnej bazy danych. Jest to dobre rozwiązanie w przypadku małych i prostych aplikacji.

Slajd 15



B2E
BUSINESS TO EDUCATION



SZCZECIŃSKI PARK
NAUKOWO-TECHNOLOGICZNY

JAVA


dla początkujących

Ćwiczenie:

Odczytaj dane z pliku i wprowadź do stworzonego formularza.

Aby ułatwić sobie odszukanie pliku skorzystaj z gotowego okna dialogowego z pakietu Swing.

```
JFileChooser chooser = new JFileChooser();
chooser.showOpenDialog(null);
File curFile = chooser.getSelectedFile();
```



DAILY
GROUP

Graficzny interfejs użytkownika - Swing

```
import java.io.BufferedReader;
import java.io.DataInputStream;
import java.io.File;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.IOException;
import java.io.InputStreamReader;

import javax.swing.JFileChooser;
import javax.swing.JFrame;
import javax.swing.JTextField;
```

```
public class Start {
```

```
    public static void main(String [] args) {
        JFrame mainFrame = new JFrame("Moje okno");
```



```
mainFrame.setBounds(200, 200, 200, 150);
mainFrame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
mainFrame.setLayout(null);

JFileChooser chooser = new JFileChooser();
chooser.showOpenDialog(null);
File curFile = chooser.getSelectedFile();



JTextField jTextField = new JTextField();
jTextField.setBounds(10, 10, 150, 20);

try {
    FileInputStream fis = new FileInputStream(curFile);
    DataInputStream in = new DataInputStream(fis);
    BufferedReader br = new BufferedReader(new InputStreamReader(in));
    jTextField.setText(br.readLine());
    fis.close();
} catch (FileNotFoundException e) {
    e.printStackTrace();
} catch (IOException e) {
    e.printStackTrace();
}

mainFrame.add(jTextField);
mainFrame.setVisible(true);
}
}
```




Slajd 16



JAVA
dla początkujących


Ćwiczenie:

Spróbuj zmienić obraz okna z formularzem. Czy komponenty zawsze są widoczne?

Wywołaj metodę:



```
mainFrame.setResizable(false);
```

Sprawdź ponownie zachowanie okna.




Graficzny interfejs użytkownika - Swing

Slajd 17



JAVA
dla początkujących

```
public class Start {  
    public static void main(String [] args) {  
        JFrame mainFrame = new JFrame("Moje okno");  
        mainFrame.setBounds(200, 200, 200, 150);  
  
        mainFrame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);  
        mainFrame.setLayout(new FlowLayout());  
  
        JLabel label1 = new JLabel("Mój napis 1");  
        mainFrame.add(label1);  
        JLabel label2 = new JLabel("Mój napis 2");  
        mainFrame.add(label2);  
        JLabel label3 = new JLabel("Mój napis 3");  
        mainFrame.add(label3);  
  
        mainFrame.setVisible(true);  
    }  
}
```



Graficzny interfejs użytkownika - Swing

We wcześniejszych przykładach do metody "setLayout" wstawialiśmy obiekt "null". Teraz skorzystamy z zarządcy rozkładu o nazwie "FlowLayout". Nie musimy już układać każdego elementu osobno. Co więcej, nawet gdy użyjemy metody "setBounds", nie uzyskamy oczekiwanego efektu. "FlowLayout" decyduje o położeniu komponentów.



Slajd 18

JAVA
dla początkujących

Moje okno
Mój napis 1 Mój napis 2
Mój napis 3

Moje okno
Mój napis 1 Mój napis 2 Mój napis 3

Mój napis 1
Mój napis 2
Mój napis 3

DAILY GROUP Graficzny interfejs użytkownika - Swing

Możemy uruchomić poprzedni przykład i zobaczyć, jak "FlowLayout" zarządza widokiem.

Slajd 19

JAVA
dla początkujących

```

JLabel label1 = new JLabel("Mój napis 1");
mainFrame.add(label1);
JLabel label2 = new JLabel("Mój napis 2");
mainFrame.add(label2);
JLabel label3 = new JLabel("Mój napis 3");
mainFrame.add(label3);

JPanel jPanel1 = new JPanel();
jPanel1.setBackground(Color.YELLOW);
JLabel label4 = new JLabel("Mój napis 4");
jPanel1.add(label4);
JLabel label5 = new JLabel("Mój napis 5");
jPanel1.add(label5);

mainFrame.add(jPanel1);
mainFrame.setVisible(true);
    
```

DAILY GROUP Graficzny interfejs użytkownika - Swing

Korzystając z klasy "JPanel" możemy grupować elementy. "Mój napis 4" i "Mój napis 5" będą znajdować się wewnątrz panelu.



Slajd 20

B2E BUSINESS TO EDUCATION

SZCZECIŃSKI PARK
NAUKOWO-TECHNOLOGICZNY

JAVA
dla początkujących

Moje okno

Mój napis 1 Mój napis 2
Mój napis 3
Mój napis 4 Mój napis 5

Moje okno

Mój napis 1 Mój napis 2
Mój napis 3
Mój napis 4 Mój napis 5

DAILY GROUP

Graficzny interfejs użytkownika - Swing

Możemy zaobserwować, że przemieszczany jest cały panel. Dzieje się tak, ponieważ "JPanel" ma własnego zarządcę rozkładu i ustawiony "FlowLayout" nie zarządza komponentami z żółtego pola.

Slajd 21

B2E BUSINESS TO EDUCATION

SZCZECIŃSKI PARK
NAUKOWO-TECHNOLOGICZNY

JAVA
dla początkujących

```
public class Start {
    public static void main(String [] args) {
        JFrame mainFrame = new JFrame("Moje okno");
        mainFrame.setBounds(200, 200, 200, 150);

        mainFrame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        mainFrame.setLayout(new BorderLayout());

        JLabel label1 = new JLabel("Mój napis 1");
        mainFrame.add(label1, BorderLayout.NORTH);
        JLabel label2 = new JLabel("Mój napis 2");
        mainFrame.add(label2, BorderLayout.WEST);
        JLabel label3 = new JLabel("Mój napis 3");
        mainFrame.add(label3, BorderLayout.EAST);
        JLabel label4 = new JLabel("Mój napis 4");
        mainFrame.add(label4, BorderLayout.SOUTH);
        JLabel label5 = new JLabel("Mój napis 5");
        mainFrame.add(label5, BorderLayout.CENTER);

        mainFrame.setVisible(true);
    }
}
```

DAILY GROUP

Graficzny interfejs użytkownika - Swing

Kolejnym przykładem jest "BorderLayout". Jeżeli nie wywołamy metody "setLayout", jest on domyślnym zarządcą. Dodając każdy element podajemy dodatkowo jeden z parametrów: "NORTH", "SOUTH", "WEST", "EAST" i "CENTER".



Slajd 22

B2E BUSINESS TO EDUCATION

SZCZECIŃSKI PARK NAUKOWO-TECHNOLOGICZNY

JAVA dla początkujących

DAILY GROUP Graficzny interfejs użytkownika - Swing

Zmieniając rozmiar okna możemy zaobserwować gdzie poszczególne obszary się znajdują.

Slajd 23

B2E BUSINESS TO EDUCATION

SZCZECIŃSKI PARK NAUKOWO-TECHNOLOGICZNY

JAVA dla początkujących

```
public class Start {
    public static void main(String [] args) {
        JFrame mainFrame = new JFrame("Moje okno");
        mainFrame.setBounds(200, 200, 200, 150);

        mainFrame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        mainFrame.setLayout(new GridLayout(3, 2));

        mainFrame.add(new JLabel("Napis 1"));
        mainFrame.add(new JLabel("Napis 2"));
        mainFrame.add(new JLabel("Napis 3"));
        mainFrame.add(new JLabel("Napis 4"));
        mainFrame.add(new JLabel("Napis 5"));
        mainFrame.add(new JLabel("Napis 6"));
        mainFrame.add(new JLabel("Napis 7"));
        mainFrame.add(new JLabel("Napis 8"));



        mainFrame.setVisible(true);
    }
}
```

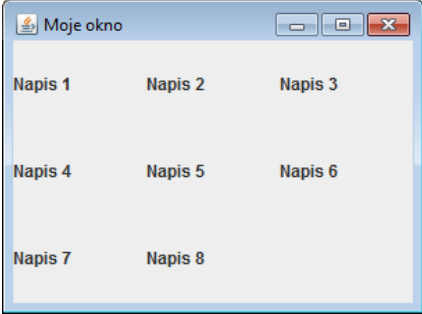
DAILY GROUP Graficzny interfejs użytkownika - Swing


"GridLayout" układa komponenty podobnie jak tabela. Tworząc go należy podać liczbę kolumn i wiersów.



Slajd 24

  **JAVA**
dla początkujących



 Graficzny interfejs użytkownika - Swing

Slajd 25

  **JAVA**
dla początkujących

Ćwiczenie:

Popraw formularz tak, aby dostosowywał się do rozmiaru okna. Możesz określić minimalny rozmiar korzystając z metody:


```
mainFrame.setMinimumSize(new Dimension(200, 150));
```

W ten sposób będziemy mieli pewność, że użytkownik nie zmniejszy okna zbyt mocno i zawsze wszystkie elementy będą widoczne.


 Graficzny interfejs użytkownika - Swing



Slajd 26



B2E
BUSINESS TO EDUCATION



SZCZECIŃSKI PARK
NAUKOWO-TECHNOLOGICZNY


JAVA
dla początkujących

Co siedzi w środku?

Interfejs graficzny użytkownika jest renderowany przy użyciu tylko jednego wątku.

EDT – event dispatching thread – jest odpowiedzialny za rysowanie i uaktualnianie widoku, jest to jedyny wątek który powinien zajmować się renderowaniem.

SwingUtilities.invokeLater(Runnable)
SwingUtilities.invokeAndWait(Runnable)
SwingWorker




Graficzny interfejs użytkownika - Swing


GUI w Swing'u jest renderowane i uaktualniane tylko przez 1 wątek – EDT. Model ten ułatwia architekturę framework'u, ale za razem wymaga ostrożnego obchodzenia się z nim. Przypuśćmy iż po naciśnięciu klawisza chcielibyśmy rozpocząć pobieranie dużego pliku z Internetu. Wątek EDT rozpoczyna wykonywać kod obsługi zdarzenia kliknięcia przycisku, rozpoczyna pobieranie i czeka, i czeka... W międzyczasie GUI się „zawiesza”, nie ma żadnego wolnego wątku który by mógł obsłużyć zdarzenia takie jak ruch myszą, kliknięcie innego przycisku czy wpisywanie tekstu. Dlatego dobrym pomysłem jest uruchomienie czasochłonnych zadań w tle, w osobnym wątku. Oczywiście możemy stworzyć nowy wątek i przekazać mu czasochłonne zadanie, wtedy GUI będzie nadal obsługiwane płynnie przez EDT. Jednak nadejdzie czas kiedy zadanie się skończy i wypadłoby zasygnalizować to jakoś użytkownikowi. Np. poprzez dodanie informacji do listy że plik został już pobrany. I tu pojawia się problem, gdyż nie powinniśmy z naszego dodatkowego wątku ingerować w komponenty GUI. Z pomocą przychodzą nam metody z klasy SwingUtilities. Przekazujemy tam obiekt implementujący Runnable, którego metoda run() zostanie wywołana już w ramach EDT. Uwaga ciało metody run() powinno się szybko wykonywać aby nie zamrozić GUI. W ten sposób możemy uaktualniać bezpiecznie widok z dowolnego wątku.

Drugim rozwiązaniem jest użycie klasy SwingWorker, gdzie tworzenie dodatkowego wątku do wykonania czasochłonnej operacji, informowanie o postępie procentowym tego zadania oraz zakończenie działania już w wątku EDT zostały znacznie ułatwione.

Slajd 27



B2E
BUSINESS TO EDUCATION




SZCZECIŃSKI PARK
NAUKOWO-TECHNOLOGICZNY

JAVA
dla początkujących


```

public class DiagonalLayout implements LayoutManager {
    @Override
    public void addLayoutComponent(String name, Component comp) {}
    @Override
    public void layoutContainer(Container parent) {
        Component[] tab = parent.getComponents();
        Dimension dim = parent.getSize();
        int h = dim.height / (tab.length + 6);
        int w = dim.width / (tab.length + 2);
        for (int i=0 ; i<tab.length ; i++) {
            tab[i].setBounds(tab[i].getWidth(), tab[i].getHeight(),
                w * (i+1), h * (i+1));
        }
    }
    @Override
    public Dimension minimumLayoutSize(Container parent) {
        return null;
    }
    @Override
    public Dimension preferredLayoutSize(Container parent) {
        return null;
    }
    @Override
    public void removeLayoutComponent(Component comp) {}
}
    
```



DAILY
GROUP

Graficzny interfejs użytkownika - Swing

Swing umożliwia nam napisanie własnego zarządcy rozkładu. Wystarczy zaimplementować "LayoutManager". Zaimplementujemy przynajmniej metodę "layoutContainer". Odpowiada ona za ponowne rozmieszczenie komponentów po zmianie rozmiaru okna.

Powyższy przykład układa elementy wzdłuż przekątnej okna.



Slajd 28



B2E
BUSINESS TO EDUCATION



SZCZECIŃSKI PARK
NAUKOWO-TECHNOLOGICZNY

JAVA
dla początkujących

```
public class Start {  
    public static void main(String [] args) {  
        JFrame mainFrame = new JFrame("Moje okno");  
        mainFrame.setBounds(200, 200, 200, 150);  
  
        mainFrame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);  
        mainFrame.setLayout(new DiagonalLayout());  
  
        mainFrame.add(new JLabel("Napis 1"));  
        mainFrame.add(new JLabel("Napis 2"));  
        mainFrame.add(new JLabel("Napis 3"));  
        mainFrame.add(new JLabel("Napis 4"));  
        mainFrame.add(new JLabel("Napis 5"));  
        mainFrame.add(new JLabel("Napis 6"));  
        mainFrame.add(new JLabel("Napis 7"));  
        mainFrame.add(new JLabel("Napis 8"));  
  
        mainFrame.setMinimumSize(new Dimension(200, 150));  
        mainFrame.setVisible(true);  
    }  
}
```



Graficzny interfejs użytkownika - Swing

Slajd 29

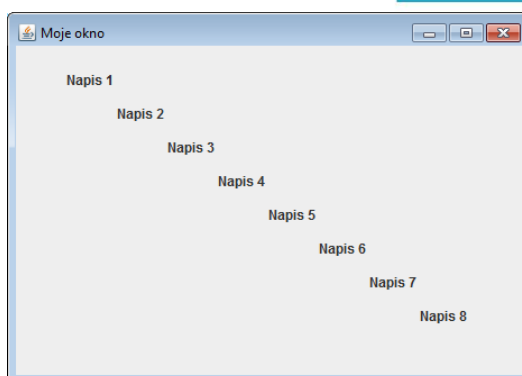


B2E
BUSINESS TO EDUCATION



SZCZECIŃSKI PARK
NAUKOWO-TECHNOLOGICZNY


JAVA
dla początkujących




Graficzny interfejs użytkownika - Swing



Slajd 30



B2E
BUSINESS TO EDUCATION



SZCZECIŃSKI PARK
NAUKOWO-TECHNOLOGICZNY

JAVA
dla początkujących

Ćwiczenie:


Napisz własnego zarządcę rozkładu („Layout”), aby uzyskać taki efekt:

Tekst 1


Tekst 1
Tekst 1

Tekst 1
Tekst 1
Tekst 1

Człowiek - najlepsza inwestycja



Projekt współfinansowany przez Unię Europejską w ramach Europejskiego Funduszu Społecznego



Graficzny interfejs użytkownika - Swing

```
import java.awt.Component;
```

```
import java.awt.Container;
```

```
import java.awt.Dimension;
```

```
import java.awt.LayoutManager;
```

```
public class DiagonalLayout implements LayoutManager {
```

```
    @Override
```

```
    public void addLayoutComponent(String name, Component comp) {}
```

```
    @Override
```

```
    public void layoutContainer(Container parent) {
```

```
        Component[] tab = parent.getComponents();
```

```
        Dimension dim = parent.getSize();
```

```
        int h = dim.height;
```

```
        int w = dim.width;
```

```
        tab[0].setBounds(tab[0].getWidth(), tab[0].getHeight(), w/2, h/5);
```

```
        tab[1].setBounds(tab[1].getWidth(), tab[1].getHeight(), w/3, (h*2)/5);
```

```
        tab[2].setBounds(tab[2].getWidth(), tab[2].getHeight(), (w*2)/3, (h*2)/5);
```

```
        tab[3].setBounds(tab[3].getWidth(), tab[3].getHeight(), w/4, (h*3)/5);
```

```
tab[4].setBounds(tab[4].getWidth(), tab[4].getHeight(), (w*2)/4, (h*3)/5);
tab[5].setBounds(tab[5].getWidth(), tab[5].getHeight(), (w*3)/4, (h*3)/5);
}
@Override
public Dimension minimumLayoutSize(Container parent) {
    return null;
}
@Override
public Dimension preferredLayoutSize(Container parent) {
    return null;
}
@Override
public void removeLayoutComponent(Component comp) {}
}
```

Opis założonych osiągnięć ucznia

Po tej lekcji uczniowie będą potrafili stworzyć proste okno ze skonfigurowanymi komponentami. Obsługą zmiany rozmiaru okna.

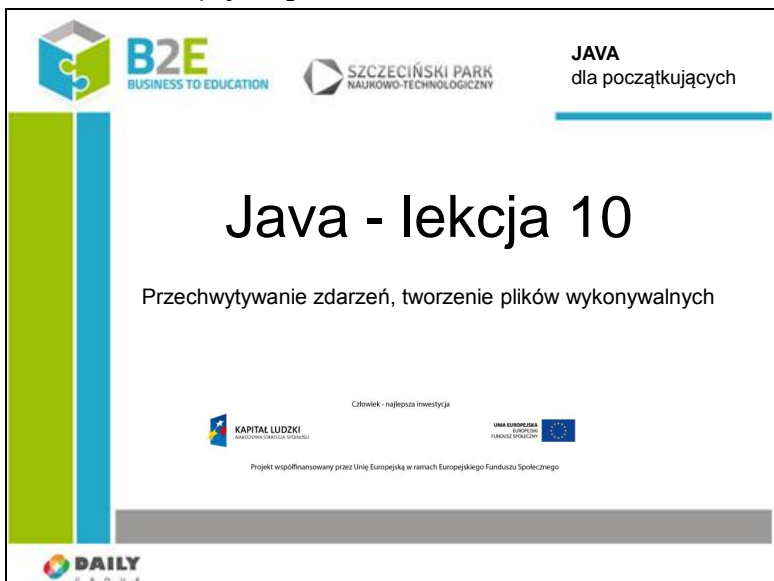
Lekcja 10 Przechwytywanie zdarzeń, tworzenie plików wykonywalnych

Cel lekcji

Celem lekcji jest pokazanie jak prawidłowo obsługiwać zdarzenia takie jak: ruchy kursora, starowania klawiatura. Omówiona zostanie budowa plików wykonywalnych jar.

Treść - slajdy z opisem

Slajd 1



Slajd 1

B2E BUSINESS TO EDUCATION

SZCZECIŃSKI PARK NAUKOWO-TECHNOLOGICZNY

JAVA dla początkujących

Java - lekcja 10

Przechwytywanie zdarzeń, tworzenie plików wykonywalnych

Człowiek - najlepsza inwestycja

KAPITAŁ LUDZKI NARODOWA STRATEGIA SPÓJNOŚCI


UNIA EUROPEJSKA EUROPEJSKI FUNDUSZ SPOŁECZNY

Projekt współfinansowany przez Unię Europejską w ramach Europejskiego Funduszu Społecznego


DAILY GROUP



Slajd 2



B2E
BUSINESS TO EDUCATION



SZCZECIŃSKI PARK
NAUKOWO-TECHNOLOGICZNY

JAVA
dla początkujących

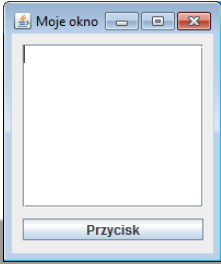
```


public class Start {
    public static void main(String [] args) {
        JFrame mainWindow = new JFrame("Moje okno");
        mainWindow.setBounds(200, 200, 300, 240);
        mainWindow.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        mainWindow.setLayout(null);

        JTextArea jTextArea = new JTextArea();
        JScrollPane jScrollPane = new JScrollPane(jTextArea);
        jScrollPane.setBounds(10, 10, 265, 150);

        JButton jButton = new JButton("Przycisk");
        jButton.setBounds(10, 170, 265, 20);


        mainWindow.add(jScrollPane);
        mainWindow.add(jButton);
        mainWindow.setVisible(true);
    }
}
    
```






Przechwytywanie zdarzeń, tworzenie plików wykonywalnych

Slajd 3

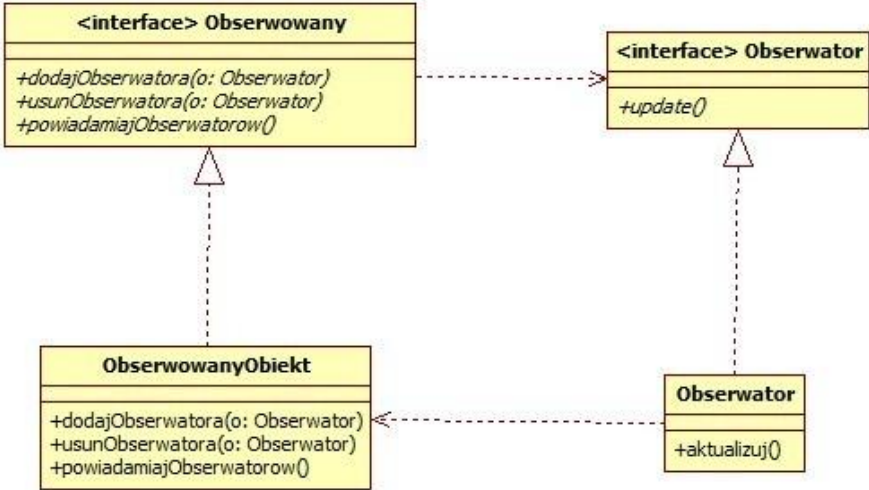


B2E
BUSINESS TO EDUCATION




SZCZECIŃSKI PARK
NAUKOWO-TECHNOLOGICZNY

JAVA
dla początkujących



```

classDiagram
    class Obserwowany {
        <<interface>>
        +dodajObserwatora(o: Obserwator)
        +usunObserwatora(o: Obserwator)
        +powiadamiajObserwatorow()
    }
    class Obserwator {
        <<interface>>
        +update()
    }
    class ObserwowanyObiekt {
        +dodajObserwatora(o: Obserwator)
        +usunObserwatora(o: Obserwator)
        +powiadamiajObserwatorow()
    }
    class Obserwator {
        +aktualizuj()
    }
    ObserwowanyObiekt ..|> Obserwowany
    Obserwator ..|> Obserwator
    Obserwator ..> Obserwowany
    
```



Przechwytywanie zdarzeń, tworzenie plików wykonywalnych

W programowaniu obiektowym obiekty posiadają pewien stan, tj. zbiór aktualnych wartości pól obiektu, który w wyniku wykonywania na nich operacji może ulegać zmianie. Od bieżącego stanu mogą być zależne inne obiekty, dlatego musi istnieć możliwość ich powiadomienia o jego zmianie tak, aby mogły one się do niej dostosować. Możemy także żądać, aby inne obiekty były powiadamiane o tym, że inny obiekt próbuje wykonać konkretną czynność, np. ponownie nawiązywać utracone połączenie z bazą danych. Pragniemy zaimplementować ogólny mechanizm, który umożliwi nam osiągnięcie tych celów.

We wzorcu obserwator wyróżniamy dwa podstawowe typy obiektów:

obserwowany (ang. observable, subject) – obiekt, o którym chcemy uzyskiwać informacje,

obserwator (ang. observer, listener) – obiekty oczekujące na powiadomienie o zmianie stanu obiektu obserwowanego.

Kiedy stan obiektu obserwowanego się zmienia, wywołuje on metodę "powiadomObserwatorow()", która wysyła powiadomienia do wszystkich zarejestrowanych obserwatorów.

Obserwator jest stosowany w aplikacjach z graficznym interfejsem użytkownika.

Slajd 4



JAVA dla początkujących

```
import java.awt.event.MouseEvent;
import java.awt.event.MouseListener;
import javax.swing.JTextArea;



public class MyListener implements MouseListener {
    private JTextArea jTextArea;
    public MyListener(JTextArea jTextArea) {
        this.jTextArea = jTextArea;
    }
    @Override
    public void mouseClicked(MouseEvent arg0) {
        jTextArea.setText(jTextArea.getText()
            + "\nnastąpiło kliknięcie");
    }
    @Override
    public void mouseEntered(MouseEvent arg0) {
        jTextArea.setText(jTextArea.getText()
            + "\nkursor wszedł w obszar przycisku");
    }
}
```



Przechwytywanie zdarzeń, tworzenie plików wykonywalnych




Slajd 5





JAVA dla początkujących

```
@Override  
public void mouseExited(MouseEvent arg0) {  
    JTextArea.setText(JTextArea.getText()  
        + "\nkursor wyszedł z obszaru przycisku");  
  
@Override  
public void mousePressed(MouseEvent arg0) {  
    JTextArea.setText(JTextArea.getText()  
        + "\nprzycisk naciśnięty");  
  
@Override  
public void mouseReleased(MouseEvent arg0) {  
    JTextArea.setText(JTextArea.getText()  
        + "\nprzycisk zwolniony");  
  
}
```


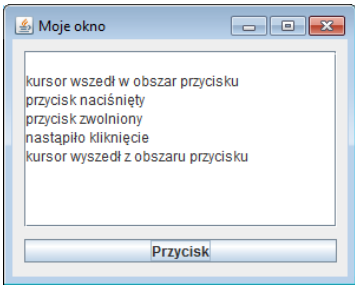


Przechwytywanie zdarzeń, tworzenie plików wykonywalnych

Slajd 6



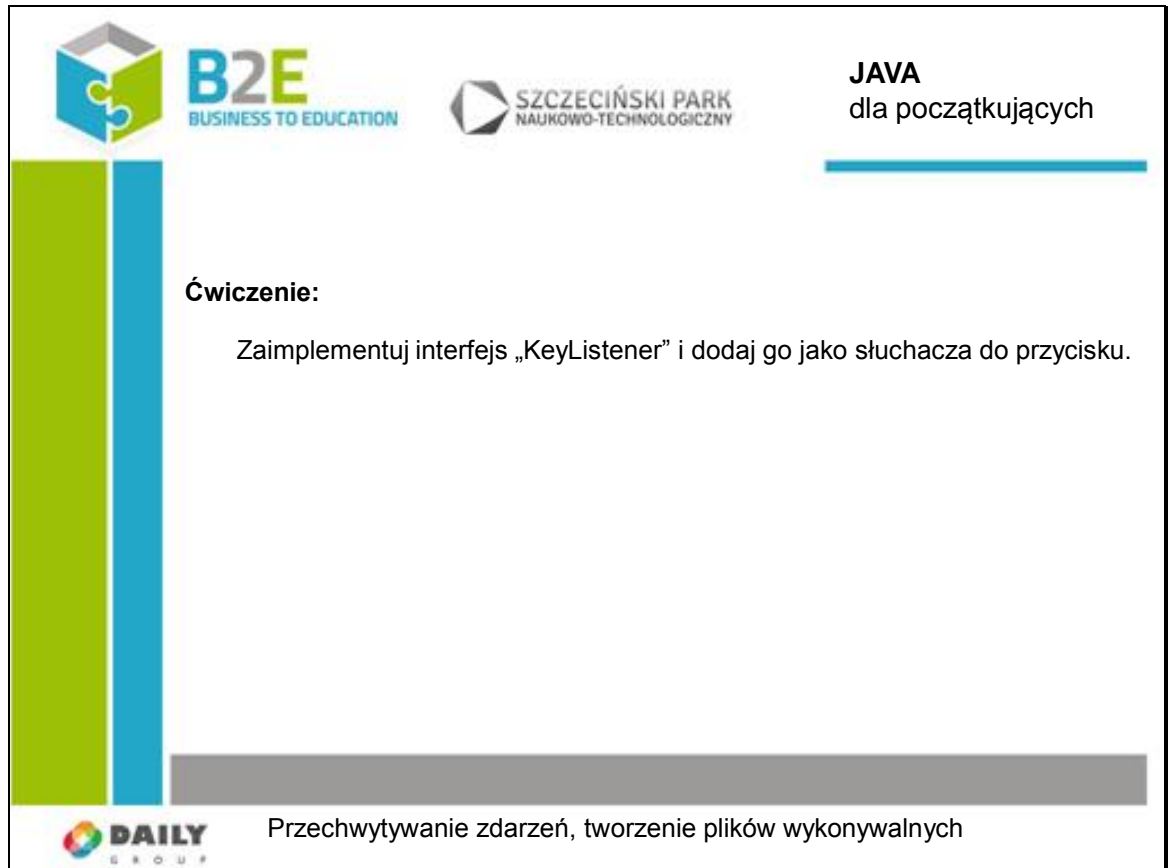
JAVA dla początkujących



Przechwytywanie zdarzeń, tworzenie plików wykonywalnych



Slajd 7



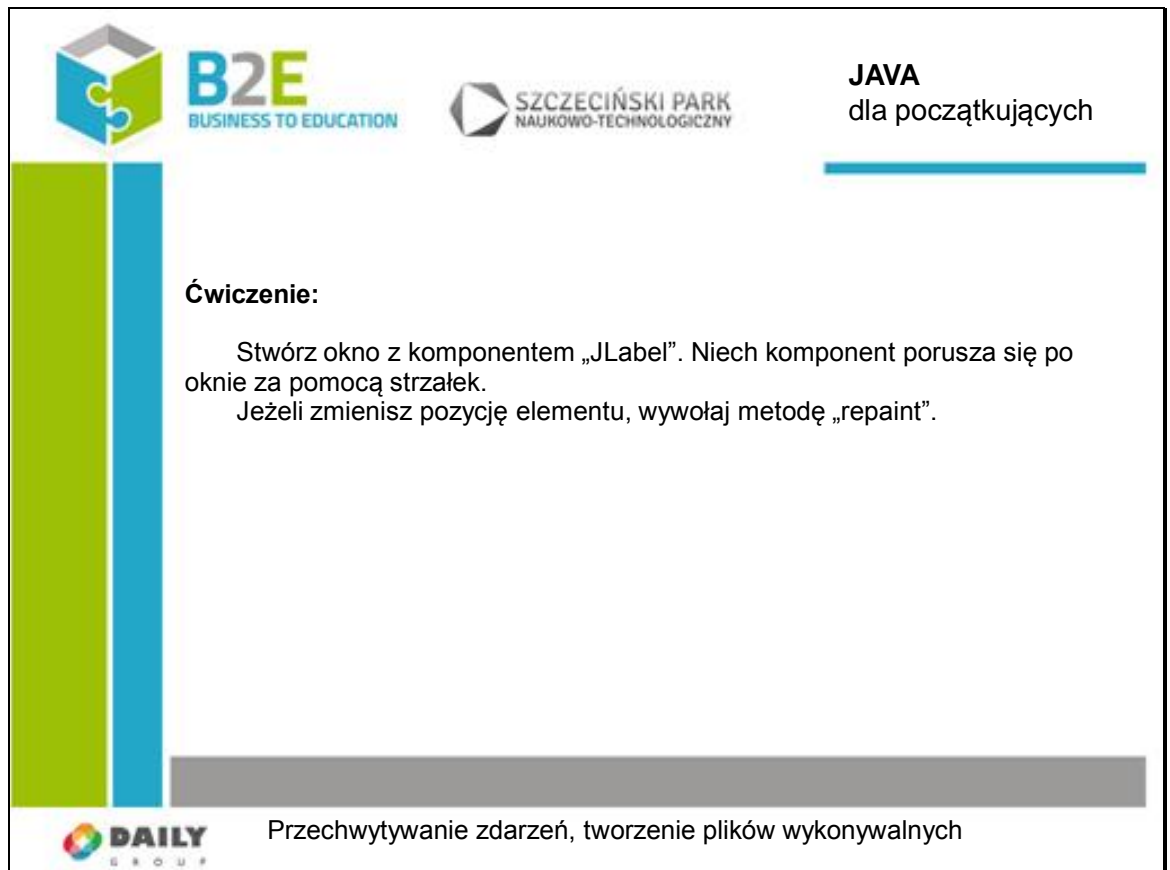
The slide features a header with logos for B2E (Business to Education), Szczeciński Park Naukowo-Technologiczny, and the Daily Group. The main content area contains the text 'Ćwiczenie: Zaimplementuj interfejs „KeyListener” i dodaj go jako słuchacza do przycisku.' Below this, a grey bar contains the text 'Przechwytywanie zdarzeń, tworzenie plików wykonywalnych'. The slide is titled 'JAVA dla początkujących' in the top right corner.

```
import java.awt.event.KeyEvent;
import java.awt.event.KeyListener;
import javax.swing.JTextArea;
public class MyKeyboardListener implements KeyListener {
    private JTextArea jTextArea;
    public MyKeyboardListener(JTextArea jTextArea) {
        this.jTextArea = jTextArea;
    }
    @Override
    public void keyPressed(KeyEvent e) {
        jTextArea.setText(jTextArea.getText()
            + "\nnastąpiło wciśnięcie klawisza "
            + e.getKeyChar());
    }
    @Override
    public void keyReleased(KeyEvent e) {
        jTextArea.setText(jTextArea.getText()
            + e.getKeyChar());
    }
}
```



```
        + "\nnastąpiło zwolnienie klawisza "  
        + e.getKeyChar());  
    }  
    @Override  
    public void keyTyped(KeyEvent e) {  
        JTextArea.setText(JTextArea.getText()  
            + "\nzostał wpisany znak "  
            + e.getKeyChar());  
    }  
}  
jButton.addKeyListener(new MyKeyboardListener(jTextArea));
```

Slajd 8



The slide features a header with logos for B2E (Business to Education), Szczeciński Park Naukowo-Technologiczny, and the Daily Group. The title is 'JAVA dla początkujących'. The main content is an exercise: 'Ćwiczenie: Stwórz okno z komponentem „JLabel”. Niech komponent porusza się po oknie za pomocą strzałek. Jeżeli zmienisz pozycję elementu, wywołaj metodę „repaint”.' The footer contains the text 'Przechwytywanie zdarzeń, tworzenie plików wykonywalnych' and the Daily Group logo.

```
public class Start {  
    public static void main(String [] args) {  
        JFrame mainFrame = new JFrame("Moje okno");  
        mainFrame.setBounds(200, 200, 300, 240);  
        mainFrame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);  
        mainFrame.setLayout(null);  
    }  
}
```

```
JLabel jLabel = new JLabel("Tekst");
jLabel.setBounds(50, 50, 50, 20);
mainFrame.addKeyListener(new MyKeyListener(jLabel, mainFrame));
mainFrame.add(jLabel);
mainFrame.setVisible(true);
}
}

public class MyKeyListener implements KeyListener {
    private JLabel jLabel;
    private JFrame jFrame;
    private int x = 50;
    private int y = 50;
    public MyKeyListener(JLabel jLabel, JFrame jFrame) {
        this.jLabel = jLabel;
        this.jFrame = jFrame;}
    @Override
    public void keyPressed(KeyEvent e) {
        int d = 10;
        if (e.getKeyCode() == KeyEvent.VK_RIGHT) {
            x += x>100 ? 0 : d;
            jLabel.setBounds(x, y, jLabel.getWidth(), jLabel.getHeight());
        } else if (e.getKeyCode() == KeyEvent.VK_LEFT) {
            x -= x<10 ? 0 : d;
            jLabel.setBounds(x, y, jLabel.getWidth(), jLabel.getHeight());
        } else if (e.getKeyCode() == KeyEvent.VK_UP) {
            y -= y<10 ? 0 : d;
            jLabel.setBounds(x, y, jLabel.getWidth(), jLabel.getHeight());
        } else if (e.getKeyCode() == KeyEvent.VK_DOWN) {
            y += y>100 ? 0 : d;
            jLabel.setBounds(x, y, jLabel.getWidth(), jLabel.getHeight());
        }
        jFrame.repaint();
    }
    @Override
```




```

public void keyReleased(KeyEvent e) {}

@Override

public void keyTyped(KeyEvent e) {}

}
    
```

Slajd 9

B2E BUSINESS TO EDUCATION **SZCZECIŃSKI PARK** NAUKOWO-TECHNOLOGICZNY **JAVA** dla początkujących

```

public class Start {

    public class MyInnerClass{

    }

    public static void main(String [] args) {
        JFrame mainFrame = new JFrame("Moje okno");
        mainFrame.setBounds(200, 200, 300, 240);
        mainFrame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    }
}
    
```

DAILY GROUP Przechwytywanie zdarzeń, tworzenie plików wykonywalnych

Do przykładu z początku lekcji dodaj klasę wewnętrzną i zapisz zmiany. Dla pewności uruchom program.

Slajd 10

B2E BUSINESS TO EDUCATION **SZCZECIŃSKI PARK** NAUKOWO-TECHNOLOGICZNY **JAVA** dla początkujących

- .settings
- bin
- src
- .classpath
- .project

- MyKeyboardListener.java
- MyListener.java
- Start.java

- MyKeyboardListener.class
- MyListener.class
- Start\$MyInnerClass.class
- Start.class

DAILY GROUP Przechwytywanie zdarzeń, tworzenie plików wykonywalnych

W projekcie znajdują się trzy pliki źródłowe:



- MyKeyListener.java,
- MyListener.java,
- Start.java.

Wejść do folderu workspace, następnie do folderu z tym projektem. W środku znajdują się dwa foldery "src" i "bin".

Sprawdź ich zawartość. W folderze "src" znajdują się pliki źródłowe i są trzy.

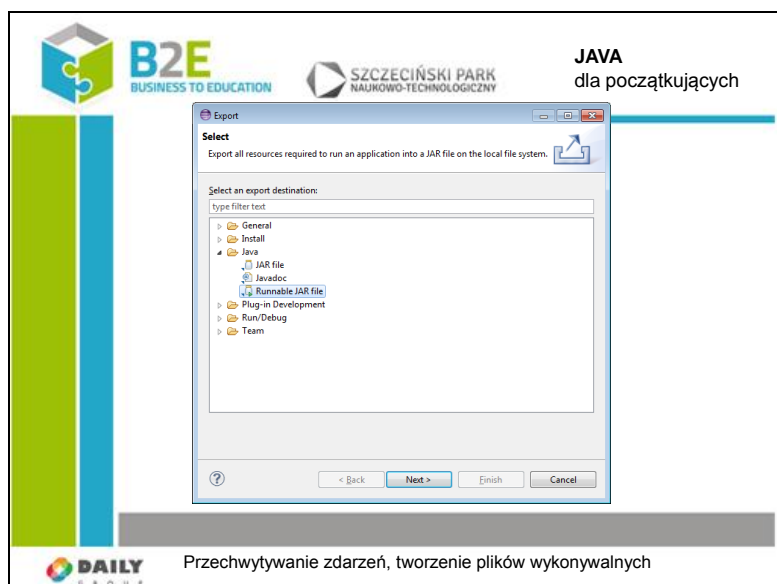
Dlaczego w folderze są cztery pliki?

Na jednej z pierwszych lekcji powiedziane było, że w jednym pliku znajdować się może tylko jedna klasa.

W plikach źródłowych mogą być klasy wewnętrzne, ale po procesie kompilacji każda klasa stanowi osobny plik.

W klasie "Start" jest klasa wewnętrzna "MyInnerClass". Po skompilowaniu jest osobnym plikiem. Po nazwie można łatwo rozpoznać, z której klasy została wyodrębniona.

Slajd 11



W środowisku Eclipse kliknij prawym przyciskiem na projekt i użyj opcji eksport.

Wybierz opcję "Java/Runnable JAR file".



Slajd 12

JAVA
dla początkujących

Przechwytywanie zdarzeń, tworzenie plików wykonywalnych

Wybierz plik, który służy do uruchamiania aplikacji. Następnie podaj nazwę i lokalizację pliku.



Slajd 13

B2E
BUSINESS TO EDUCATION

SZCZECIŃSKI PARK
NAUKOWO-TECHNOLOGICZNY

JAVA
dla początkujących

```
C:\Windows\system32\cmd.exe  
C:\tmp\szkolenie>java -jar lekcja10.jar
```

DAILY
GROUP

Przechwytywanie zdarzeń, tworzenie plików wykonywalnych

Uruchom konsolę i wywołaj polecenie tak, jak na slajdzie.

Powinno nastąpić prawidłowe uruchomienie programu.

Zmień rozszerzenie pliku z "jar" na "zip" i rozpakuj go.

Zauważ, że jest to zwykłe archiwum zip. Znajdują się w nim wszystkie pliki bajtkodu, należące do danego projektu.

W folderze "META-INF" znajduje się plik manifestu. Zobacz jego zawartość.

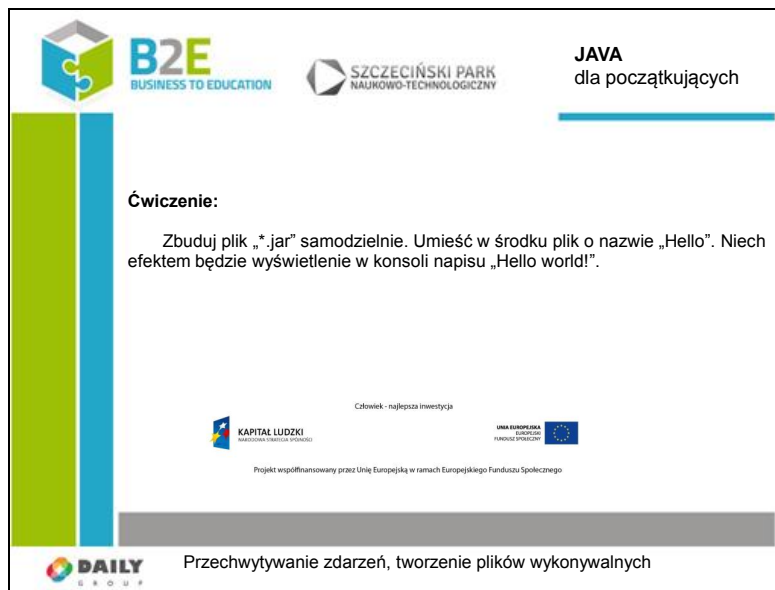
Manifest-Version: 1.0

Class-Path: .

Main-Class: Start

W ostatnim wersie jest nazwa klasy, z której wywoływany jest program.

Slajd 14



B2E BUSINESS TO EDUCATION **SZCZECIŃSKI PARK** NAUKOWO-TECHNOLOGICZNY **JAVA** dla początkujących

Ćwiczenie:

Zbuduj plik „*.jar” samodzielnie. Umieść w środku plik o nazwie „Hello”. Niech efektem będzie wyświetlenie w konsoli napisu „Hello world!”.

Człowiek - najlepsza inwestycja

KAPITAŁ LUDZKI NARODOWA STRATEGIA SPÓJNOŚCI UNIA EUROPEJSKA EUROPEJSKI FUNDUSZ SPOŁECZNY

Projekt współfinansowany przez Unię Europejską w ramach Europejskiego Funduszu Społecznego

DAILY GROUP Przechwytywanie zdarzeń, tworzenie plików wykonywalnych

Opis założonych osiągnięć ucznia

Po tej lekcji uczniowie będą umieli obsłużyć zdarzenia systemowe związane z kursorem i klawiaturą. Będą rozumieć jak wygląda tworzenie plików wykonywalnych.


Lekcja 11 Projekt Interdyscyplinarny

Cele Projektu

Celem projektu jest zdobycie umiejętności wykorzystania dotychczasowej wiedzy podczas programowania aplikacji mobilnych na platformie Android.

Treść – slajdy z opisem

Slajd 1



B2E BUSINESS TO EDUCATION **SZCZECIŃSKI PARK** NAUKOWO-TECHNOLOGICZNY **JAVA** dla początkujących

Java - lekcja 11

Projekt interdyscyplinarny:
baza teleadresowa kolegów i koleżanek ze szkoły
Aplikacja mobilna na platformę Android

Człowiek - najlepsza inwestycja


KAPITAŁ LUDZKI NARODOWA STRATEGIA SPÓJNOŚCI UNIA EUROPEJSKA EUROPEJSKI FUNDUSZ SPOŁECZNY


Projekt współfinansowany przez Unię Europejską w ramach Europejskiego Funduszu Społecznego

DAILY GROUP



Slajd 2


 **B2E**
BUSINESS TO EDUCATION

 **SZCZECIŃSKI PARK**
NAUKOWO-TECHNOLOGICZNY

JAVA
dla początkujących

Założenia:

- widok klas w szkole z nazwa klasy, imieniem i nazwiskiem wychowawcy oraz zdjęciem wychowawcy
- widok uczniów w klasie z imieniem i nazwiskiem ucznia oraz zdjęciem ucznia
- widok pojedynczego ucznia ze zdjęciem, imieniem i nazwiskiem, adresem e-mail i numerem telefonu

 **DAILY**
GROUP

Aplikacja mobilna na platformę Android

Aplikacja będzie służyć do prezentacji danych klas oraz uczniów w szkole. Jako dodatkowa funkcjonalność będzie pozwalać na szybkie wybranie numeru ucznia i wykonanie telefonu lub wysłanie e-mail'a.

Slajd 3

 **B2E**
BUSINESS TO EDUCATION

 **SZCZECIŃSKI PARK**
NAUKOWO-TECHNOLOGICZNY

JAVA
dla początkujących

Przygotowanie środowiska pracy:

- www.oracle.com
- Java JRE (Java Runtime Environment)
- Java JDK (Java Development Kit)
- <https://developer.android.com/sdk/installing/studio.html>
- Android Studio

 **DAILY**
GROUP

Aplikacja mobilna na platformę Android

Środowisko uruchomieniowe Javy, jak i pakiet developerski, mamy już zainstalowane.

Na tej lekcji będziemy używali innego środowiska developerskiego. O ile możliwe jest tworzenie aplikacji na platformę Android w Eclipse, o tyle aktualnie zalecanym środowiskiem jest Android Studio. Wobec czego skorzystamy z doskonalszego IDE podczas tej lekcji.



Slajd 4

B2E BUSINESS TO EDUCATION

SZCZECIŃSKI PARK
NAUKOWO-TECHNOLOGICZNY

JAVA
dla początkujących

Developers - Design - Develop - Distribute

Training API Guides Reference Tools Google Services Samples

Download

Android Studio BETA

Android Studio is a new Android development environment based on IntelliJ IDEA. It provides new features and improvements over Eclipse ADT and will be the official Android IDE once it's ready. On top of the capabilities you expect from IntelliJ, Android Studio offers:

- Flexible Gradle-based build system.
- Build variants and multiple APK generation.
- Expanded template support for Google Services and various device types.
- Rich layout editor with support for theme editing.
- Lint tools to catch performance, usability, version compatibility, and other problems.
- ProGuard and app-signing capabilities.
- Built-in support for Google Cloud Platform (GCP), making it easy to integrate Google Cloud Messaging and app engine.

Download Android Studio Beta v0.8.8 with the Android SDK for Windows

This download includes:

- Android Studio Beta
- All the Android SDK Tools to design, test, and debug your app
- A version of the Android platform to compile your app
- A version of the Android system image to run your app in the emulator

Caution: Android Studio is currently in beta. Some features are not yet implemented and you may encounter bugs. If you are not comfortable using an unfinished product, you may want to instead download (or continue to use) Eclipse with ADT.

VIEW ALL DOWNLOADS AND SIZES

SYSTEM REQUIREMENTS

DAILY GROUP Aplikacja mobilna na platformę Android

Pobieramy oprogramowanie, lub korzystamy z dostarczonej wersji.

Slajd 5

B2E BUSINESS TO EDUCATION

SZCZECIŃSKI PARK
NAUKOWO-TECHNOLOGICZNY

JAVA
dla początkujących

Android Studio Setup

Welcome to the Android Studio Setup Wizard

This wizard will guide you through the installation of Android Studio.

It is recommended that you close all other applications before starting Setup. This will make it possible to update relevant system files without having to reboot your computer.

Click Next to continue.


Next > Cancel


DAILY GROUP Aplikacja mobilna na platformę Android

Rozpoczynamy instalację IDE

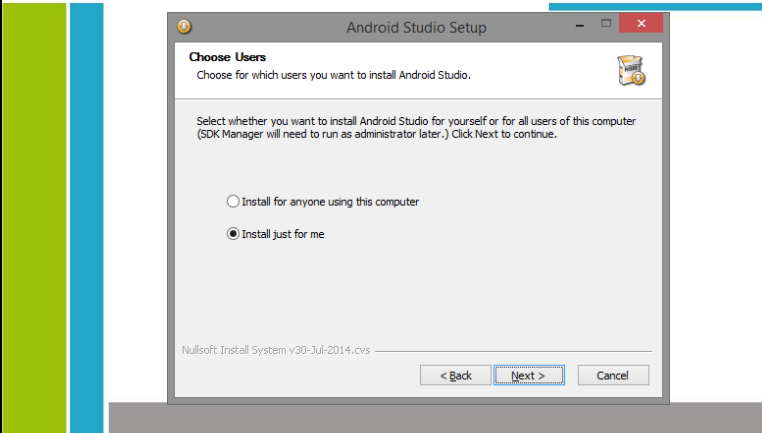


Slajd 6

 **B2E**
BUSINESS TO EDUCATION

 SZCZECIŃSKI PARK
NAUKOWO-TECHNOLOGICZNY


JAVA
dla początkujących




DAILY
GROUP

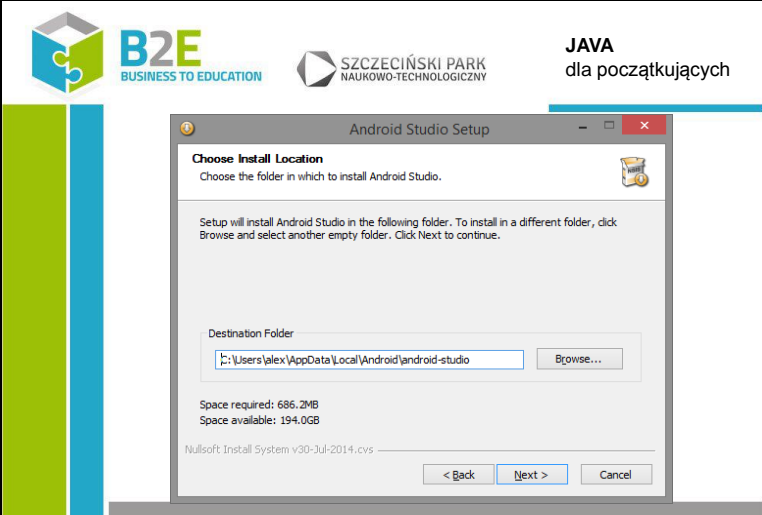
Aplikacja mobilna na platformę Android

Slajd 7

 **B2E**
BUSINESS TO EDUCATION

 SZCZECIŃSKI PARK
NAUKOWO-TECHNOLOGICZNY

JAVA
dla początkujących



DAILY
GROUP

Aplikacja mobilna na platformę Android



Slajd 8

B2E
BUSINESS TO EDUCATION

SZCZECIŃSKI PARK
NAUKOWO-TECHNOLOGICZNY

JAVA
dla początkujących

Android Studio Setup

Choose Start Menu Folder

Choose a Start Menu folder for the Android Studio shortcuts.

Select the Start Menu folder in which you would like to create the program's shortcuts. You can also enter a name to create a new folder.

Android Studio

7-Zip
888poker
Accessibility
Accessories
Administrative Tools
Aplikacje Chrome
Bitcoin
Bitvise SSH Client
Cisco
ClipX
Cygwin

Do not create shortcuts

Nullsoft Install System v30-Jul-2014.cvs

< Back Install Cancel

DAILY
GROUP

Aplikacja mobilna na platformę Android

Slajd 9

B2E
BUSINESS TO EDUCATION

SZCZECIŃSKI PARK
NAUKOWO-TECHNOLOGICZNY

JAVA
dla początkujących

Android Studio Setup

Installing

Please wait while Android Studio is being installed.

Extract: appcompat-v7-19.0.0.aar

Show details

Nullsoft Install System v30-Jul-2014.cvs

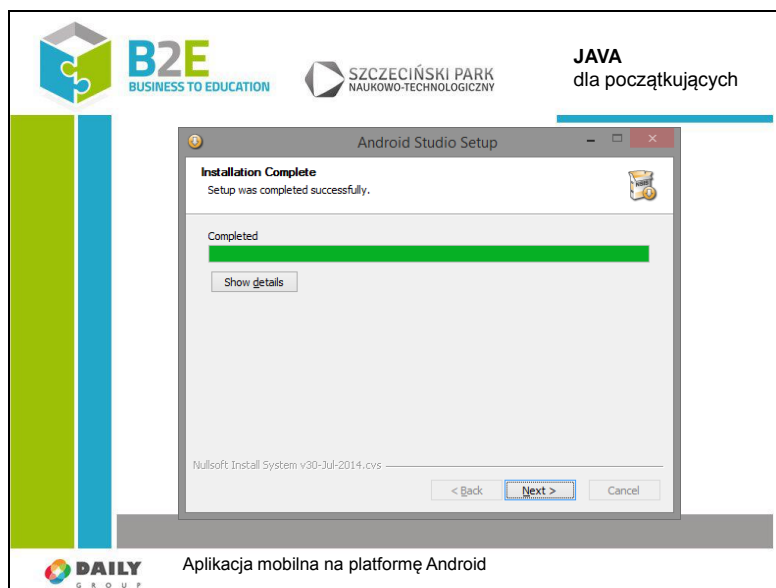
< Back Next > Cancel

DAILY
GROUP

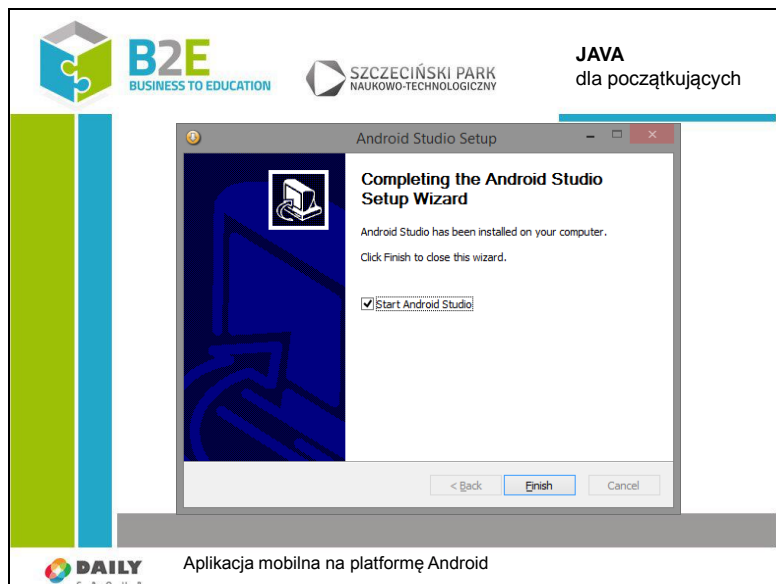
Aplikacja mobilna na platformę Android



Slajd 10



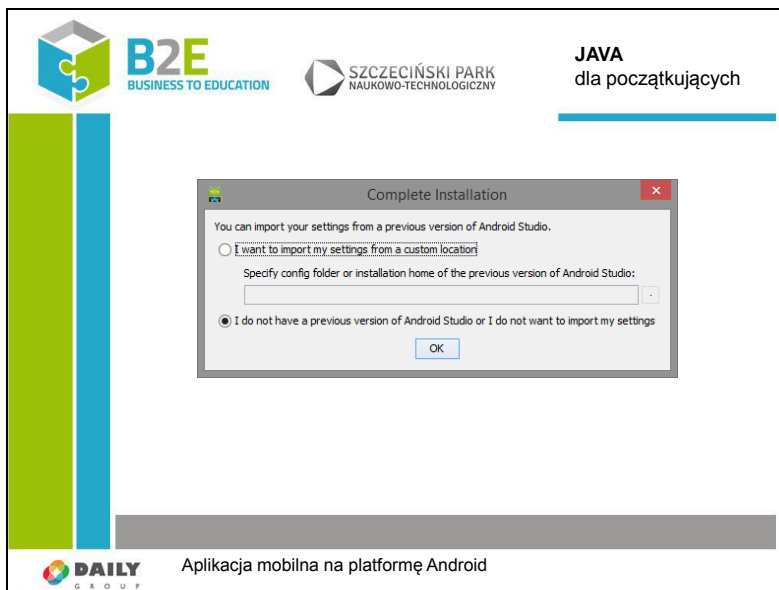
Slajd 11



Kończymy instalację i uruchamiamy Android Studio.

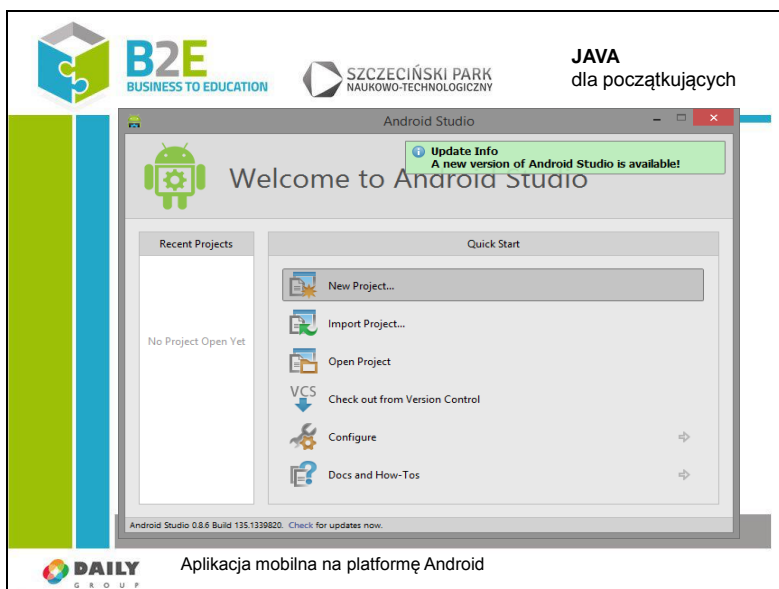


Slajd 12



Jeśli pojawi się opcja importu ustawień, to odrzucamy to udogodnienie. W przeciwnym razie moglibyśmy zacząć pracować na niestandardowych ustawieniach poprzedniego użytkownika komputera.

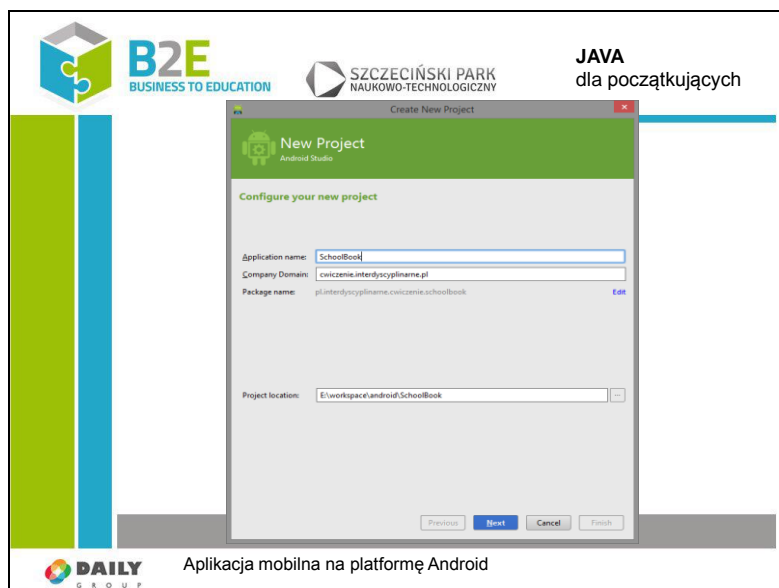
Slajd 13



Tworzymy nowy projekt.

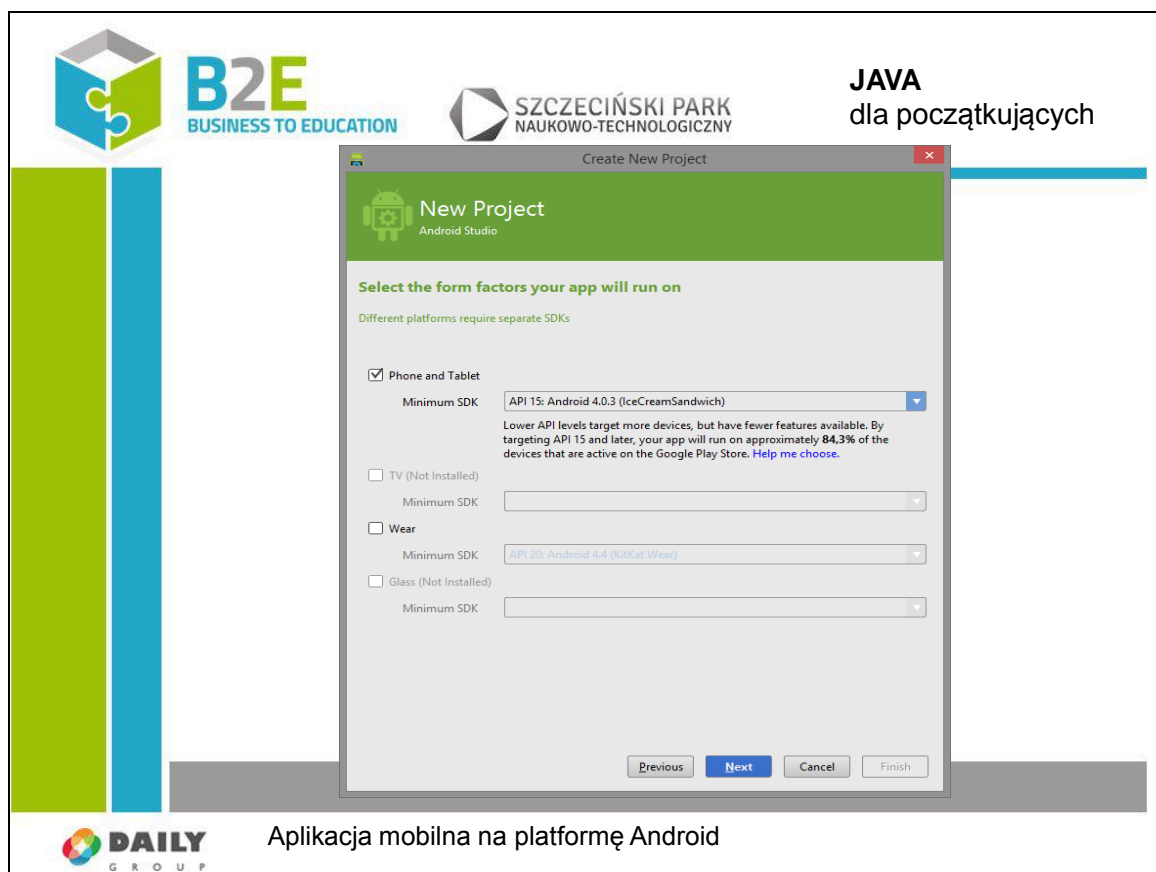


Slajd 14



Nadajemy projektowi nazwę, domyślny pakiet oraz lokalizację na dysku twardym.

Slajd 15

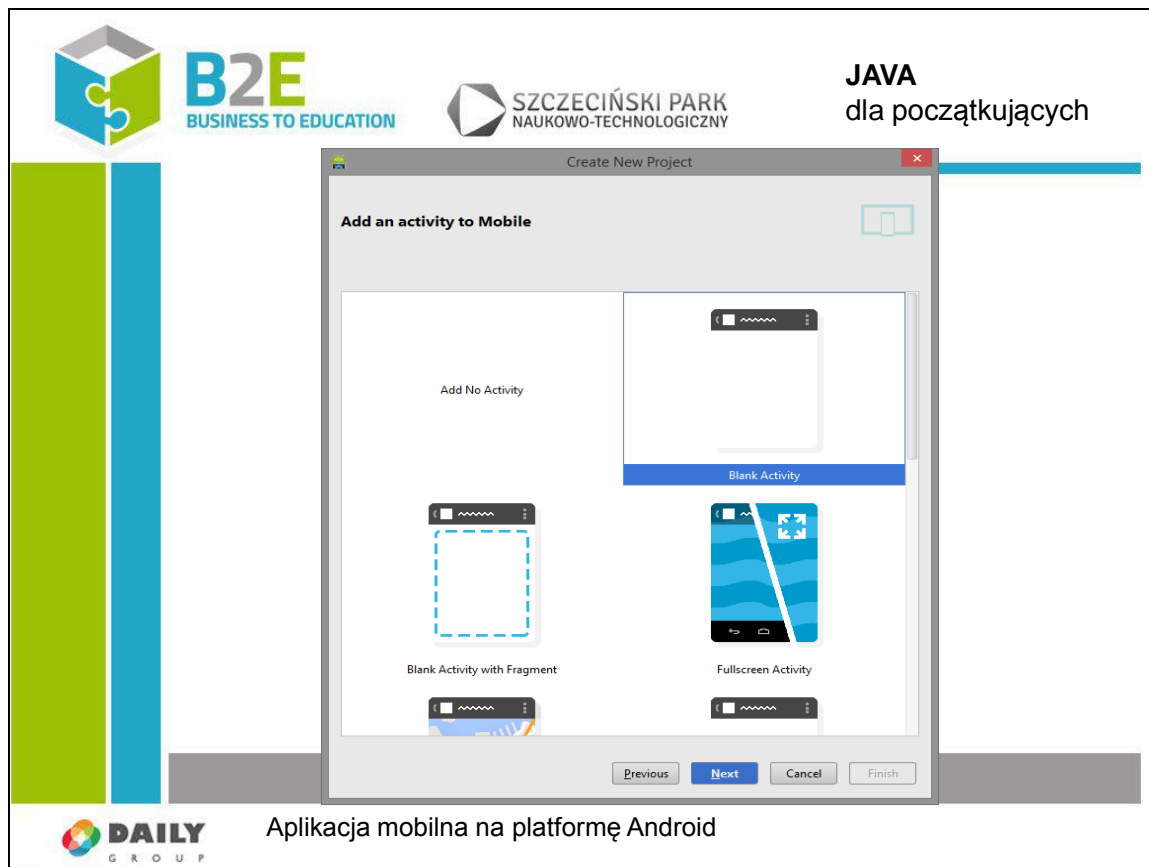


Wybieramy minimalną wersję API Androida od której nasza aplikacja będzie mogła zostać uruchomiona. Należy zwrócić uwagę że czym niższa wersja zostanie wybrana, tym więcej urządzeń będzie mogło uruchomić naszą aplikację. Jednak oznacza to również utratę nowych funkcjonalności dostępnych tylko w wyższych wersjach systemu. Istnieje możliwość emulacji nowych funkcji na starszych urządzeniach ale niestety wykracza to poza zakres tej lekcji.

Wybieramy w miarę nowe API na potrzeby tej lekcji.



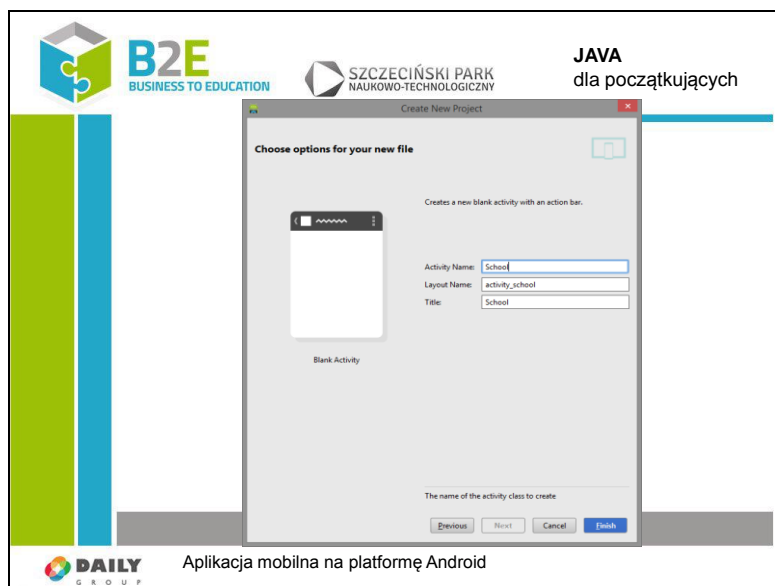
Slajd 16



Kolejnym krokiem jest wybór pustego „activity”. Aplikacje w Androidzie składają się z widoków, w nomenklaturze Androida jest to właśnie „activity”. Dla przykładu, nasza aplikacja będzie składać się z trzech widoków:

- listy klas w szkole,
- listy uczniów w klasie,
- szczegółów danego ucznia.

Slajd 17

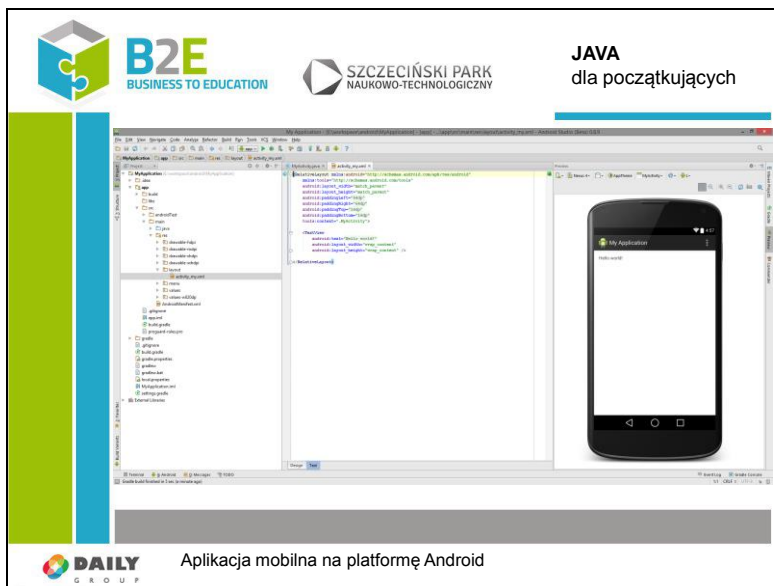


Nadajemy nazwę nowemu widokowi i kończymy działanie kreatora. Musimy chwilę poczekać aż IDE wygeneruje nasz nowy projekt.



Projekt współfinansowany przez Unię Europejską w ramach Europejskiego Funduszu Społecznego

Slajd 18



Naszym oczom ukazuje się interfejs IDE i kod przykładowego pustego activity. Jak widać widok od razu jest renderowany aby developer mógł jak najszybciej widzieć wprowadzone zmiany. Istnieje również możliwość edycji wizualnej widoków, w tym celu trzeba się przełączyć na inną zakładkę „Design”.

Slajd 19





Podczas tworzenia aplikacji możemy ją uruchamiać w emulatorze lub na prawdziwym sprzęcie. W ramach zajęć będziemy korzystać z emulatora. W tym celu należy zainstalować pliki emulujące interesujące nas urządzenie.

Uruchamiamy Android SDK manager.



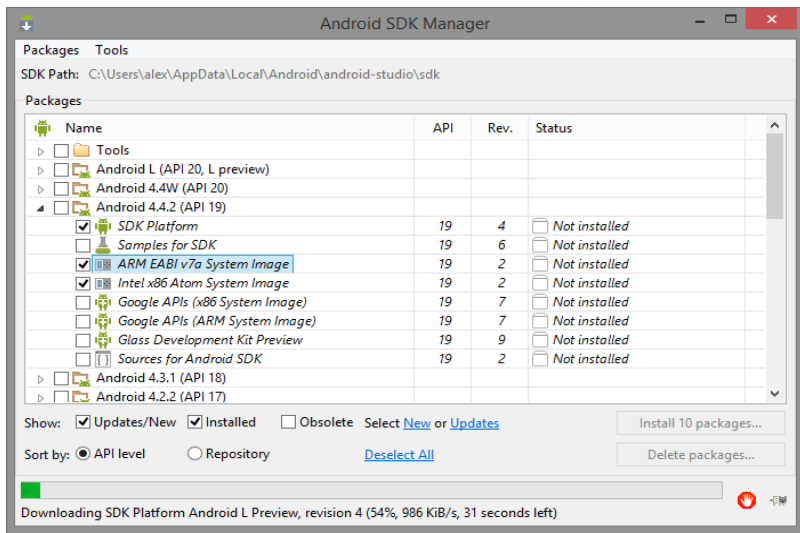
Slajd 20






JAVA

dla początkujących





Aplikacja mobilna na platformę Android

Zaznaczamy „ARM EABI v7a System Image” i instalujemy. Trzeba chwilę poczekać.

Slajd 21





JAVA

dla początkujących





Aplikacja mobilna na platformę Android

Po instalacji czas uruchomić naszą pierwszą aplikację na Androida. Naciskamy zielony przycisk strzałki w górnej części IDE.



Slajd 22

B2E BUSINESS TO EDUCATION

SZCZECIŃSKI PARK NAUKOWO-TECHNOLOGICZNY

JAVA dla początkujących

Choose Device

Choose a running device

Device	Serial Number	State	Conn.
Nothing to show			

Launch emulator

Android virtual device: myDevice

Use same device for future launches

OK Cancel

DAILY GROUP Aplikacja mobilna na platformę Android

Pojawi się okno wyboru urządzenia. Powinno być puste, jako że nie dysponujemy jeszcze wirtualnymi urządzeniami. Zaznaczamy „lunch emulator” i klikamy na przycisk z kropkami.

Slajd 23

B2E BUSINESS TO EDUCATION

SZCZECIŃSKI PARK NAUKOWO-TECHNOLOGICZNY

JAVA dla początkujących

Edit Android Virtual Device (AVD)

AVD Name: myDevice

Device: Nexus 4 (4.7", 768 x 1280: xhdpi)

Target: Android 4.4.2 - API Level 19

CPU/ABI: ARM (armeabi-v7a)

Keyboard: Hardware keyboard present

Skin: Skin with dynamic hardware controls

Front Camera: None

Back Camera: None

Memory Options: RAM: 512 VM Heap: 64

Internal Storage: 400 MiB

SD Card: Size: MiB File: Browse...

Emulation Options: Snapshot Use Host GPU

Override the existing AVD with the same name

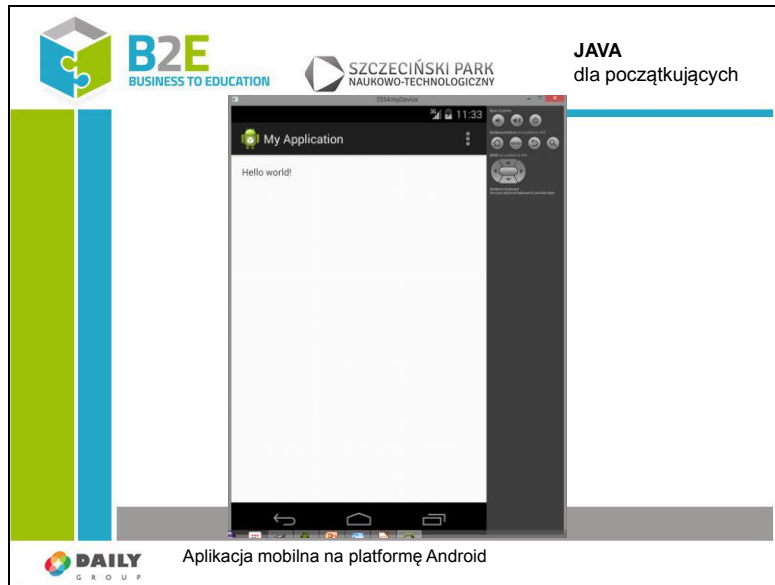
OK Cancel

DAILY GROUP Aplikacja mobilna na platformę Android

Startuje AVD – Android Virtual Device Maanager. Wciskamy przycisk „Create” i wypełniamy specyfikację techniczną wirtualnego urządzenia. Po czym klikamy OK i czekamy na utworzenie urządzenia.

Teraz z listy wybieramy nasze urządzenie i klikamy OK. Nasza aplikacja zostanie uruchomiona w emulatorze.

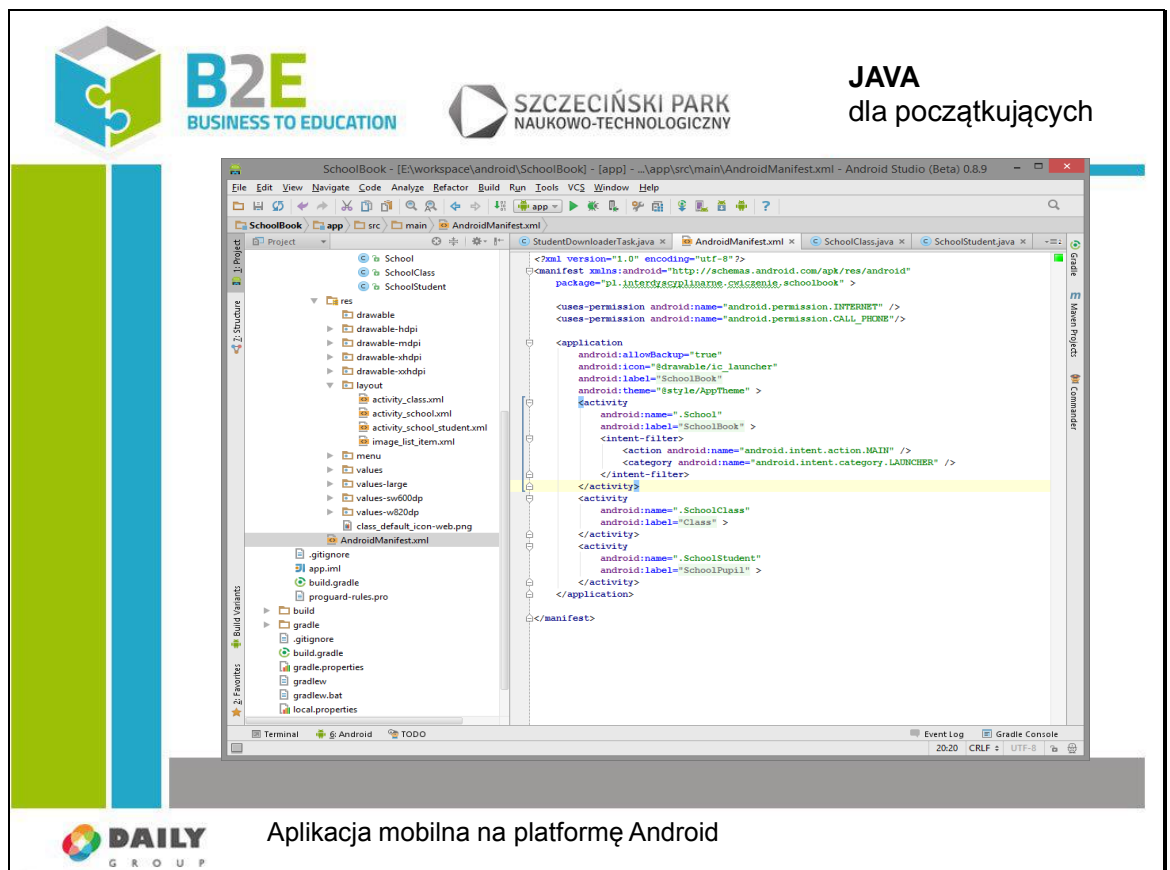
Slajd 24



Aplikacja się uruchomiła. Proszę nie wyłączać emulatora, gdyż strasznie długo się uruchamia, za to można go wykorzystać wielokrotnie podczas kolejnych uruchomień naszej aplikacji.

Teraz przejrzymy najważniejsze pliki w projekcie i zaczniemy budować naszą aplikację.

Slajd 25



Jednym z ważniejszych plików jest plik „AndroidManifest.xml”. Jest to plik konfiguracyjny aplikacji, gdzie można ustawić wiele opcji takich jak nazwę, czy ikonę programu. Zarejestrowane są tu też wszystkie widoki aplikacji, ale proszę ich nie dodawać ręcznie gdyż IDE zrobi to za nas w swoim czasie.


Kolejnym istotnym elementem są uprawnienia aplikacji. Aplikacja musi jasno zadeklarować z jakich funkcji systemu chce korzystać. Dzięki temu podczas instalowania aplikacji użytkownik może zdecydować czy aplikacja jest bezpieczna i czy chce ją uruchomić na swoim sprzęcie. My poprosimy


o dwa przywileje: dostęp do Internetu – stamtąd będziemy pobierać informacje do wyświetlenia oraz o możliwość inicjowania połączeń głosowych. Proszę na początku pliku dodać:

```
<uses-permission android:name="android.permission.INTERNET" />
```

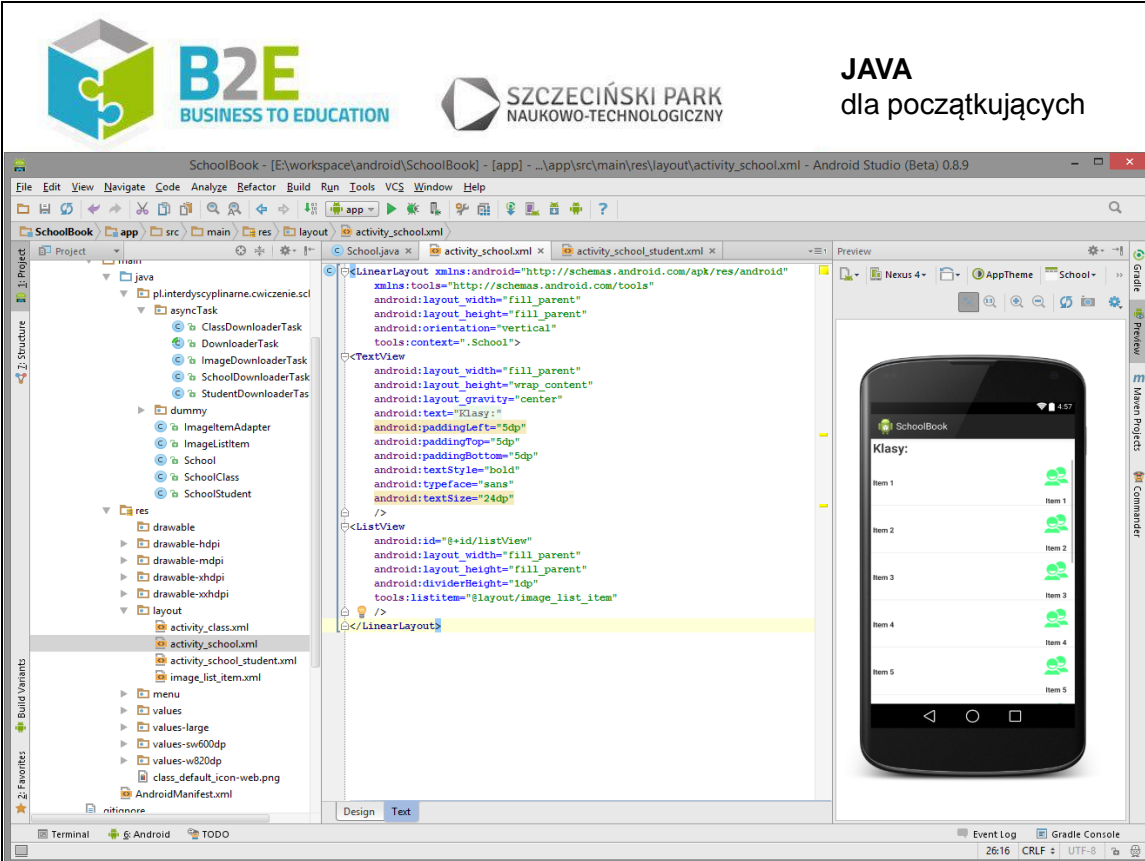
```
<uses-permission android:name="android.permission.CALL_PHONE"/>
```

Slajd 26





JAVA
dla początkujących



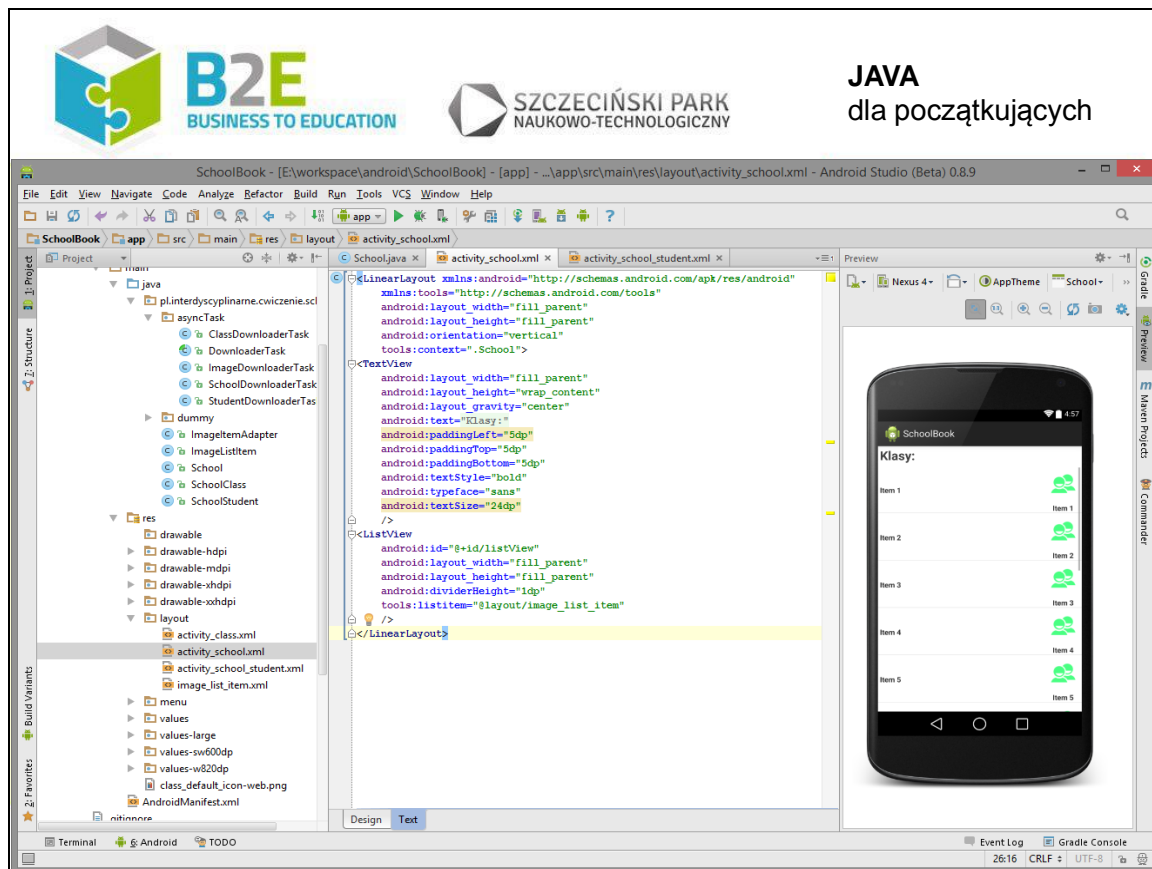
Activity jest plikiem xml'a definiującym strukturę widoku. W skład tej struktury mogą wchodzić różne komponenty, widgety, layouty i inne.

Do naszego widoku dodamy nagłówek oraz kontener ListView. Kontener ten służy do wyświetlania listy, czyli kolejnych wierszy z danymi tego samego typu. Niestety sposób reprezentacji danych może być różny w związku z czym to na nas spoczywa obowiązek zdefiniowania wyglądu wiersza listy. W następnym kroku stworzymy plik layoutu dla wiersza. Na slajdzie tym już w jednej z ostatnich linii wskazałem o który widok elementu listy mi chodzi:

```
tools:listitem="@layout/image_list_item"
```



Slajd 27



```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
```

```
    xmlns:tools="http://schemas.android.com/tools"
```

```
    android:layout_width="fill_parent"
```

```
    android:layout_height="fill_parent"
```

```
    android:orientation="vertical"
```

```
    tools:context=".School">
```

```
<TextView
```

```
    android:layout_width="fill_parent"
```

```
    android:layout_height="wrap_content"
```

```
    android:layout_gravity="center"
```

```
    android:text="@string/TitleSchool"
```

```
    android:paddingLeft="5dp"
```

```
    android:paddingTop="5dp"
```

```
    android:paddingBottom="5dp"
```

```
    android:textStyle="bold"
```

```
    android:typeface="sans"
```

```
    android:textSize="24dp"
```



```

/>
<ListView
    android:id="@+id/listView"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:dividerHeight="1dp"
    tools:listitem="@layout/image_list_item"
/>
</LinearLayout>

```

Slajd 28



Klikamy na folderze layoutu prawym klawiszem myszy i wybieramy „new->Layout resource file” w celu zdefiniowania layoutu wiersza.

<LinearLayout

```

    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:clickable="false">

```

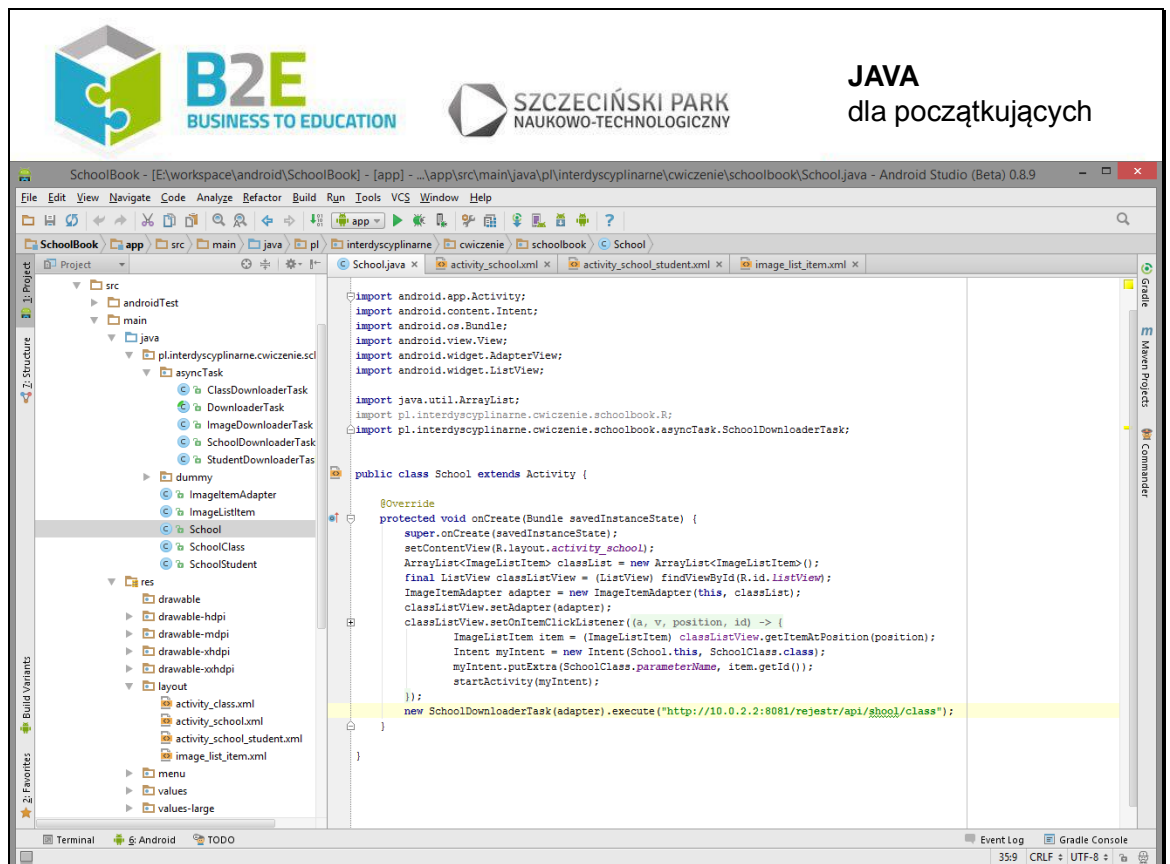
```
<RelativeLayout
    android:layout_width="fill_parent"
    android:layout_height="100dp">
<LinearLayout
    android:orientation="vertical"
    android:layout_width="wrap_content"
    android:layout_height="fill_parent"
    android:layout_alignParentLeft="true"
    >
<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_weight="1"
    android:gravity="center_vertical"
    android:text=""
    android:id="@+id/firstName"
    android:padding="5dp"
    android:textSize="15dp"
    android:textStyle="bold"
    android:typeface="sans" />
<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_weight="1"
    android:gravity="center_vertical"
    android:text=""
    android:id="@+id/lastName"
    android:padding="5dp"
    android:textSize="15dp"
    android:textStyle="bold"
    android:typeface="sans" />
</LinearLayout>
<ImageView
```



```
        android:id="@+id/studentPicture"
        android:src="@drawable/class_default_icon"
        android:layout_width="100dp"
        android:layout_height="100dp"
        android:layout_alignParentRight="true"/>
</RelativeLayout>
<RelativeLayout
    android:layout_width="fill_parent"
    android:layout_height="50dp">
    <TextView
        android:layout_width="wrap_content"
        android:layout_height="fill_parent"
        android:gravity="center_vertical"
        android:text=""
        android:id="@+id/email"
        android:padding="5dp"
        android:textSize="15dp"
        android:textStyle="bold"
        android:typeface="sans" />
    <ImageButton
        android:layout_width="100dp"
        android:layout_height="wrap_content"
        android:id="@+id/btnMail"
        android:layout_alignParentRight="true"
        android:src="@android:drawable/sym_action_email"
        android:background="@android:color/background_light"
        android:padding="5dp" />
</RelativeLayout>
<RelativeLayout
    android:layout_width="fill_parent"
    android:layout_height="50dp">
    <TextView
        android:layout_width="wrap_content"
        android:layout_height="fill_parent"
```

```
android:gravity="center_vertical"  
android:text=""  
android:id="@+id/phoneNumber"  
android:padding="5dp"  
android:textSize="15dp"  
android:textStyle="bold"  
android:typeface="sans" />  
<ImageButton  
android:layout_width="100dp"  
android:layout_height="wrap_content"  
android:id="@+id/btnCall"  
android:layout_alignParentRight="true"  
android:src="@android:drawable/ic_menu_call"  
android:background="@android:color/background_light"  
android:padding="5dp" />  
</RelativeLayout>  
</LinearLayout>
```

Slajd 29



W ten sposób zakończyliśmy pierwszy etap dodawania nowego widoku. Kolejnym krokiem jest dodanie zachowania/funkcjonalności do widoku. Programowanie jak można było przypuszczać odbywa się w języku Java. Będziemy chcieli wyświetlić listę klas, dlatego za chwilę stworzymy pomocniczą klasę, która jest w stanie przechowywać wszystkie niezbędne informacje dla jednego wiersza. Klasę nazwiemy `ImageListItem`, gdyż będzie zawierać również informacje o zdjęciu wychowawcy.

Obiekty tej klasy przetrzymywać będziemy w `ArrayList`.

Następnym elementem układanki jest funkcja `findViewById` umożliwiająca pobranie referencji do elementu widoku. W naszym przypadku odszukujemy `ListView`.

Mając model danych i widok czas połączyć je razem. Do tego celu stworzymy kolejną klasę `ImageItemAdapter`, zapełni nam ona widok wierszami pokazującymi dane z listy.

Ustawiamy adapter na naszym `ListView`.

Dodanie obsługi kliknięcia możemy na tym etapie pominąć.

Na samym dole jest jeszcze jedno ciekawe wywołanie. Ponieważ `activity` jest elementem GUI, a pobieranie listy klas z Internetu może długo potrwać, to z pewnością chcielibyśmy uniknąć efektu zamrożenia interfejsu użytkownika. W związku z czym pobieranie danych zlecimy do osobnego wątku.

Czas pokazać wymienione elementy szczegółowo.

```
package pl.interdyscyplinarne.cwiczenie.schoolbook;
import android.app.Activity;
import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import android.widget.AdapterView;
import android.widget.AdapterView.OnItemClickListener;
import android.widget.ListView;
import java.util.ArrayList;
import pl.interdyscyplinarne.cwiczenie.schoolbook.R;
import pl.interdyscyplinarne.cwiczenie.schoolbook.asyncTask.SchoolDownloaderTask;
public class School extends Activity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_school);
        ArrayList<ImageListItem> classList = new ArrayList<ImageListItem>();
        final ListView classListView = (ListView) findViewById(R.id.listView);
        ImageItemAdapter adapter = new ImageItemAdapter(this, classList);
        classListView.setAdapter(adapter);
    }
}
```





```

classListView.setOnItemClickListener(new AdapterView.OnItemClickListener() {
    @Override
    public void onItemClick(AdapterView<?> a, View v, int position, long id) {
        ImageListItem item = (ImageListItem) classListView.getItemAtPosition(position);
        Intent myIntent = new Intent(School.this, SchoolClass.class);
        myIntent.putExtra(SchoolClass.parameterName, item.getId());
        startActivity(myIntent);
    }
});
new SchoolDownloaderTask(adapter).execute();
}
}

```

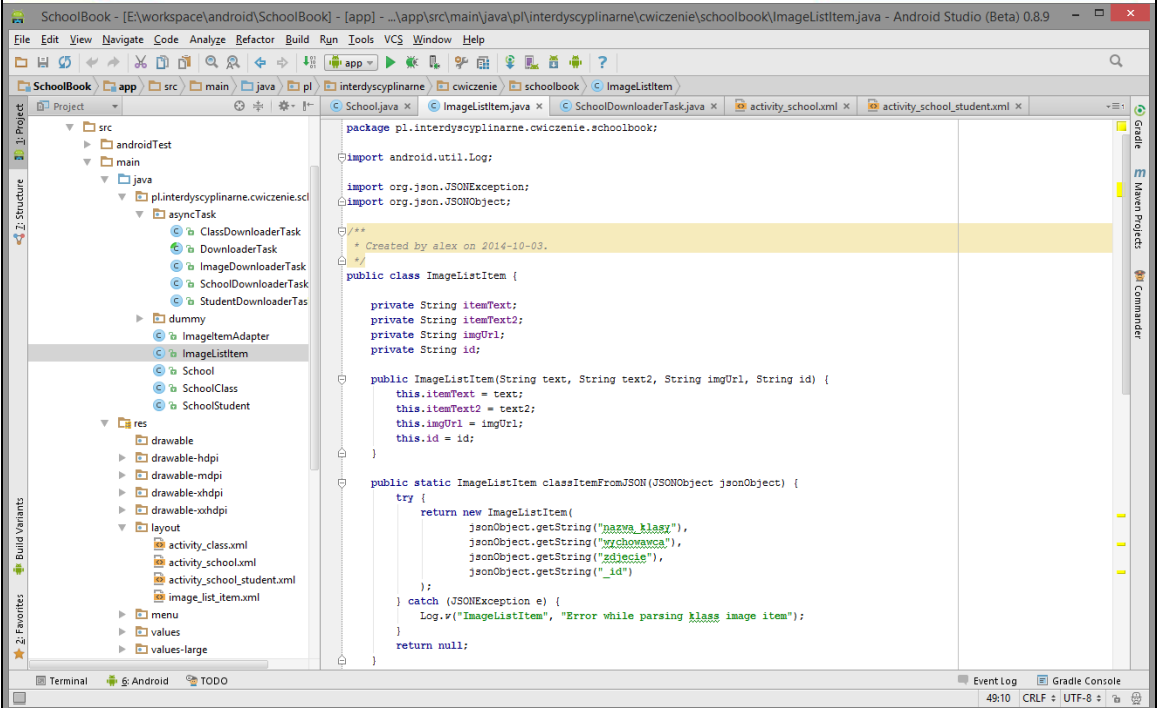
Slajd 30





JAVA

dla początkujących



Klasa ImageListItem jest dość prosta. Zawiera wszystkie niezbędne pola dla jednego wiersza widoku listy, dodatkowo gettery i settery. Oraz kod niezbędny do stworzenia instancji tej klasy z rezultatu wywołania zapytania do webserwisu REST.

REST - Representational State Transfer – jest jednym z wzorców sposobu komunikacji aplikacji poprzez protokół HTTP. Często do transportu danych wykorzystuje się format JSON. W ramach tej lekcji nie będziemy zbyt wnikliwie w szczegóły tego rodzaju komunikacji. Jednak trzeba wiedzieć, że w JSON'ie rozróżniamy obiekty, które mogą mieć pola o wartościach prymitywnych (liczba, ciąg

znaków), obiektowych (czyli zagnieżdżony obiekt JSON'a) lub typ tablicowy. Z kolei typ tablicowy może zawierać elementy typów: tablicowego, obiektowego czy prymitywnego.

Jak widać na slajdzie, w naszym przypadku z obiektu JSON'a odczytujemy cztery wartości typu String, a tak wyglądać może pojedynczy element listy w reprezentacji JSON:

```
{"_id":1,"nazwa_klasy":"1A","wychowawca":"Micha\u0142  
Kowalski","zdjecie":"http://localhost:8081/rejestr/upload/nauczyciel2.jpg"}
```

```
package pl.interdyscyplinarne.cwiczenie.schoolbook;
```

```
import android.util.Log;
```

```
import org.json.JSONException;
```

```
import org.json.JSONObject;
```

```
/**
```

```
 * Created by alex on 2014-10-03.
```

```
 */
```

```
public class ImageListItem {
```

```
    private String itemText;
```

```
    private String itemText2;
```

```
    private String imgUrl;
```

```
    private String id;
```

```
    public ImageListItem(String text, String text2, String imgUrl, String id) {
```

```
        this.itemText = text;
```

```
        this.itemText2 = text2;
```

```
        this.imgUrl = imgUrl;
```

```
        this.id = id;
```

```
    }
```

```
    public static ImageListItem classItemFromJSON(JSONObject jsonObject) {
```

```
        try {
```

```
            return new ImageListItem(
```

```
                jsonObject.getString("nazwa_klasy"),
```

```
                jsonObject.getString("wychowawca"),
```

```
                jsonObject.getString("zdjecie"),
```

```
                jsonObject.getString("_id")
```

```
            );
```

```
        } catch (JSONException e) {
```

```
            Log.w("ImageListItem", "Error while parsing class image item");
```

```
        }
```



```
        return null;
    }
    public static ImageListItem studentItemFromJSON(JSONObject jsonObject) {
        try {
            return new ImageListItem(
                jsonObject.getString("imie")+" "+jsonObject.getString("nazwisko"),
                "",
                jsonObject.getString("zdjecie"),
                jsonObject.getString("_id")
            );
        } catch (JSONException e) {
            Log.w("ImageListItem", "Error while parsing student image item");
        }
        return null;
    }
    public String getItemText() {
        return itemText;
    }
    public void setItemText(String itemText) {
        this.itemText = itemText;
    }
    public String getImgUrl() {
        return imgUrl;
    }
    public void setImgUrl(String imgUrl) {
        this.imgUrl = imgUrl;
    }
    public String getItemText2() {
        return itemText2;
    }
    public void setItemText2(String itemText2) {
        this.itemText2 = itemText2;
    }
    public String getId() {
```




```


return id;
}

public void setId(String id) {
    this.id = id;
}
}

```

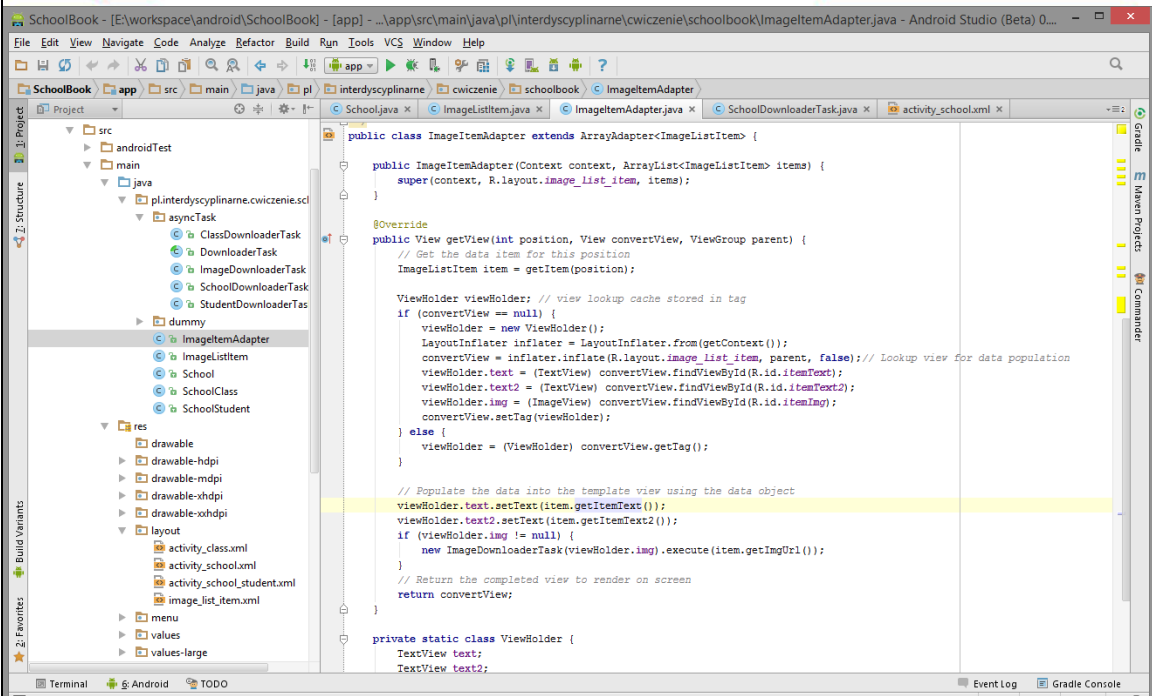
Slajd 31





JAVA

dla początkujących



```

public class ImageItemAdapter extends ArrayAdapter<ImageListItem> {
    public ImageItemAdapter(Context context, ArrayList<ImageListItem> items) {
        super(context, R.layout.image_list_item, items);
    }

    @Override
    public View getView(int position, View convertView, ViewGroup parent) {
        // Get the data item for this position
        ImageListItem item = getItem(position);

        ViewHolder viewHolder; // view lookup cache stored in tag
        if (convertView == null) {
            viewHolder = new ViewHolder();
            LayoutInflater inflater = LayoutInflater.from(getContext());
            convertView = inflater.inflate(R.layout.image_list_item, parent, false); // Lookup view for data population
            viewHolder.text = (TextView) convertView.findViewById(R.id.itemText);
            viewHolder.text2 = (TextView) convertView.findViewById(R.id.itemText2);
            viewHolder.img = (ImageView) convertView.findViewById(R.id.itemImg);
            convertView.setTag(viewHolder);
        } else {
            viewHolder = (ViewHolder) convertView.getTag();
        }

        // Populate the data into the template view using the data object
        viewHolder.text.setText(item.getItemText());
        viewHolder.text2.setText(item.getItemText2());
        if (viewHolder.img != null) {
            new ImageDownloaderTask(viewHolder.img).execute(item.getImgUrl());
        }
        // Return the completed view to render on screen
        return convertView;
    }

    private static class ViewHolder {
        TextView text;
        TextView text2;
    }
}

```

ListView chcąc wyrenderować kolejny element listy, odwoła się do naszej klasy ImageItemAdapter w celu uzyskania widoku. Nasz adapter rozszerza domyślny adapter dla list, w związku z czym musimy tylko zaimplementować jedną metodę tworzącą nasz specjalizowany widok. Najpierw w instrukcji if sprawdzamy, czy przypadkiem wcześniej już nie stworzyliśmy widoku dla danego wiersza. Jeśli nie, to przy pomocy metody LayoutInflater.inflate „dmuchamy” widok z naszego xml'a i wydobywamy z niego interesujące nas elementy przy użyciu findViewById. Jeśli jednak mieliśmy już stworzony wcześniej widok wiersza, to nie tworzymy nowego, tylko uaktualnimy danymi już istniejący.

Poniżej instrukcji if ustawiamy dwa teksty, a operację ściągnięcia obrazka znów zlecamy osobnemu



wątkowi.

```
package pl.interdyscyplinarne.cwiczenie.schoolbook;

import android.content.Context;
import android.graphics.Bitmap;
import android.graphics.BitmapFactory;
import android.os.AsyncTask;
import android.util.Log;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.AdapterView;
import android.widget.AdapterView.OnItemClickListener;
import android.widget.ImageView;
import android.widget.TextView;
import java.io.InputStream;
import java.util.ArrayList;
import pl.interdyscyplinarne.cwiczenie.schoolbook.ImageListItem;
import pl.interdyscyplinarne.cwiczenie.schoolbook.R;
import pl.interdyscyplinarne.cwiczenie.schoolbook.asyncTask.ImageDownloaderTask;
/**
 * Created by alex on 2014-10-03.
 */
public class ImageListAdapter extends ArrayAdapter<ImageListItem> {
    public ImageListAdapter(Context context, ArrayList<ImageListItem> items) {
        super(context, R.layout.image_list_item, items);
    }
    @Override
    public View getView(int position, View convertView, ViewGroup parent) {
        // Get the data item for this position
        ImageListItem item = getItem(position);
        ViewHolder viewHolder; // view lookup cache stored in tag
        if (convertView == null) {
            viewHolder = new ViewHolder();
            LayoutInflater inflater = LayoutInflater.from(getContext());
            convertView = inflater.inflate(R.layout.image_list_item, parent, false); // Lookup view for
```





data population

```
        viewHolder.text = (TextView) convertView.findViewById(R.id.itemText);
        viewHolder.text2 = (TextView) convertView.findViewById(R.id.itemText2);
        viewHolder.img = (ImageView) convertView.findViewById(R.id.itemImg);
        convertView.setTag(viewHolder);
    } else {
        viewHolder = (ViewHolder) convertView.getTag();
    }
    // Populate the data into the template view using the data object
    viewHolder.text.setText(item.getItemText());
    viewHolder.text2.setText(item.getItemText2());
    if (viewHolder.img != null) {
        new ImageDownloaderTask(viewHolder.img).execute(item.getImgUrl());
    }
    // Return the completed view to render on screen
    return convertView;
}

private static class ViewHolder {
    TextView text;
    TextView text2;
    ImageView img;
}
}
```

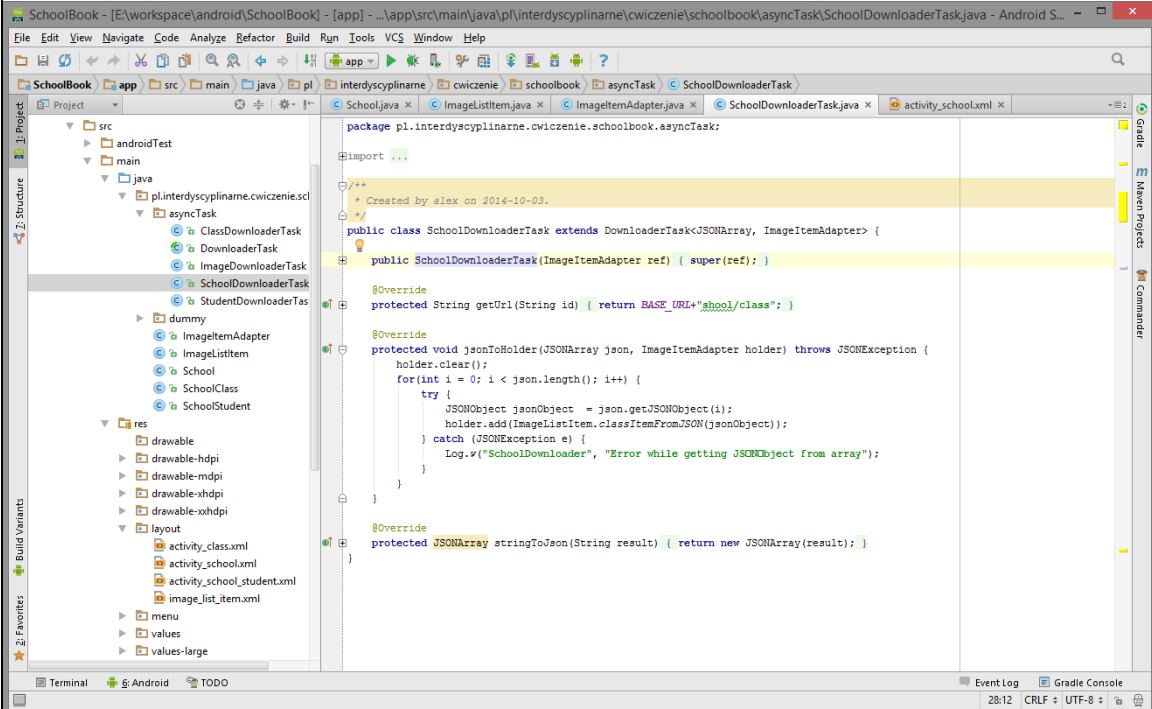
Slajd 32





JAVA

dla początkujących



```

package pl.interdyscyplinarne.cwiczenie.schoolbook.asyncTask;

import ...

/**
 * Created by alex on 2014-10-03.
 */
public class SchoolDownloaderTask extends DownloaderTask<JSONArray, ImageItemAdapter> {

    @Override
    protected String getUrl(String id) { return BASE_URL+"school/class"; }

    @Override
    protected void jsonToHolder(JSONArray json, ImageItemAdapter holder) throws JSONException {
        holder.clear();
        for(int i = 0; i < json.length(); i++) {
            try {
                JSONObject jsonObject = json.getJSONObject(i);
                holder.add(ImageListItem.classItemFromJSON(jsonObject));
            } catch (JSONException e) {
                Log.v("SchoolDownloader", "Error while getting JSONObject from array");
            }
        }
    }

    @Override
    protected JSONArray stringToJson(String result) { return new JSONArray(result); }
}
    
```

DownloaderTask – wykonuje całą pracę związaną z połączeniem z webserwisem oraz odebraniem odpowiedzi i zamienienia jej na obiektową reprezentację JSON'a, a później modelu danych. Wszystkie operacje wspólne zaimplementowane są właśnie w tej klasie, a specyficzne oddegutowane do klas podrzędnych przy użyciu metod abstrakcyjnych.

```

package pl.interdyscyplinarne.cwiczenie.schoolbook.asyncTask;

import android.net.http.AndroidHttpClient;

import android.os.AsyncTask;

import android.util.Log;

import org.apache.http.HttpEntity;
import org.apache.http.HttpResponse;

import org.apache.http.HttpStatus;

import org.apache.http.client.methods.HttpGet;

import org.json.JSONArray;

import org.json.JSONException;

import org.json.JSONObject;

import java.io.BufferedReader;

import java.io.InputStream;
    
```

```
import java.io.InputStreamReader;
import java.lang.ref.WeakReference;
import pl.interdyscyplinarne.cwiczenie.schoolbook.ImageItemAdapter;
import pl.interdyscyplinarne.cwiczenie.schoolbook.ImageListItem;
/**
 * Created by alex on 2014-10-07.
 */
public abstract class DownloaderTask<T, J> extends AsyncTask<String, Void, T> {
    protected final static String BASE_URL = "http://10.0.2.2:8081/rejestr/api/";
    protected final WeakReference<J> holderRef;
    public DownloaderTask(J ref) {
        holderRef = new WeakReference<J>(ref);
    }
    @Override
    // Actual download method, run in the task thread
    protected T doInBackground(String... params) {
        // params comes from the execute() call: params[0] is the url.
        return downloadJSON(params[0]);
    }
    @Override
    // Once the image is downloaded, associates it to the imageView
    protected void onPostExecute(T json) {
        if (isCancelled()) {
            json = null;
        }
        if (holderRef != null) {
            J holder = holderRef.get();
            if (holder != null) {
                if (json != null) {
                    try {
                        jsonToHolder(json, holder);
                    } catch (JSONException e) {
                        Log.w("DownloaderTask", "Error while parsing JSON");
                    }
                }
            }
        }
    }
}
```




```
    }
  }

}


private T downloadJSON(String id) {
    final AndroidHttpClient client = AndroidHttpClient.newInstance("Android");
    final HttpGet getRequest = new HttpGet(getUrl(id));
    try {
        HttpResponse response = client.execute(getRequest);
        final int statusCode = response.getStatusLine().getStatusCode();
        if (statusCode != HttpStatus.SC_OK) {
            Log.w("DownloaderTask", "Error " + statusCode + " while retrieving response from " +
                getUrl(id));
            return null;
        }
        final HttpEntity entity = response.getEntity();
        if (entity != null) {
            InputStream inputStream = null;
            try {
                inputStream = entity.getContent();
                BufferedReader bufferedReader = new BufferedReader(new
                InputStreamReader(inputStream));
                String line = "";
                String result = "";
                while ((line = bufferedReader.readLine()) != null) {
                    result += line;
                }
                return stringToJson(result);
            } finally {
                if (inputStream != null) {
                    inputStream.close();
                }
                entity.consumeContent();
            }
        }
    }
}
```




```
    }  
  }  
  } catch (Exception e) {  
    // Could provide a more explicit error message for IOException or  
    // IllegalStateException  
    getRequest.abort();  
    Log.w("DownloaderTask", "Error while retrieving result from " + getUrl(id));  
  } finally {  
    if (client != null) {  
      client.close();  
    }  
  }  
  return null;  
}  
  
protected abstract String getUrl(String id);  
protected abstract <T> T stringToJson(String result) throws JSONException;  
protected abstract void jsonToHolder(T json, J holder) throws JSONException;  
}
```



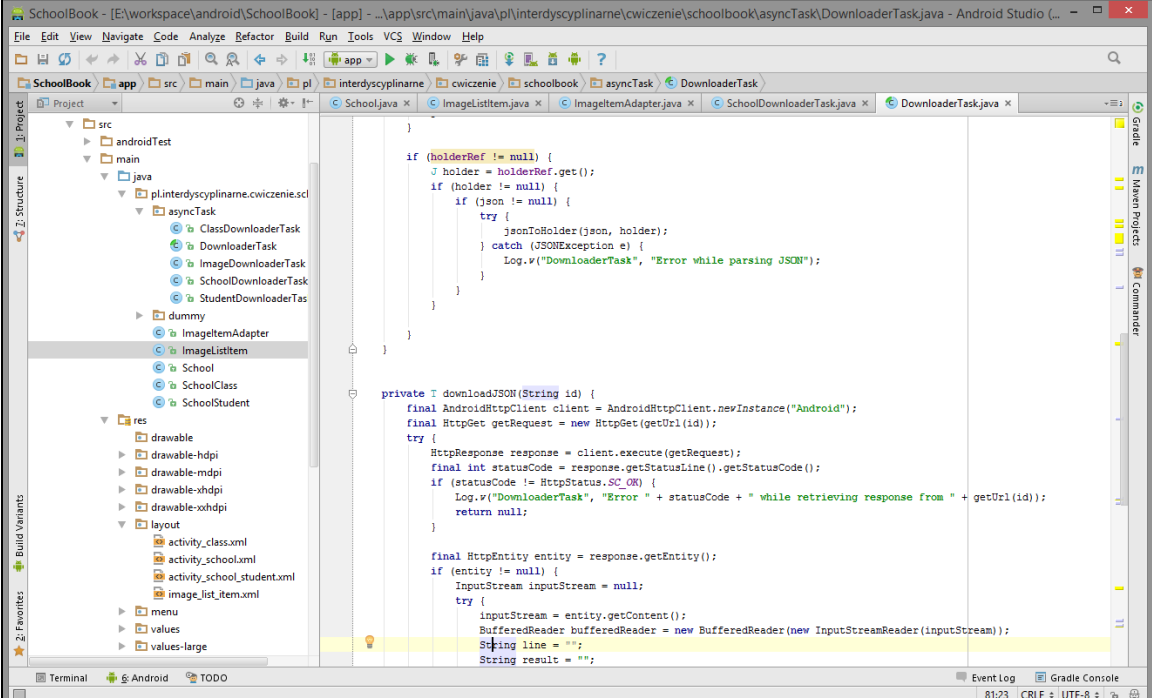
Slajd 33





JAVA


dla początkujących




Jeśli na komputerze developera jest uruchomiony serwer www z web serwisami, to mimo iż adresy do wywołania webserwisów zaczynają się od localhost, to w Androidzie trzeba wpisać inny adres widoczny na slajdzie. Spowodowane jest to tym, że aplikacja odpalana w emulatorze działa w maszynie wirtualnej, w związku z czym adres localhost odnosiłby się do adresu maszyny wirtualnej, a nie komputera developer'a. Adres podany w przykładzie jest standardowym dressem dla emulatorów Androida, jeśliby jednak nie zadziałał, to można podać rzeczywisty adres IP komputera developera.



Slajd 34

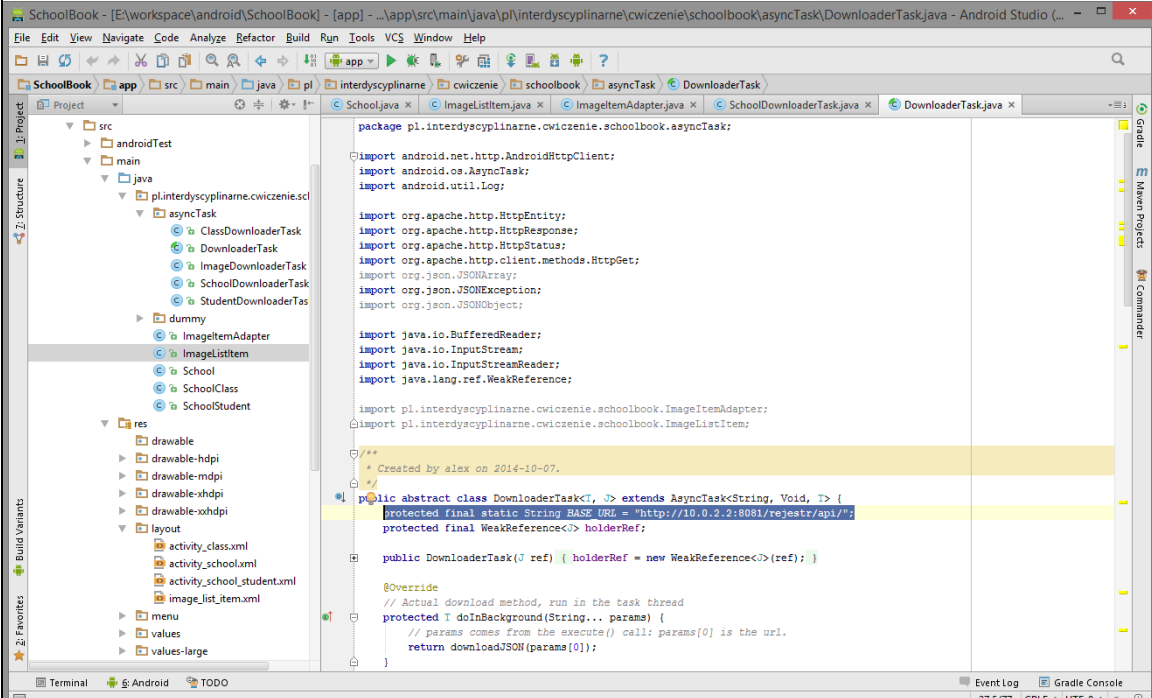


B2E
BUSINESS TO EDUCATION



SZCZECIŃSKI PARK
NAUKOWO-TECHNOLOGICZNY

JAVA
dla początkujących



```

package pl.interdyscyplinarne.cwiczenie.schoolbook.asyncTask;

import android.net.http.AndroidHttpClient;
import android.os.AsyncTask;
import android.util.Log;

import org.apache.http.HttpEntity;
import org.apache.http.HttpResponse;
import org.apache.http.HttpStatus;
import org.apache.http.client.methods.HttpGet;
import org.json.JSONArray;
import org.json.JSONException;
import org.json.JSONObject;

import java.io.BufferedReader;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.lang.ref.WeakReference;

import pl.interdyscyplinarne.cwiczenie.schoolbook.ImageItemAdapter;
import pl.interdyscyplinarne.cwiczenie.schoolbook.ImageListItem;


/**
 * Created by alex on 2014-10-07.
 */
public abstract class DownloaderTask<T, > extends AsyncTask<String, Void, T> {
    protected final static String BASE_URL = "http://10.0.2.2:8081/rejestr/api/";
    protected final WeakReference<> holderRef;

    public DownloaderTask(T ref) { holderRef = new WeakReference<>(ref); }


    @Override
    // Actual download method, run in the task thread
    protected T doInBackground(String... params) {
        // params comes from the execute() call: params[0] is the url.
        return downloadJSON(params[0]);
    }

```

Slajd 35



B2E
BUSINESS TO EDUCATION




SZCZECIŃSKI PARK
NAUKOWO-TECHNOLOGICZNY

JAVA
dla początkujących

Zadanie:


Wzorując się na implementacji widoku listy klas szkoły stwórz widok dla listy uczniów w klasie. W tym zadaniu nie odpytuj webserwisu, a listę możesz zappełnić ręcznie przykładowymi danymi.




Wstęp do języka i środowiska pracy



Slajd 36



B2E
BUSINESS TO EDUCATION



SZCZECIŃSKI PARK
NAUKOWO-TECHNOLOGICZNY


JAVA
dla początkujących

Zadanie:

Dodaj nawigację między widokiem listy klas, a widokiem uczniów w klasie za pomocą obsługi zdarzenia kliknięcie wiersza danej klasy. Uruchom aplikację w emulatorze i przetestuj działanie.


Podpowiedź: Kod obsługi zdarzenia kliknięcia:

```
class ListView.setOnItemClickListener(new AdapterView.OnItemClickListener() {
    @Override
    public void onItemClick(AdapterView<?> a, View v, int position, long id) {
        ImageListItem item = (ImageListItem) classListView.getItemAtPosition(position);
        Intent myIntent = new Intent(School.this, SchoolClass.class);
        myIntent.putExtra(SchoolClass.parameterName, item.getId());
        startActivity(myIntent);
    }
});
```




Wstęp do języka i środowiska pracy

Slajd 37



B2E
BUSINESS TO EDUCATION




SZCZECIŃSKI PARK
NAUKOWO-TECHNOLOGICZNY

JAVA
dla początkujących

Zadanie:

Wiedząc iż adres webserwisu do pobierania listy uczniów w klasie to `http://localhost/rejestr/api/shool/class/ + {id ucznia}`
Dodaj implementację zachowania widoku listy uczniów w klasie. Stwórz nową klasę `ClassDownloader` rozszerzając `DownloaderTask` i użyj jej w implementacji „activity”.



Podpowiedź: Sprawdź rezultaty zwracane przez webserwis wpisując adresy w przeglądarce internetowej:
`http://localhost/rejestr/api/shool/class`
`http://localhost/rejestr/api/shool/class/2`
`http://localhost/rejestr/api/person/1`



Wstęp do języka i środowiska pracy



Slajd 38


JAVA

dla początkujących

Zadanie:



Znając już adresy webserwisów z poprzedniego zadania stwórz „activity” dla ostatniego widoku wraz z pełną funkcjonalnością i komunikacją z webserwisem oraz nawigacją między widokami.

Dodaj przyciski służące do inicjowania rozmowy telefonicznej oraz wysyłania e-maila. Kod do obsługi interakcji z systemem widoczny jest na następnym slajdzie:



Wstęp do języka i środowiska pracy

Slajd 39


JAVA

dla początkujących

```

holder.btnemail.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        Intent emailIntent = new Intent(Intent.ACTION_SENDTO, Uri.fromParts("mailto",email, null));
        emailIntent.putExtra(Intent.EXTRA_SUBJECT, "Hello "+name);
        holder.caller.startActivity(Intent.createChooser(emailIntent, "Send email..."));
    }
});

holder.btnCall.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        Intent callIntent = new Intent(Intent.ACTION_CALL);
        callIntent.setData(Uri.parse("tel:" + tel));
        holder.caller.startActivity(callIntent);
    }
});
    
```



Wstęp do języka i środowiska pracy

Opis założonych osiągnięć ucznia

Uczeń nauczy się obsługiwać nowe IDE i tworzyć proste aplikacje mobilne. Nawiązywanie połączenia z zewnętrznym serwerem oraz komunikacja za pomocą formatu JSON powinny być znane uczniowi.

Moduł szkoleniowy JavaScript

Wstęp

JavaScript jest nazywany "językiem internetu" – to jedyny język rozumiany przez przeglądarki internetowe. Wszystkie nowoczesne serwisy internetowe działają w oparciu, lub z wykorzystaniem języka JavaScript.

JavaScript pozwala na dodawanie do stron internetowych elementów takich jak animowane menu, pokazy slajdów, galerie zdjęć itp. Dzięki wykorzystaniu tego języka strony internetowe mogą reagować na działania użytkowników. JavaScript jest także podstawą technologii AJAX – umożliwiającą interakcję z innymi systemami bez konieczności przeładowywania strony w przeglądarce. Dzięki temu witryny korzystające z tej technologii zaczynają działać jak aplikacje uruchamiane bezpośrednio w systemie operacyjnym.

W ostatnich latach język JavaScript wyrósł na podstawowy język wykorzystywany w nowoczesnych rozwiązaniach mobilnych. Jego wieloplatformowość pozwala na stworzenie jednej podstawowej wersji aplikacji i uruchamianie jej na różnych urządzeniach (np. telefonach z systemem Android, iOS itp.)

Dla osoby pragnącej tworzyć nowoczesne aplikacje internetowe, oprócz znajomości języków HTML i CSS niezbędna jest także umiejętność korzystania z JavaScript.

W poszczególnych częściach kursu szczegółowo zostaną opisane podstawowe elementy składni języka JavaScript. Zapoznanie się z ich treścią pozwoli na swobodną pracę z tym językiem przy tworzeniu nowoczesnych witryn www.

Cele

Celem kursu jest poznanie na poziomie podstawowym języka programowania obiektowego jakim jest JavaScript, jego składni, podstawowych poleceń i struktur danych. Dodatkowo w trakcie kursu uczestnik pozna kilka praktycznych zastosowań języka przy tworzeniu nowoczesnych, interaktywnych witryn WWW. Zostanie przedstawiona również biblioteka jQuery jako jedno z najpopularniejszych obecnie rozszerzeń języka.

Opis sposobu realizacji celów

10 półtoragodzinnych lekcji składających się z przypomnienia wiedzy z poprzedniej lekcji, przedstawienia materiału wraz z przykładami, ćwiczeń praktycznych. Po zakończeniu całego cyklu - przeprowadzenie egzaminu sprawdzającego wiedzę.

Dodatkowo wprowadzono lekcję 11 zawierającą część projektu interdyscyplinarnego łączącego 4 dziedziny wiedzy z zakresu informatyki (SQL, PHP, JavaScript oraz JAVA) w zakresie wzbogacenia interfejsu użytkownika oraz automatyczną walidację poprawności danych.

Treści kształcenia

Treść kursu została podzielona na 10 bloków tematycznych – po jednym do każdej lekcji:

1. Wprowadzenie do kursu, podłączenie skryptów do strony www, podstawowe polecenia
2. Zmienne, typy danych i operatory

3. Instrukcje warunkowe i pętle
4. Tablice
5. Obiekty i funkcje
6. Model dokumentu – Document Object Model
7. Zdarzenia
8. Obsługa błędów
9. Wykorzystanie JavaScript
10. Framework jQuery
11. Projekt interdyscyplinarny – baza teleadresowa kolegów i koleżanek ze szkoły.

Opis założonych osiągnięć ucznia

Uczestnik po odbyciu kursu pozna podstawy języka JavaScript, pozna składnię i typy danych tego języka. Nauczy się wykorzystywać JavaScript w tworzeniu witryn WWW.

Korelacja z podstawą programową

Programowanie w języku JavaScript		
Cele kształcenia	Treści kształcenia	Podstawa programowa
- uczeń poznaje podstawowe typy danych języka JavaScript, oraz podstawowe instrukcje	Lekcja 2, 4	Uczeń korzysta z wbudowanych typów danych. Uczeń stosuje instrukcje, funkcje, procedury (...) wybranych języków programowania.
- uczeń zna podstawowe instrukcje języka, - uczeń tworzy własne funkcje i obiekty	Lekcja 3, 5	Uczeń tworzy własne typy danych. Uczeń stosuje instrukcje, funkcje, procedury (...). Uczeń tworzy własne funkcje, procedury, obiekty (...)
- uczeń poznaje model DOM jako reprezentację struktury strony w przeglądarce, poznaje jego podstawowe obiekty i uczy się je wykorzystywać - uczeń poznaje zdarzenia dostępne w trakcie przeglądania strony - uczeń potrafi obsługiwać zdarzenia w przeglądarce	Lekcja 6, 7	Uczeń stosuje skrypty wykonywane po stronie klienta przy tworzeniu aplikacji internetowych
- uczeń poznaje sposoby obsługi błędów w tworzonych skryptach	Lekcja 8	Uczeń testuje tworzoną aplikację i modyfikuje jej kod źródłowy, Uczeń przestrzega zasad programowania
- uczeń poznaje praktyczne zastosowania języka JavaScript - uczeń potrafi wykorzystywać JavaScript w programowaniu	Lekcja 9	Uczeń stosuje skrypty wykonywane po stronie klienta przy tworzeniu aplikacji internetowych

witryn www		
- uczeń poznaje podstawy biblioteki funkcji jQuery, - uczeń potrafi wykorzystać funkcje dostarczane przez bibliotekę w tworzeniu stron www	Lekcja 10	Uczeń wykorzystuje frameworki do tworzenia własnych aplikacji, Uczeń przestrzega zasad programowania

Sposoby osiągnięcia celów

Po odbyciu kursu uczeń powinien wykonać projekt/zadanie, które zmotywuje go do pracy indywidualnej ze środowiskiem programistycznym i bazą danych. Zadanie utrwali zdobytą na kursie wiedzę i zmusi do wykorzystania zdobytej wiedzy teoretycznej w praktyce. Zadanie powinno być sformułowane w sposób otwarty, tak aby każdy z uczniów mógł wybrać coś dla niego interesującego. Powinna też zostać dostarczona lista możliwych tematów projektu do wyboru - dla osób, które nie będą miały pomysłu na projekt.

Projekt zaliczeniowy

1. Utwórz skrypty JS które przygotują animowane menu na witrynie www.

Propozycje kryteriów oceny i metod sprawdzania osiągnięć ucznia

Oceną zaliczającą kurs będzie wynik testu sprawdzającego wiedzę z zakresu kursu, ocena pracy ucznia w trakcie semestru oraz przygotowanie prostego projektu (dla chętnych)

Ocenę **dopuszczającą (2)** otrzymuje uczeń, który potrafi podłączyć i uruchomić skrypt na stronie internetowej, zna podstawowe pojęcia związane z językiem JavaScript, potrafi własnymi słowami wyjaśnić podstawowe instrukcje języka.

Ocenę **dostateczną (3)** otrzymuje uczeń, który spełnia wymagania oceny dopuszczającej oraz potrafi stworzyć własne skrypty pozwalające na operowanie elementami strony internetowej. Zna podstawowe struktury danych w języku JavaScript (zmienne i tablice). Wykonuje ćwiczenia z niewielką pomocą nauczyciela. Potrafi wskazać podstawowe zastosowania języka JavaScript w tworzeniu stron internetowych

Ocenę **dobrą (4)** otrzymuje uczeń, który spełnia wymagania na ocenę dostateczną oraz potrafi stworzyć skrypt i podłączyć skrypty reagujące na zdarzenia na stronie internetowej, zna i potrafi wykorzystać obiektowy model programowania w języku JavaScript. Na lekcjach wykonuje samodzielnie zadane ćwiczenia. Umiejętnie korzysta z mechanizmów pozwalających na modyfikację witryny internetowej, jej elementów i stylu.

Ocenę **bardzo dobrą (5)** otrzymuje uczeń, który spełnia wymagania na ocenę dobrą oraz bardzo dobrze zna teorię omawianą na zajęciach, w trakcie wykonywania ćwiczeń wykazuje inicjatywę, potrafi wskazać kilka rozwiązań zadania, tworzy skrypty języka z wykorzystaniem dobrych praktyk programistycznych. Wykorzystuje funkcje dostępne w bibliotekach jQuery.

Ocenę **celującą (6)** otrzymuje uczeń, którego wiedza wykracza poza omawiany zakres. Np. potrafi stworzyć aplikację/grę w języku JavaScript na platformy mobilne (www, lub smartfony). Tworzy zaawansowane interaktywne witryny www, udziela się na forach i grupach dyskusyjnych poświęconych językowi JavaScript.

Kryteria oceny testu:

- 0-7 poprawnych odpowiedzi - 1
- 8-9 poprawnych odpowiedzi - 2
- 10-11 poprawnych odpowiedzi - 3
- 12-13 poprawnych odpowiedzi - 4
- 14-15 poprawnych odpowiedzi - 5

Test końcowy sprawdzający wiedzę

Krótki (ok 20 minut) test składający się z 15 pytań

Przykładowy test (pogrubioną czcionką zaznaczono poprawne odpowiedzi)

1. Wewnątrz jakiego elementu należy umieścić treść skryptu
 - a. `<js>`
 - b. `<script>`**
 - c. `<javascript>`
 - d. `<scripting>`
2. Jaka jest prawidłowa składnia polecenia wypisującego tekst „Witaj świecie!”
 - a. `("Witaj świecie!");`
 - b. `"Witaj świecie!";`
 - c. `response.write("Witaj świecie!");`
 - d. `document.write("Witaj świecie!");`**
3. Jaka jest poprawna składnia polecenia ładującego zewnętrzny plik skrypt.js?
 - a. `<script type="text/javascript" name="skrypt.js">`
 - b. `include("skrypt.js");`
 - c. `<script type="text/javascript" src="skrypt.js">`**
 - d. `<script type="text/javascript" href="skrypt.js">`
4. W jaki sposób tworzymy funkcję w JavaScript
 - a. `function = mojaFunkcja()`
 - b. `function:mojaFunkcja()`
 - c. `function mojaFunkcja()`**
5. Jaki jest poprawny sposób utworzenia tablicy JavaScript?
 - a. `var tab = new Array("Jan", "Adam", "Jacek");`**
 - b. `var tab = new Array(1: "Jan", 2:"Adam", 3:"Jacek");`
 - c. `var tab = new Array:1=("Jan")2=("Adam")3=("Jacek");`
 - d. `var tab = new Array = "Jan", "Adam", "Jacek"`
6. W jaki sposób można wywołać funkcję o nazwie mojaFunkcja?
 - a. `exec mojaFunkcja()`
 - b. `mojaFunkcja()`**
 - c. `call mojaFunkcja()`
 - d. `call function mojaFunkcja()`
7. W jaki sposób zapisać instrukcję warunkową jeśli i jest równe 5 to wykonaj instrukcje
 - a. `if i=5 then instrukcje;`
 - b. `check(i = 5) instrukcje`
 - c. `if (i == 5) { instrukcje }`**
 - d. `if (i == 5) then { instrukcje }`



8. Jak definiujemy pętlę While
 - a. **while (i <= 10)**
 - b. while i= 1 to 10
 - c. while(i <= 10; i++)
9. Warunek pętli do ... while jest sprawdzany
 - a. przed każdą iteracją pętli
 - b. **po każdej iteracji pętli**
 - c. raz – na końcu pętli
 - d. raz – na początku pętli
10. Jakie zdarzenie sprawdza czy pole formularza zmieniło wartość
 - a. onsubmit
 - b. onblur
 - c. **onchange**
 - d. onclick
11. Ile parametrów można przekazać do funkcji
 - a. żadnego
 - b. tyle ile chcesz
 - c. **jeden na każdy argument funkcji**
 - d. jeden
12. Kiedy zdarzenie JavaScript nie zostanie uruchomione
 - a. kiedy inne zdarzenie jest jeszcze obsługiwane
 - b. **kiedy JavaScript jest wyłączony w przeglądarce**
 - c. kiedy strona używa arkusza stylu CSS
 - d. kiedy strona uruchamiana jest lokalnie a nie na serwerze
13. Tablica w JavaScript jest:
 - a. zmienną
 - b. **obiektem**
 - c. metodą
 - d. funkcją
14. W jaki sposób można zmienić tekst zawierający przecinki na tablicę
 - a. tablica = tekst.indexOf(",")
 - b. **tablica = tekst.split(",");**
 - c. tablica = tekst.trim(",");
 - d. tablica = tekst.substring(",");
15. W którym obiekcie DOM dostępna jest treść strony
 - a. window
 - b. browser
 - c. **document**
 - d. location

Lekcje

Lekcja 1 Wprowadzenie

Cel lekcji

Celem lekcji jest wprowadzenie do tematyki związanej z programowaniem interaktywnych stron internetowych. Uczestnik pozna podstawowe informacje o tworzeniu stron WWW z wykorzystaniem JavaScript, historię tego języka oraz sposoby umieszczania skryptów w plikach HTML.

Sposoby osiągnięcia celów kształcenia

Czynności nauczyciela	Czynności ucznia
wprowadza nowe pojęcia związane z tematem zajęć (materiał ze slajdów) przygotowuje ćwiczenia z zaprezentowanego materiału	Wykonuje zadane przez nauczyciela ćwiczenia na komputerze
Prowadzi z uczniami dyskusję na temat wybranych przez nich rozwiązań ćwiczeń	Prezentuje rozwiązania ćwiczeń nauczycielowi i pozostałym uczniom, tłumaczy wybrane przez siebie rozwiązanie
Podsumowuje lekcję, ocenia pracę uczniów na zajęciach.	Dokonuje samooceny swojej pracy, omawia z nauczycielem rozwiązania ćwiczeń

Treść - slajdy z opisem

Slajd 1

W kursie przedstawimy podstawy języka JavaScript, jego podstawowe polecenia, typy danych. Zaprezentujemy sposoby jego wykorzystania w witrynach WWW. Przedstawimy bibliotekę jQuery jako obecnie najpopularniejszą i wykorzystywaną na większości stron WWW. Zaczniemy od informacji co to jest JavaScript, w jaki sposób można go podłączyć do strony www i do czego można go wykorzystać. W kolejnych lekcjach wprowadzimy podstawowe pojęcia i instrukcje języka JavaScript, pokażemy model obiektowy języka, model dokumentu html i możliwości reagowania na działania



użytkownika.

Slajd 2

JavaScript



- język skryptowy
- nie wymaga kompilowania przed uruchomieniem
- wykonywany przez program zwany interpreterem/parserem
- parser jest wbudowany w większość przeglądarek

DAILY GROUP

JavaScript to język tekstowy który nie wymaga żadnej konwersji przed uruchomieniem. Inne języki takie jak Java i C++ muszą zostać skompilowane zanim będzie można je uruchomić, natomiast JavaScript jest uruchamiany od razu przez program zwany parserem lub interpreterem. Praktycznie każda nowoczesna przeglądarka zawiera w sobie parser języka JavaScript.

JavaScript nie jest używany jako samodzielny język, został bowiem zaprojektowany do łatwego osadzania w innych produktach i aplikacjach, jak na przykład przeglądarkach internetowych. Wewnątrz swojego bazowego środowiska, JavaScript może być połączony z obiektami otoczenia, w którym się znajduje, aby zapewnić nad nim programową kontrolę. Wykorzystywany w witrynach www do zapewnienia interakcji z użytkownikiem.

Slajd 3



JavaScript

Krótką historia

- Stworzony przez firmę Netscape w 1995 roku.
- Głównym twórcą jest Brandon Eich
- Początkowo nazywany "Mocha", później "LiveScript"
- Po podpisaniu umowy z SUN Microsystem wprowadzono nazwę JavaScript
- Organizacja ECMA rozpoczęła pracę nad standardem języka w 1996 roku, standard został nazwany ECMAScript
- Obecna wersja to 5.1 z czerwca 2011



W roku 1995 Netscape Navigator był dominującą przeglądarką na rynku i firma zdecydowała dodać interaktywność do stron HTML dzięki lekkim językowi programowania. Prace nad językiem zostały zlecone Brandanowi Eichowi, który napisał pierwszą wersję języka w 10 dni. Wstępna nazwa "Mocha" została zmieniona na "LiveScript" a w wyniku porozumienia z firmą Sun Microsystems na "JavaScript".

Język ten miał służyć jako klej łączący aplety wykonane w technologii Java, dodatkowo chciano wykorzystać szum medialny związany z Javą.

Pod taką nazwą został opublikowany w przeglądarce Netscape Navigator 2.0B3.

W 1996 roku rozpoczęły się prace nad opracowaniem standardu przez organizację standaryzującą ECMA International. Z racji tego, że Sun był właścicielem marki Java nowy standard nie mógł zostać nazwany JavaScript. W wyniku tego nazwa standardu to ECMAScript a jego implementacje to JavaScript, JScript (Microsoft) itd.

Obecna wersja ECMAScript to 5.1 – jest ona implementowana przez większość nowych przeglądarek (Chrome w wersji 19+, Mozilla Firefox 4+, Internet Explorer 9+)



Slajd 4

B2E
BUSINESS TO EDUCATION

SZCZECIŃSKI PARK
NAUKOWO-TECHNOLOGICZNY

JavaScript

Dodawanie JS do strony www

- `<script type="text/javascript" language="JavaScript 1.5">`
- `alert(12 > 6);`
- `</script>`

DAILY GROUP

W języku HTML za umieszczanie skryptów JS odpowiedzialny jest element `<script>` z argumentem `type text/javascript`, `application/javascript` oraz argumentem `language` o wartości `javascript`. Atrybut `language` jest jednak przestarzały i zaleca się go pomijać. Jest on pozostałością z czasów, kiedy istniał jeszcze język skryptowy VBScript żeby odróżnić skrypty napisane w tych językach

Slajd 5

B2E
BUSINESS TO EDUCATION

SZCZECIŃSKI PARK
NAUKOWO-TECHNOLOGICZNY

JavaScript

Zgodność ze starszymi przeglądarkami

```
<script type="text/javascript" language="JavaScript 1.5">
<!--
alert(12 > 6);
//-->
</script>
```

DAILY GROUP

Przeglądarki nie obsługujące skryptów nie potrafią zinterpretować znacznika `<script>`. Podstawowym działaniem przeglądarki jest ignorowanie znaczników, których nie rozumieją. Takie działanie jest poprawne, gdy znacznik jest pojedynczym wyrażeniem, ale w znaczniku `<script></script>` można umieścić wiele wyrażeń.

Starsze przeglądarki nie wiedzą, że mają się spodziewać znacznika zamykającego `</script>` więc po prostu wygenerują wszystkie linie występujące po otwarciu znacznika `<script>`.


Aby rozwiązać ten problem można otoczyć wyrażenia javascript znacznikami komentarza html `<!-- -->`.

linia zamykająca komentarz html rozpoczyna się znakiem komentarza JS (`//`), który oznacza polecenie zignorowania tej linii przez interpreter JavaScript, ale przeglądarki nie wspierające skryptów zinterpretują znacznik końca komentarza i rozpoczną generowanie


strony od kolejnego znacznika.

Rozwiązanie to zapewniało zgodność ze starszymi przeglądarkami. Obecnie można ten mechanizm pominąć, ale przytoczymy go tutaj dla porządku

Slajd 6



B2E
BUSINESS TO EDUCATION



SZCZECIŃSKI PARK
NAUKOWO-TECHNOLOGICZNY

JavaScript

Zgodność z XHTML Strict

```

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0
Strict//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-
strict.dtd">
<script type="text/javascript">
/*  */
var x = 3;
alert('Witajcie! - x to ' + x);
/* ]]&gt; */
&lt;/script&gt;
                </pre>
</div>
<div style="text-align: center; margin-top: 10px;">
<img alt="DAILY GROUP logo" data-bbox="172 516 259 536"/>
</div>
</div>
</div>
<div data-bbox="135 537 830 585" data-label="Text">
<p>Jeśli jako typ dokumentu zdefiniujemy Strict XHTML (&lt;!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd"&gt;) musimy otoczyć skrypt znacznikiem CDATA.</p>
</div>
<div data-bbox="135 585 830 633" data-label="Text">
<p>Parseery XHTML nie pozwalają na symbole '&lt;' i '&gt;', które nie są elementami xml, więc treść skryptu należy umieścić w znaczniku specjalnym [CDATA], który nakazuje interpreterowi XML traktować wyrażenia wewnątrz znacznika jako treść.</p>
</div>
<div data-bbox="135 633 830 681" data-label="Text">
<p>W XHTML nie można używać argumentu language w znaczeniu określenia wersji języka JS (atrybut, jeżeli jest użyty, powinien przyjąć dwuznakowe wartości opisane standardem ISO 639, np. EN, DE, PL)</p>
</div>
<div data-bbox="67 947 103 964" data-label="Page-Footer">
<p>232</p>
</div>
<div data-bbox="641 947 837 965" data-label="Page-Footer">
<p>Przewodnik Metodyczny</p>
</div>
```




Slajd 7

Podłączenie zewnętrznego pliku

Zalecany sposób:

```
<script type="text/javascript" src="plik.js"></script>
```

Zaleca się jednak przechowywać skrypty w osobnych plikach i podłączać je do stron za pomocą atrybutu src

```
<script type="text/javascript" src="plik.js"></script>
```

Plik "plik.js" zawiera tylko definicję skryptów (bez dodatkowych znaczników <script>). Ogólnie przyjętą praktyką jest aby plik miał rozszerzenie ".js", chociaż nie jest to wymagane.

Zaletą takiego podejścia jest możliwość ponownego użycia skryptu na innej stronie.

Jeśli skrypt będzie zagnieżdżony w znaczniku HTML, każda strona będzie zawierała nadmiarową treść i modyfikacja skryptu będzie wymagała takich samych zmian w każdym pliku.

Z kolei podłączenie zewnętrznego skryptu to tylko jedna dodatkowa linia w każdym pliku html i zewnętrzny skrypt może być zmodyfikowany raz, aby zmiany były widoczne na wszystkich stronach, które go wykorzystują.

Plik ze skryptem zostanie umieszczony w pamięci podręcznej przeglądarki (cache) i wykorzystywany przy przechodzeniu między witrynami na stronie.

Dodatkową zaletą jest brak konieczności stosowania komentarzy i znaczników CDATA.

Slajd 8

Znacznik noscript

- `<noscript>`
- `<p>`
- Twoja przeglądarka nie obsługuje JavaScriptu. Aby zobaczyć stronę w pełnej funkcjonalności, zainstaluj inną przeglądarkę lub włącz opcję wyświetlania JS w przeglądarce
- `</p>`
- `</noscript>`

Dodatkowym znacznikiem który możemy wykorzystać jest znacznik `<noscript></noscript>`. W przypadku nowoczesnych przeglądarek z wyłączoną obsługą JavaScript zostanie wyświetlona treść znajdująca się między znacznikami `<noscript>`. Możemy umieścić tam informację o konieczności włączenia obsługi skryptów, lub prośbę o wyświetlenie naszej strony w innej przeglądarce. Starsze przeglądarki również wyświetlą taką informację (wynika to z wcześniej wspomnianego mechanizmu działania przeglądarek - nieznanym znacznik jest pomijany, a zawartość jest traktowana jako treść do wygenerowania i wysyłana do przeglądarki).

Slajd 9

Gdzie umieszczają skrypty?

- w nagłówku strony

```

<!DOCTYPE html...>
<html>
  <head>
    <title>Tytuł</title>
    <script src="plik.js"></script>
  </head>
  <body>
    <!-- treść strony -->
  </body>
</html>

```

Technicznie można umieścić znacznik `<script>` w dowolnym miejscu na stronie. Może to być sekcja `<head>` lub sekcja `<body>`. Skrypty znajdujące się w nagłówku strony zostaną wykonane jeszcze przed załadowaniem właściwej treści strony, natomiast w przypadku umieszczenia skryptu w ciele strony przed uruchomieniem skryptu zostanie wygenerowana część strony znajdująca się przed znacznikiem `<script>`



Klasyczne najlepsze praktyki wskazują umieszczenie skryptów w nagłówku strony. Zaletą takiego podejścia jest jak wspomnieliśmy wczytanie całego skryptu przed załadowaniem strony. Dodatkowo łatwiej jest innym programistom znaleźć znacznik script, co ułatwia debugowanie strony.

Natomiast wady takiego rozwiązania są następujące:

Ładowanie treści jest wstrzymane aż do momentu aż załadują i wykonają się wszystkie skrypty,

skrypty nie mają dostępu do znaczników HTML w dokumencie, ponieważ nie został on jeszcze wczytany. Trzeba opóźnić wywołanie skryptów modyfikujących stronę, aż do momentu jej pełnego załadowania (można to zrobić za pomocą zdarzeń, które omówimy w późniejszych lekcjach).

Slajd
10

The slide is titled "Gdzie umieszczać skrypty?" (Where to place scripts?). It features logos for B2E (Business to Education), Szczeciński Park Naukowo-Technologiczny, and JavaScript. The main content is a list item: "- na końcu sekcji <body>". Below this, it shows a snippet of HTML code:


```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Tytuł</title>
  </head>
  <body>
    <!-- treść strony -->
    <script src="myscripts.js"></script>
  </body>
</html>
```

 The code shows the script tag placed at the end of the body section. At the bottom of the slide, there is a logo for "DAILY GROUP".

Niektórzy specjaliści od wydajności zalecają umieszczenie skryptów na końcu sekcji <body>. Ładowanie strony nie jest przerywane przez wczytywanie skryptów, w skrypcie nie trzeba czekać na zakończenie ładowania strony ponieważ cała treść jest już dostępna.

Wadą są takie same :) strona jest dostępna dla użytkownika jeszcze zanim zostaną wgrane pliki JavaScript. Oznacza to, że jeśli np. na naszej witrynie zastosowaliśmy formularz, którego poprawne wypełnienie sprawdzamy za pomocą skryptu, to użytkownik może wysłać taki formularz zanim skrypt sprawdzający będzie dostępny.

Aby rozwiązać taki problem możemy podzielić nasze skrypty na dwie grupy – jedną zawierającą ważne funkcje dołączyć do strony w nagłówku, a drugą z mniej ważnymi funkcjami w stopce strony.

Slajd
11

JavaScript

Bezpieczeństwo JavaScript

- JavaScript nie zawiera żadnych mechanizmów bezpieczeństwa.
- Każdy skrypt ma takie same prawa – każdy może uzyskać dostęp do innego skryptu i zmodyfikować jego zmienne i funkcje.
- Skrypt może podpisać nawet natywne funkcje JavaScriptu.

JavaScript nie zawiera żadnych mechanizmów bezpieczeństwa. Każdy skrypt ma takie same prawa – każdy może uzyskać dostęp do innego skryptu i zmodyfikować jego zmienne i funkcje. Skrypty mogą odczytywać pliki cookies, i wykorzystując prototyp funkcji można podpisać zawartość każdej natywnej funkcji JavaScript. Dodatkowo JavaScript można wyłączyć, więc nie należy blokować dostępu do strony za pomocą mechanizmów JavaScript.

Slajd
12

JavaScript

Dlaczego używać JavaScriptu?

- uruchamiany po stronie klienta
- ogranicza czas wykonywania strony (strona nie jest wysyłana na serwer i odsyłana do klienta po przetworzeniu)

W czasach gdy ceny hostingu były stosunkowo wysokie, a prędkości połączenia z internetem niskie możliwość wykonywania kodu po stronie klienta była nieoceniona. Można było reagować na działania użytkowników bez konieczności przesyłania strony do serwera. Strony z dobrze wykorzystanymi możliwościami JavaScript były bardziej interaktywne i szybsze. Obecnie prędkości i ceny są inne, ale brak konieczności odsyłania stron na serwer nadal powoduje że interakcja z nimi jest dużo szybsza i płynniejsza.



Slajd
13

Do czego można wykorzystać JS

- JavaScript jest prosty w implementacji
- działa na komputerze użytkownika
- dodaje dynamiczną zawartość na stronie (menu itp.)
- może załadować treść do strony bez przesyłania jej na serwer (AJAX)
- w skrypcie można sprawdzić, czy dana funkcjonalność jest dostępna w przeglądarce i odpowiednio zareagować,
- javascript może być wykorzystany w celu poprawy błędów np. obsłudze stylu css przez przeglądarkę

Do czego można wykorzystać JavaScript?

JavaScript jest prosty w implementacji – wystarczy napisać prosty plik tekstowy i podłączyć go do strony WWW, działa po stronie klienta – zapewnia szybszą reakcję na działania użytkownika i ogranicza zużycie zasobów po stronie serwera, dodaje dynamiczną zawartość na stronie – nie trzeba przysyłać strony na serwer żeby wykonać jakąś zmianę w treści, może załadować treść w momencie gdy użytkownik jej potrzebuje – bez konieczności przeladowania całej strony (AJAX), w skrypcie można sprawdzić, czy dana funkcjonalność jest dostępna w przeglądarce i odpowiednio zareagować, javascript może być wykorzystany w celu poprawy błędów np. obsłudze stylu css przez przeglądarkę

Slajd
14

Zastosowania JavaScript

- Dawniej służył głównie do obsługi formularzy
 - alert, confirm, prompt
- Obecnie:
 - dostęp do elementów strony, modyfikacja ich wyglądu, treści i atrybutów
 - tworzenie nowych elementów na stronie
 - podpowiadanie tekstu w wyszukiwarce, a nawet zwracanie wstępnych wyników

Dawniej służył głównie do obsługi formularzy

- alert, confirm, prompt

Obecnie dzięki obsłudze DOM można odczytać dowolny element na stronie i zmieniać jego wygląd, zawartość i atrybuty. Możliwe jest także tworzenie nowych elementów.

Tym samym nie musimy obecnie wykorzystywać funkcji alert, aby wyświetlić komunikat użytkownikowi. Możemy dodać dodatkowy element na stronie, w którym wpisujemy treść naszego komunikatu. JavaScript może zostać również wykorzystany do tworzenia podpowiedzi w oknach wyszukiwania, a nawet zwracania wstępnych wyników jeszcze w trakcie pisania zapytania.

Slajd
15



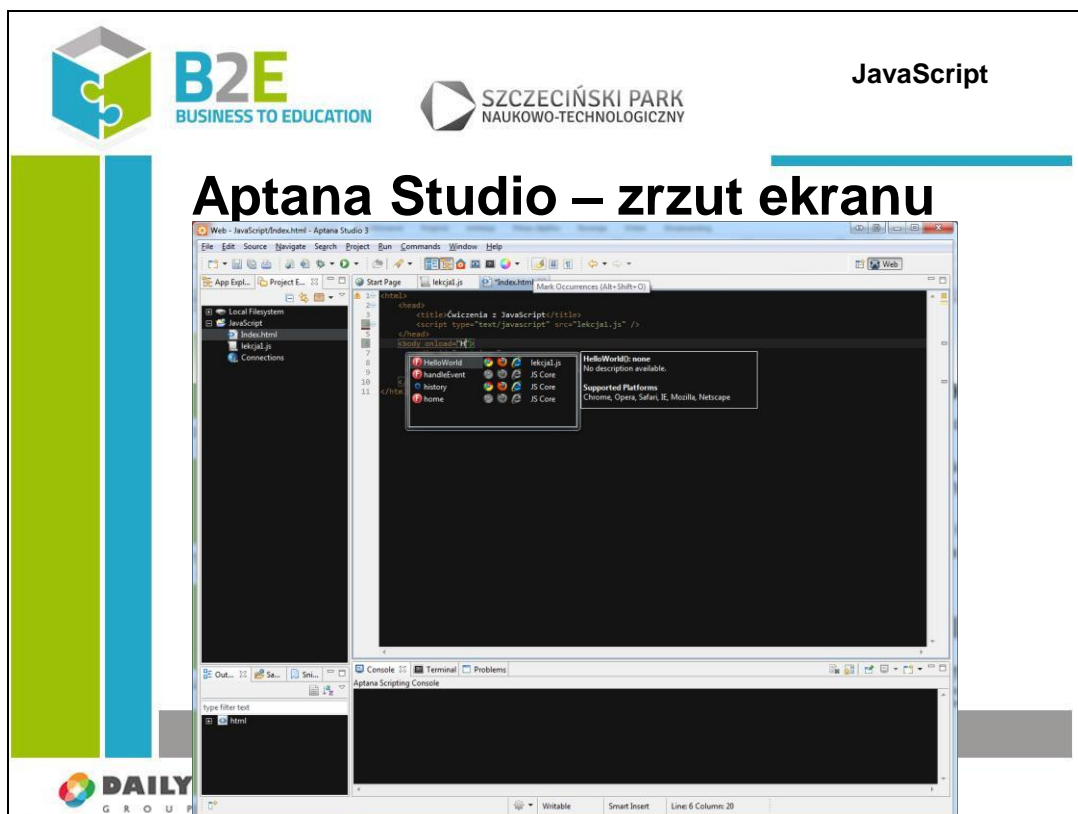
JavaScript

Środowisko do pracy z JS

- JavaScript można uruchomić w dowolnej przeglądarce.
- Nie wymaga specjalistycznego środowiska programistycznego (można korzystać z dowolnego edytora tekstu)
- Edytory z kolorowaniem składni (np. notepad++)
- Środowiska programistyczne dla webdeveloperów Aptana Studio, Netbeans, Eclipse

Skrypty JS można uruchomić w dowolnej przeglądarce internetowej, a napisać w dowolnym edytorze tekstu. Dla łatwiejszego czytania kodu warto skorzystać z edytorów automatycznie kolorujących wpisywany tekst. Istnieją także wyspecjalizowane środowiska programistyczne dla webdeveloperów (obsługujące php, html i JavaScript). Przykładem takiego środowiska jest Aptana Studio – dostępna albo jako plugin do eclipse, albo jako wersja samodzielna.

Slajd
16




JavaScript


Aptana Studio – zrzut ekranu

Na ekranie mamy zrzut ekranu z Aptana Studio – widać, że środowisko podpowiada nam przykładowe zdefiniowane funkcje, koloruje składnie, uzupełnia brakujące nawiasy itp.

Slajd
17



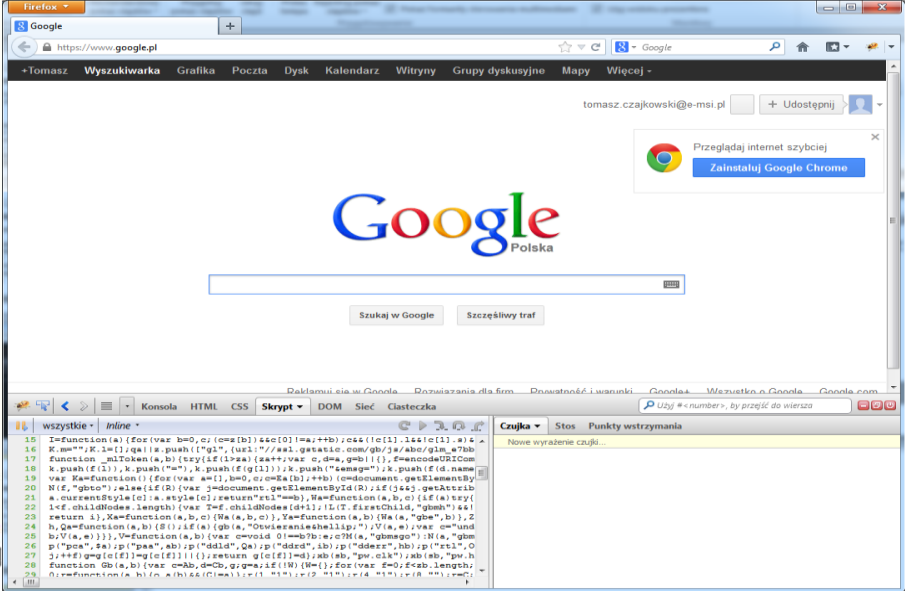
B2E
BUSINESS TO EDUCATION



SZCZECIŃSKI PARK
NAUKOWO-TECHNOLOGICZNY

JavaScript

Firebug – wtyczka dla developerów




Oprócz tworzenia i wyświetlania efektów działania skryptów, często potrzebujemy też narzędzia pozwalającego na sprawdzenie działania skryptu – np. konsoli wyświetlającej błędy, wartości zmiennych tymczasowych itp.

Można w tym celu wykorzystać dodatkowe narzędzia udostępniane przez przeglądarki internetowe (wbudowane w przeglądarkę, albo dodawane jako wtyczki).


Przykładem takich narzędzi jest Firebug, dostępny dla przeglądarki Firefox (okrojone wersje istnieją też dla innych przeglądarek – np. Firebug lite dla Chrome)

Przeglądarka Google Chrome ma wbudowane narzędzia dla programistów Web (dostępne w menu -> narzędzia -> narzędzia dla programistów, konsola JavaScript)

Slajd
18



B2E
BUSINESS TO EDUCATION



SZCZECIŃSKI PARK
NAUKOWO-TECHNOLOGICZNY

JavaScript

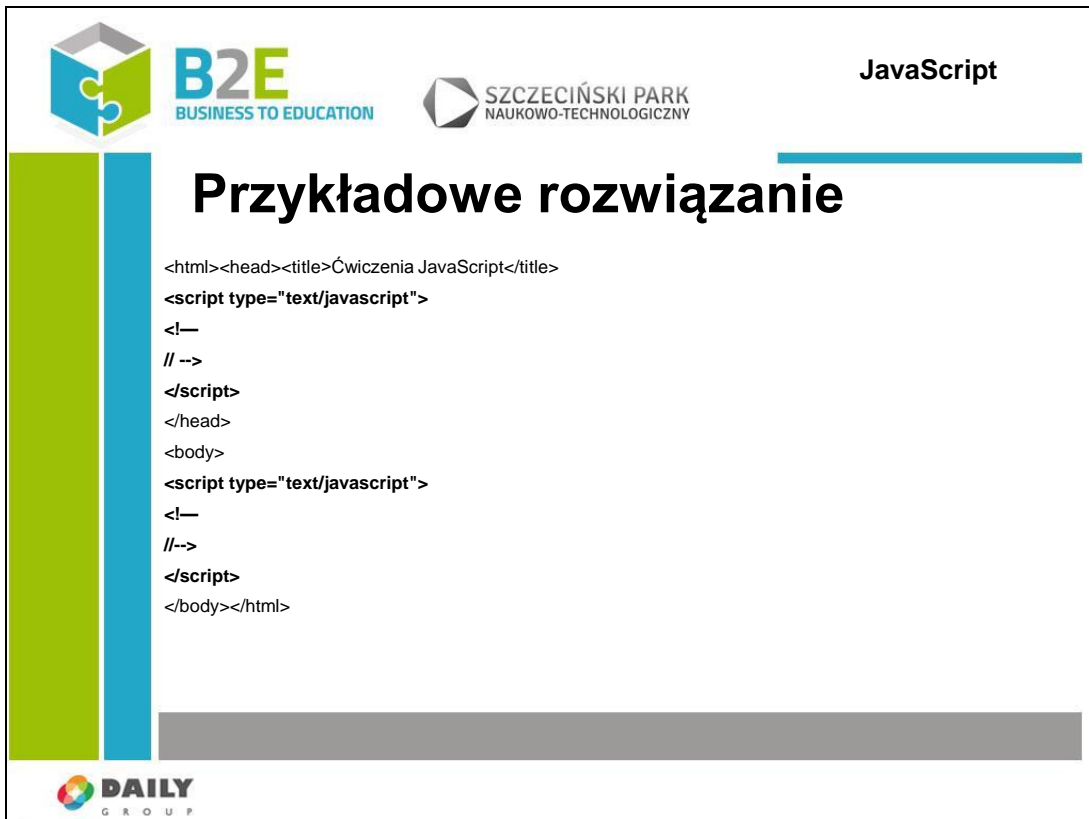
Ćwiczenie

- Przygotuj plik index.html z podstawowymi znacznikami html i dodaj do niego znaczniki umożliwiające obsługę skryptów JS

Utworzony plik będzie wykorzystywany na zajęciach jako podstawa do wykonywania ćwiczeń.



Slajd
19



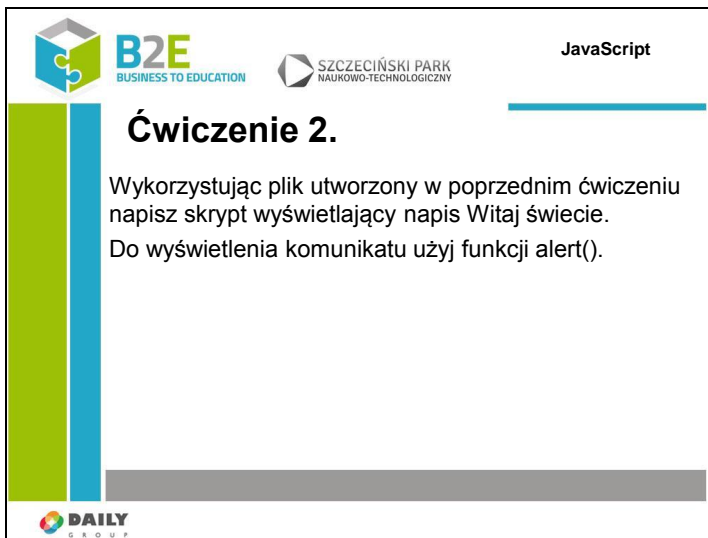
Slide 19 content: The slide features a header with logos for B2E (Business to Education), Szczeciński Park Naukowo-Technologiczny, and JavaScript. The main title is "Przykładowe rozwiązanie". Below the title, there is a code block showing two ways to include a JavaScript script in an HTML document: one in the <head> section and one in the <body> section. The code is as follows:

```
<html><head><title>Ćwiczenia JavaScript</title>
<script type="text/javascript">
<!--
// -->
</script>
</head>
<body>
<script type="text/javascript">
<!--
//-->
</script>
</body></html>
```

The slide also includes logos for DAILY GROUP at the bottom left.

Blok skryptu może być zagnieżdżony w nagłówku strony (skrypt zostanie wtedy wczytany i wykonany przed przetworzeniem treści strony) lub w sekcji <body> - skrypt zostanie wykonany po wczytaniu zdefiniowanej wcześniej zawartości strony. Oba rozwiązania są poprawne.

Slajd
20



Slide 20 content: The slide features a header with logos for B2E (Business to Education), Szczeciński Park Naukowo-Technologiczny, and JavaScript. The main title is "Ćwiczenie 2.". Below the title, there is a text block with the following instructions:

Wykorzystując plik utworzony w poprzednim ćwiczeniu napisz skrypt wyświetlający napis Witaj świecie. Do wyświetlenia komunikatu użyj funkcji alert().

The slide also includes logos for DAILY GROUP at the bottom left.

Komunikat Witaj świecie to najprostszy sposób na rozpoczęcia pracy z każdym językiem programowania.



Slajd
21



B2E
BUSINESS TO EDUCATION



SZCZECIŃSKI PARK
NAUKOWO-TECHNOLOGICZNY


JavaScript

Rozwiązanie


```
<html><head><title>Ćwiczenia JavaScript</title>
</head>
<body>
<script type="text/javascript">
<!--
alert("Witaj świecie");
//-->
</script>
</body></html>
```



Slajd
22



B2E
BUSINESS TO EDUCATION




SZCZECIŃSKI PARK
NAUKOWO-TECHNOLOGICZNY

JavaScript

Ćwiczenie 3.

Zmodyfikuj skrypt z poprzedniego ćwiczenia, tak aby na stronie pojawił się komunikat "Witaj imie". Zmienna imie utwórz przez przypisanie wyniku wywołania funkcji prompt. Do wypisania danych na ekranie użyj funkcji *document.write()* dopisującej treść do strony html.



Język JavaScript uruchamiany jest po stronie klienta (przeglądarki), więc umożliwia bezpośrednią komunikację z użytkownikiem (pozwala na wysyłanie komunikatów do użytkownika i reagowanie na wprowadzane przez niego odpowiedzi). Zmienne tworzymy przez jej zadeklarowanie a następnie przypisujemy jej wartość lub wynik obliczeń.



Slajd
23



Slide 23 content: The slide features a header with logos for B2E (Business to Education), Szczeciński Park Naukowo-Technologiczny, and JavaScript. The main title is "Rozwiązanie". Below it is a code block for JavaScript that prompts the user for their name and displays a greeting. The slide also includes a vertical green and blue bar on the left and a horizontal blue bar under the title. Logos for DAILY GROUP are visible at the bottom.

B2E
BUSINESS TO EDUCATION

SZCZECIŃSKI PARK
NAUKOWO-TECHNOLOGICZNY

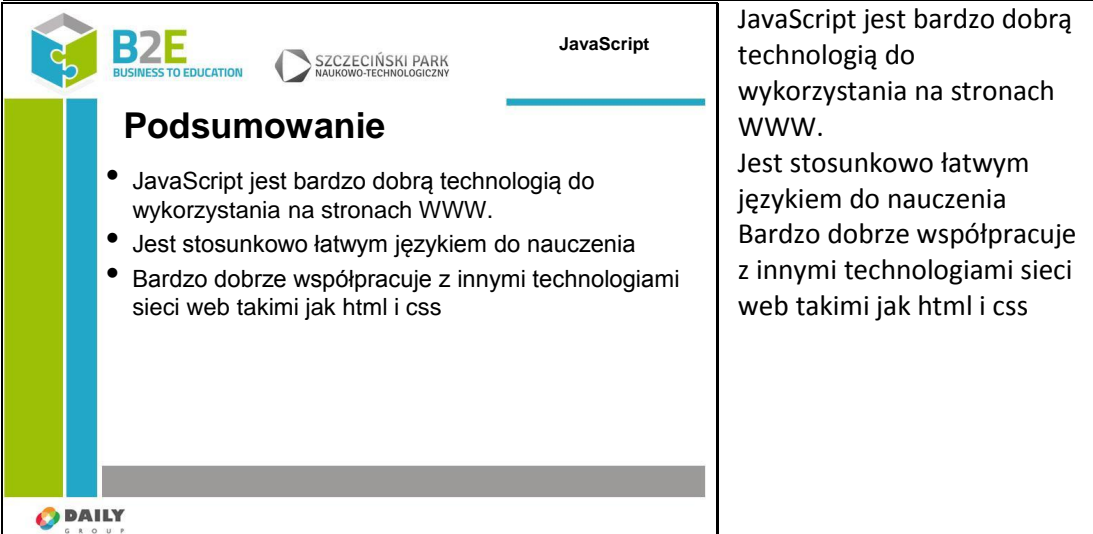
JavaScript

Rozwiązanie

```
<script type="text/javascript">  
<!--  
var imie = prompt("Podaj swoje imię: ");  
document.write("Witaj " + imie);  
//-->  
</script>
```

DAILY GROUP

Slajd
24



Slide 24 content: The slide features a header with logos for B2E (Business to Education), Szczeciński Park Naukowo-Technologiczny, and JavaScript. The main title is "Podsumowanie". Below it is a bulleted list summarizing JavaScript's benefits. The slide also includes a vertical green and blue bar on the left and a horizontal blue bar under the title. Logos for DAILY GROUP are visible at the bottom.

B2E
BUSINESS TO EDUCATION

SZCZECIŃSKI PARK
NAUKOWO-TECHNOLOGICZNY

JavaScript

Podsumowanie

- JavaScript jest bardzo dobrą technologią do wykorzystania na stronach WWW.
- Jest stosunkowo łatwym językiem do nauczania
- Bardzo dobrze współpracuje z innymi technologiami sieci web takimi jak html i css

DAILY GROUP

JavaScript jest bardzo dobrą technologią do wykorzystania na stronach WWW.
Jest stosunkowo łatwym językiem do nauczania
Bardzo dobrze współpracuje z innymi technologiami sieci web takimi jak html i css

Slajd
25



Ćwiczenia

Utwórz przykładową stronę www, zawierającą podstawowe znaczniki HTML i JavaScript. Utworzoną stronę będziemy wykorzystywać jako przykłady w innych lekcjach

Opis założonych osiągnięć ucznia

Po tej lekcji uczestnicy będą znać podstawowe zastosowania języka JavaScript związane z obsługą stron WWW, będą potrafili utworzyć prostą stronę internetową wykorzystującą JavaScript

Lekcja 2 Zmienne, typy danych i operatory

Cel lekcji

Celem lekcji jest zapoznanie uczestników z podstawowymi typami danych dostępnymi w języku JavaScript, sposobami definiowania zmiennych oraz z podstawowymi operatorami służącymi do manipulacji wartością zmiennych skryptu.

Sposoby osiągnięcia celów kształcenia

Czynności nauczyciela	Czynności ucznia
podanie tematu zajęć, przypomnienie informacji z poprzednich lekcji	poznaje temat, odpowiada na pytania nauczyciela związane z materiałem z poprzednich zajęć
wprowadza nowe pojęcia związane z tematem zajęć (materiał ze slajdów) przygotowuje ćwiczenia z zaprezentowanego materiału	Wykonuje zadane przez nauczyciela ćwiczenia na komputerze
Podsumowuje lekcję, ocenia pracę uczniów na zajęciach.	Dokonuje samooceny swojej pracy, omawia z nauczycielem rozwiązania ćwiczeń

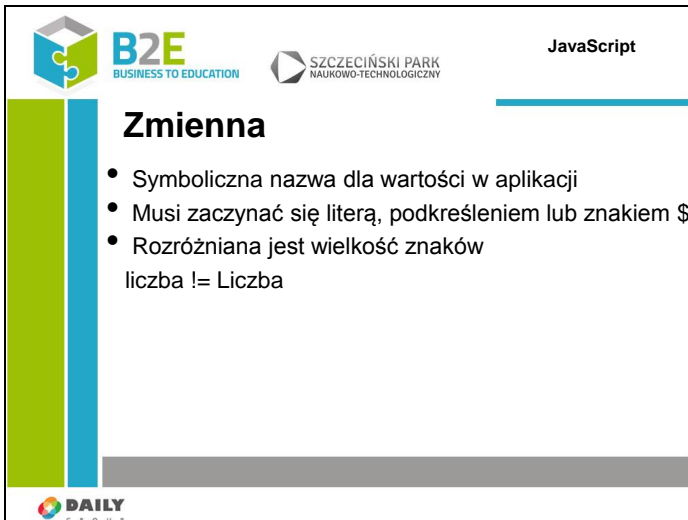
Treść - slajdy z opisem

Slajd 1



Slide 1: JavaScript title slide. The slide features a green and blue vertical bar on the left. Logos for B2E (Business to Education), Szczeciński Park Naukowo-Technologiczny, and DAILY GROUP are visible. The main title is 'JavaScript' in large black font, with the subtitle 'Zmienne, typy danych operatory' below it. At the bottom, there are logos for 'KAPITAŁ LUDZKI' and 'UNIA EUROPEJSKA'.

Slajd 2



Slide 2: Variable definition. The slide features a green and blue vertical bar on the left. Logos for B2E (Business to Education), Szczeciński Park Naukowo-Technologiczny, and DAILY GROUP are visible. The main title is 'Zmienna' in large black font. Below it is a bulleted list:

- Symboliczna nazwa dla wartości w aplikacji
- Musi zaczynać się literą, podkreśleniem lub znakiem \$
- Rozróżniana jest wielkość znaków
liczba != Liczba

W poprzedniej lekcji wspomnieliśmy o zastosowaniach języka JavaScript i sposobach podłączania skryptów js w naszej witrynie. W tej lekcji zajmiemy się omówieniem podstawowych elementów służących do przechowywania danych oraz zobaczymy jakie podstawowe typy danych są dla nas dostępne. Zaczniemy od wprowadzenia pojęcia zmiennej.

Zmiennych używa się jako symbolicznych nazw dla wartości w Twojej aplikacji. Każda zmienna ma swoją nazwę, która pozwala na identyfikację tej zmiennej w kodzie skryptu. Zmienna charakteryzuje się również typem, który określa rodzaj danych jakie przechowuje. Nazwy zmiennych, nazywane *identyfikatorami*, podporządkowane są pewnym regułom.

Identyfikator JavaScript musi zaczynać się literą, podkreśleniem (_) lub znakiem dolara (\$); kolejne znaki mogą być cyframi (0-9). Ponieważ JavaScript rozróżnia duże/małe litery, litery oznaczają znaki od "A" do "Z" (wielkie litery) oraz znaki od "a" do "z" (małe litery).



Slajd 3

JavaScript

Deklaracja zmiennych

Zmienną można zadeklarować na 2 sposoby:

- Zmienne lokalne: za pomocą słowa kluczowego `var`.
Na przykład:
`var liczba = 10,`
`var nazwisko = "Kowalski"`
- Zmienne globalne: za pomocą prostego przypisania wartości. Na przykład:
`liczba = 10, nazwisko = "Kowalski"`

DAILY GROUP

Zmienna zadeklarowana za pomocą słowa kluczowego `var` jest zmienną lokalną, natomiast bez słowa kluczowego zmienną globalną (więcej informacji na temat zmiennych lokalnych i globalnych będzie w późniejszych lekcjach)

Slajd 4

JavaScript

Typy danych

- typ liczbowy
- typ łańcuchowy
- typ logiczny
- typ specjalny

DAILY GROUP

Typ danych określa rodzaj danych jaki przechowuje. W JavaScriptcie możemy podzielić typy na następujące rodzaje:
typ liczbowy
typ łańcuchowy
typ logiczny
typ specjalny



Slajd 5

Typ liczbowy

- służy do reprezentacji liczb.
- liczby zapisywane są za pomocą literałów liczbowych, czyli ciągów znaków składających się na liczbę

```
var pi = 3.1415;  
var licznik = 10;
```

Typ liczbowy służy do reprezentacji liczb, przy czym nie ma występującego w klasycznych językach programowania rozróżnienia na typy całkowitoliczbowe i rzeczywiste (zmiennopozycyjne). Liczby zapisywane są za pomocą literałów (stałych napisowych, z ang. string constant, literal constant) liczbowych, czyli ciągów znaków składających się na liczbę.

Obowiązują przy tym następujące zasady.

Jeżeli ciąg cyfr nie jest poprzedzony żadnym znakiem lub jest poprzedzony znakiem +, reprezentuje wartość dodatnią, jeżeli natomiast jest poprzedzony znakiem -, reprezentuje wartość ujemną.


Jeżeli literał rozpoczyna się od cyfry 0, jest traktowany jako wartość ósemkowa.

Jeżeli literał rozpoczyna się od ciągu znaków 0x, jest traktowany jako wartość szesnastkowa (heksadecymalna). W zapisie wartości szesnastkowych mogą być wykorzystywane zarówno małe, jak i wielkie litery alfabetu, od A do F.

Literały mogą być zapisywane w notacji wykładniczej, w postaci X.YeZ, gdzie X to część całkowita, Y — część dziesiętna, natomiast Z to wykładnik potęgi liczby 10. Zapis taki oznacza to samo, co $X.Y * 10^Z$



Slajd 6




JavaScript

Typ łańcuchowy


- służy do reprezentacji ciągów znaków (napisów).
- powinien być ujęty w znaki cudzysłowu, aczkolwiek dopuszczalne jest również wykorzystanie znaków apostrofu.

```
var napis = "Witaj świecie!";
var nazwisko = 'Kowalski';
```



Typ łańcuchowy (inaczej, typ string) służy do reprezentacji ciągów znaków (napisów). Ciągi te (inaczej, stałe napisowe) powinny być ujęte w znaki cudzysłowu, aczkolwiek dopuszczalne jest również wykorzystanie znaków apostrofu. Przykładowy ciąg ma postać: "abcdefg" lub 'abcdefg'

Slajd 7




JavaScript

Typ łańcuchowy

Ciągi mogą też zawierać poniższe znaki specjalne

- \b - backspace
- \n - nowa linia
- \r - powrót karetki
- \f - nowa strona
- \t - tabulacja
- \" - cudzysłów
- \' - apostrof
- \\ - lewy ukośnik

```
var cytāt = "Marcin powiedział: \"Jedziemy na wycieczkę\"";
```



Ciągi mogą też zawierać poniższe znaki specjalne

- \b - backspace
- \n - nowa linia
- \r - powrót karetki
- \f - nowa strona
- \t - tabulacja
- \" - cudzysłów
- \' - apostrof
- \\ - lewy ukośnik

Slajd 8

JavaScript

Typ logiczny

- pozwala na określenie dwóch wartości logicznych: prawda i fałsz.
- wartości tego typu są wykorzystywane przy konstruowaniu wyrażeń logicznych, porównywaniu danych itp.

DAILY GROUP

Typ logiczny (boolean) pozwala na określenie dwóch wartości logicznych: prawda i fałsz. Wartość prawda jest w JavaScriptcie reprezentowana przez słowo true, natomiast wartość fałsz przez false. Wartości tego typu są używane przy konstruowaniu wyrażeń logicznych, porównywaniu danych, wskazywaniu, czy dana operacja zakończyła się sukcesem itp.

Slajd 9

JavaScript

Typ specjalny

2 wartości:

- null
- undefined

DAILY GROUP

Możemy wyróżnić dwa rodzaje typów specjalnych: null i undefined. Choć podobne, nie są tożsame.

typ null to specjalne słowo oznaczające wartość null (pustą), ponieważ JavaScript rozróżnia małe/duże litery, null to nie to samo co Null, NULL czy jakkolwiek inaczej.

typ undefined to podstawowa właściwość, której wartość jest nieokreślona. W tym kontekście wartość undefined ma niezainicjalizowana zmienna, zmienna, której jawnie przypisano wartość undefined, bądź też nieistniejąca właściwość obiektu.



Slajd 10

JavaScript

Zmienne a typy danych

- zmienne mogą zmieniać swój typ w trakcie działania programu

```
var mojaZmienna = "12345"

mojaZmienna = mojaZmienna - 345
```

wynik: 12000

W JavaScriptcie w przeciwieństwie do wielu języków programowania zmienna może automatycznie zmieniać swój typ. Np. zmienna łańcuchowa może zostać zamieniona na zmienną liczbową lub odwrotnie w zależności od sposobu odwołania do zmiennej. Np. poniższa deklaracja przypisze zmiennej mojaZmienna wartość typu tekst "12345"

```
var mojaZmienna = "12345"
```

Pomimo tego, że ten łańcuch składa się z samych cyfr jest zmienną tekstową. Jednak JavaScript jest na tyle sprytny, że w poniższym przypisaniu zamieni typ znakowy na liczbę przed odejmowaniem i wynikową wartość (w tym przypadku liczba 12000) zapisze ponownie do zmiennej mojaZmienna

```
mojaZmienna = mojaZmienna - 345
```

Slajd 11

JavaScript

Zmienne a typy danych cd.

- Ale:
- `var mojaZmienna = "12345"`
- `mojaZmienna = mojaZmienna + 678`
- Wynik = „12345678”
- `mojaZmienna = Number(mojaZmienna) + 678`

Niestety taka dynamiczna konwersja ma swoje wady. W przypadku przypisania

```
var mojaZmienna = "12345"
```

mojaZmienna = mojaZmienna +678

interpreter JS nie zmieni łańcucha "12345" na liczbę tylko zamieni liczbę 678 na łańcuch "678" i połączy oba łańcuchy co da w wyniku "12345678".

Aby zapobiec takiej sytuacji należy wskazać, że mojaZmienna jest typem liczbowym

mojaZmienna = Number(mojaZmienna) + 678

Taka instrukcja nazywana jest rzutowaniem na typ. W języku JavaScript dostępne są następujące funkcje do zmiany typu zmiennej:

Boolean() - zmienia wartość na prawdę lub fałsz

Number() - zmienia wartość na liczbową


String() - zmienia wartość na łańcuch znaków

Przy rzutowaniu na wartość logiczną łańcuch znaków o minimalnej długości 1, dowolny obiekt lub liczba większa od 0 będzie miała wartość "true", z kolei pusty łańcuch znaków, liczba 0 oraz typy specjalne null i undefined będą miały wartość "false"


Jeśli wartość rzutowana na typ liczbowy, zawiera cokolwiek oprócz cyfr to wynikiem jest NaN (not a number).

Funkcja String() zwraca dowolną wartość jako łańcuch znaków, nawet typy specjalne jak null i undefined.

Slajd 12



B2E
BUSINESS TO EDUCATION



SZCZECIŃSKI PARK
NAUKOWO-TECHNOLOGICZNY

JavaScript

Instrukcje


Instrukcja to polecenie do wykonania, zakończone średnikiem. Możemy np.

- wyświetlić okno dialogowe lub pobrać dane od użytkownika,


```
alert("Komunikat");
            prompt("Podaj imię: ");
```
- pobrać element strony


```
document.getElementById("tekst");
```
- umieścić treść na stronie www


```
document.write("Komunikat");
```



Instrukcję możemy potraktować jak polecenie do wykonania. Skrypt napisany w języku JavaScript to w istocie ciąg instrukcji, a zatem ciąg poleceń. Możemy zatem nakazać wyświetlenie okna dialogowego, umieścić jakąś treść na stronie WWW, zmodyfikować zawartość warstwy czy innego elementu witryny itp.

Co ważne, każdą instrukcję powinniśmy zakończyć znakiem średnika. To informacja dla interpretera, że jest to koniec instrukcji.

Przykładowe instrukcje:

```
var zmienna = 10;
```

```
var element = document.getElementById("content");
```

```
content.innerHTML = "Tekst";
```



Slajd 13



Slide 13 content: JavaScript examples. Includes logos for B2E, Szczeciński Park Naukowo-Technologiczny, and Daily Group. The slide title is 'Przykłady instrukcji' and it shows a code snippet for variable declaration and DOM manipulation.

JavaScript

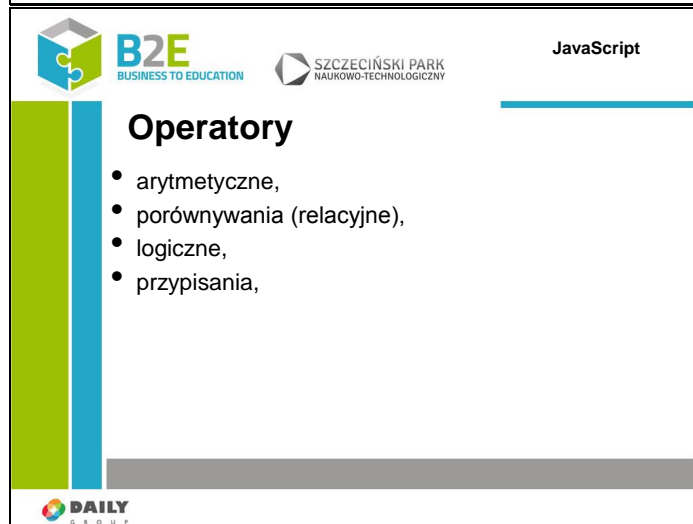
Przykłady instrukcji

Przykładowe instrukcje:

```
var zmienna = 10;  
var element =  
document.getElementById("content");  
content.innerHTML = "Tekst";
```

DAILY GROUP

Slajd 14



Slide 14 content: JavaScript operators. Includes logos for B2E, Szczeciński Park Naukowo-Technologiczny, and Daily Group. The slide title is 'Operatory' and it lists four categories of operators: arithmetic, relational, logical, and assignment.

JavaScript

Operatory



- arytmetyczne,
- porównywania (relacyjne),
- logiczne,
- przypisania,

DAILY GROUP

W JavaScriptcie, podobnie jak w innych językach programowania, występują operatory pozwalające na wykonywanie rozmaitych operacji. Operatory możemy podzielić na następujące grupy:
arytmetyczne,
porównywania (relacyjne),
logiczne,
przypisania,




Slajd 15



JavaScript

Operatory arytmetyczne


Operator	Opis	Przykład	Wynik
+	Dodawanie	3 + 11	14
-	Odejmowanie	9 - 4	5
*	Mnożenie	3 * 4	12
/	Dzielenie	21 / 7	3
%	Reszta z dzielenia	21 % 8	5
++	Zwiększanie	a= 5; ++a	6
--	Zmniejszanie	a= 5; --a	4



Operatory arytmetyczne w JavaScript pozwalają na tworzenie wyrażeń numerycznych.

Operator % to operator modulo - podaje jako wynik resztę z dzielenia, operatory ++ i -- są uproszczonymi wersjami operatorów dodawania o 1 i odejmowania o 1

Slajd 16



JavaScript

Operatory porównania

Operator	Opis	Przykład	Wynik
==	Równe	1 == 1	Prawda
===	Równe w wartości i typie	1 === '1'	Falsz
!=	Nie równe	1 != 2	Prawda
!==	Nie równe w wartości i typie	1 !== '1'	Prawda
>	Większe niż	1 > 2	Falsz
<	Mniejsze niż	1 < 2	Prawda
>=	Większe lub równe	1 >= 1	Prawda
<=	Mniejsze lub równe	2 <= 1	Falsz




Operator == i === różnią się sposobem dokładnością sprawdzania




warunek `if (1 == '1')` da w wyniku wartość "true" ponieważ '1' może być automatycznie zamienione na 1.

natomiast warunek `if (1 === '1')` da w wyniku wartość "false" ponieważ mimo że wartości mogą być uznane za takie same to zmienne są różnego typu

Slajd 17



B2E
BUSINESS TO EDUCATION




SZCZECIŃSKI PARK
NAUKOWO-TECHNOLOGICZNY

JavaScript

Operatory logiczne

Operator	Opis	Przykład	Wynik
&&	I	<code>1 == 1 && 2 == 2</code>	Prawda
	Lub	<code>1 == 1 2 == 3</code>	Prawda
!	Negacja	<code>!(1 == 1)</code>	Fałsz





JavaScript wspiera 3 operatory logiczne, za pomocą których można rozszerzać instrukcję `if`.

Operatory w języku JavaScript sprawdzane są w kierunku od lewej do prawej. Tzn. jeśli w warunku zawierającym operator `&&` na pierwszym miejscu będzie wyrażenie którego wartość nie będzie spełniona to sprawdzanie całego warunku zostanie przerwane i jako wynik będzie zwrócona wartość Fałsz




Slajd 18



JavaScript

Operatory przypisania

Operator	Opis	Przykład	Wynik w a
=	Proste przypisanie	a = 42	42
+=	Przypisanie z dodawaniem	a += 5	47
-=	Przypisanie z odejmowaniem	a -= 2	45
*=	Przypisanie z mnożeniem	a *= 2	90
/=	Przypisanie z dzieleniem	a /= 10	9
%=	Przypisanie z modulo	a %= 4	1



Przykładowo zamiast pisać a = a + 5 można użyć prostszego a += 5.

Powyższych operatorów można używać w połączeniu z innymi operatorami i zmiennymi

a = 10;

b = 25;

a += (b / 5)

wynikiem operacji jest 15 (10 + (25/5))

Slajd 19



JavaScript

Kolejność operatorów

Priorytet	Operatory	Priorytet	Operatory
1	. [] new	10	&
2	()	11	^
3	++ --	12	
4	! ~ unary+ unary- typeof void delete	13	&&
5	* / %	14	
6	+ -	15	?:
7	<< >> >>>	16	= += -= *= /= %= << >>= >>>= &= ^= !=
8	< <= > >= in instanceof	17	,
9	== != === !==		





Oprócz znajomości operatorów, niezbędna jest jeszcze wiedza na temat ich priorytetów, czyli kolejności wykonywania. Wiadomo np., że mnożenie jest „silniejsze” od dodawania, zatem najpierw mnożymy, potem dodajemy (kolejność tę można zmienić, stosując nawiasy okrągłe, dokładnie w taki sam sposób, w jaki zmienia się kolejność działań w matematyce). W JavaScriptcie jest podobnie — siła każdego operatora kolejność wykonywania jest ściśle określona.


Slajd 20

Slajd 21


```
var a = prompt("Podaj pierwszą liczbę: ");
var b = prompt("Podaj drugą liczbę: ");
a = Number(a);
b = Number(b);
var tekst = "Suma liczb " + a + " i " + b + " wynosi " + (a+b);// + "<br>";
document.writeln(tekst);
tekst = "Różnica liczb " + a + " i " + b + " wynosi " + (a-b);// + "<br>";
document.writeln(tekst);
tekst = "Iloczyn liczb " + a + " i " + b + " wynosi " + (a * b);// + "<br>";
document.writeln(tekst);
tekst = "Iloraz liczb " + a + " i " + b + " wynosi " + (a/b);// + "<br>";
document.writeln(tekst);
```



Slajd 22



B2E
BUSINESS TO EDUCATION




SZCZECIŃSKI PARK
NAUKOWO-TECHNOLOGICZNY

JavaScript

Przykładowe rozwiązanie

```
<script type="text/javascript">
var a = prompt("Podaj pierwszą liczbę: ");
var b = prompt("Podaj drugą liczbę: ");
a = Number(a);
b = Number(b);
var tekst = "Suma liczb " + a + " i " + b + " wynosi " + (a+b); // + "<br>";
document.writeln(tekst);
tekst = "Różnica liczb " + a + " i " + b + " wynosi " + (a-b); // + "<br>";
document.writeln(tekst);
tekst = "Iloczyn liczb " + a + " i " + b + " wynosi " + (a * b); // + "<br>";
document.writeln(tekst);
tekst = "Iloraz liczb " + a + " i " + b + " wynosi " + (a/b); // + "<br>";
document.writeln(tekst);
</script>
```



Wynikiem działania funkcji prompt jest zmienna tekstowa zawierająca treść wpisaną przez użytkownika.

Przed wykonaniem obliczeń matematycznych za pomocą funkcji Number musimy zmienić typ zmiennych na numeryczny.

Do wyświetlenia wyników używamy np. funkcji document.writeln() (dodaje przejście do nowej linii po wyświetlonym tekście) lub funkcji document.write().



Slajd 23



B2E
BUSINESS TO EDUCATION



SZCZECIŃSKI PARK
NAUKOWO-TECHNOLOGICZNY

JavaScript

Ćwiczenia:

Ćwiczenie1
Napisz skrypt wyświetlający na ekranie napis Witaj i Twoje imię.

Ćwiczenie 2:
Zmodyfikuj skrypt z poprzedniego ćwiczenia w taki sposób, aby przed wyświetleniem pojawiło się okno dialogowe z prośbą o podanie imienia. Wyświetl wpisane dane na ekranie



KAPITAŁ LUDZKI
NARODOWA STRATEGIA SPÓJNOŚCI

Człowiek - najlepsza inwestycja



UNIA EUROPEJSKA
EUROPEJSKI
FUNDUSZ SPOŁECZNY

Projekt współfinansowany przez Unię Europejską w ramach Europejskiego Funduszu Społecznego



```
document.write("witaj Tomek");
var element = document.getElementById("content");
element.innerHTML = "Witaj Tomek";
alert("Witaj Tomek");
Ćwiczenie 2
var imie = prompt("Podaj imię");
document.write("Witaj " + imie);
var element = document.getElementById("content");
element.innerHTML = "Witaj " + imie;
var tekst = "Witaj" + imie;
alert(tekst);
```

Ćwiczenia

Napisz skrypt obliczający wartość wprowadzonego wyrażenia arytmetycznego. Do obliczeń wykorzystaj funkcję eval()

Opis założonych osiągnięć ucznia

Uczestnik zna podstawowe typy danych i operatory języka JavaScript oraz potrafi wykorzystać je w praktyce

Lekcja 3 Instrukcje warunkowe i pętle

Cel lekcji.

Celem lekcji jest wprowadzenie instrukcji pozwalających na sterowanie wykonywaniem programu. Poznamy różne warianty instrukcji warunkowych oraz pętli oraz zobaczymy praktyczne przykłady ich wykorzystania. Instrukcje wprowadzone w tej lekcji są podstawą każdego języka programowania, bez ich znajomości nie można tworzyć żadnych złożonych programów.

Sposoby osiągnięcia celów kształcenia

Czynności nauczyciela	Czynności ucznia
wprowadza nowe pojęcia związane z tematem zajęć (materiał ze slajdów) przygotowuje ćwiczenia z zaprezentowanego materiału	Wykonuje zadane przez nauczyciela ćwiczenia na komputerze
Prowadzi z uczniami dyskusję na temat wybranych przez nich rozwiązań ćwiczeń	Prezentuje rozwiązania ćwiczeń nauczycielowi i pozostałym uczniom, tłumaczy wybrane przez siebie rozwiązanie
Podsumowuje lekcję, ocenia pracę uczniów na zajęciach.	Dokonuje samooceny swojej pracy, omawia z nauczycielem rozwiązania ćwiczeń

Treść – slajdy z opisem

Slajd 1

JavaScript

JavaScript

Instrukcje warunkowe

KAPITAŁ LUDZKI
NARODOWA STRATEGIA SPÓJNOŚCI

SZCZECIŃSKI PARK
NAUKOWO-TECHNOLOGICZNY

UNIA EUROPEJSKA
EUROPEJSKI
FUNDUSZ SPOŁECZNY

Człowiek - najlepsza inwestycja

Projekt współfinansowany przez Unię Europejską w ramach Europejskiego Funduszu Społecznego

DAILY GROUP

W poprzedniej lekcji wprowadziliśmy pojęcia zmiennej oraz instrukcji. Dla przypomnienia – zmienna to symboliczna nazwa dla wartości przechowywanych w trakcie działania programu, a instrukcja to polecenie sprawdzające lub ustawiające jej stan. W tej lekcji pokażemy podstawowe instrukcje warunkowe w języku JavaScript

Slajd 2

JavaScript

Instrukcje warunkowe

Instrukcje warunkowe pozwalają na sterowanie przepływem programu wg. zdefiniowanych warunków

KAPITAŁ LUDZKI
NARODOWA STRATEGIA SPÓJNOŚCI

SZCZECIŃSKI PARK
NAUKOWO-TECHNOLOGICZNY

UNIA EUROPEJSKA
EUROPEJSKI
FUNDUSZ SPOŁECZNY

Człowiek - najlepsza inwestycja

Projekt współfinansowany przez Unię Europejską w ramach Europejskiego Funduszu Społecznego

DAILY GROUP

Instrukcje warunkowe pozwalają na sterowanie przepływem programu wg. zdefiniowanych warunków. Możemy podzielić wykonywanie programu na różne ścieżki i w zależności od wartości zmiennych wybierać te, które powinny zostać wykonane.



Slajd 3

JavaScript

B2E BUSINESS TO EDUCATION

SZCZECIŃSKI PARK NAUKOWO-TECHNOLOGICZNY

Instrukcje warunkowe

Wyróżniamy następujące rodzaje instrukcji:

- if
- if ... else
- if else if ...
- switch

DAILY GROUP

Wyróżniamy następujące rodzaje instrukcji:

- if
- if ... else
- if else if ...
- switch

Pozwalają one na wykonywanie instrukcji w zależności od tego czy określony warunek został spełniony.

Slajd 4

JavaScript

B2E BUSINESS TO EDUCATION

SZCZECIŃSKI PARK NAUKOWO-TECHNOLOGICZNY

Instrukcja *if*

```
if (warunek) {
    instrukcje
}
```

lub jeśli jest tylko jedna instrukcja do wykonania

```
if (warunek)
    instrukcja;
```

DAILY GROUP

Wyrażenia w nawiasach klamrowych zostaną wykonane tylko w przypadku, gdy warunek zostanie spełniony. Warunek musi być typu Boolean. Można go zbudować za pomocą operatorów poznanych we wcześniejszej lekcji.

Nawiasy klamrowe wydzielają blok instrukcji wykonywanych po spełnieniu warunku. Po zamykającym nawiasie nie trzeba wstawiać średnika kończącego polecenie JavaScript.

W przypadku gdy w wyniku spełnienia warunku należy wykonać tylko 1 instrukcję nawiasy można pominąć, ale wymagany wtedy jest ; na końcu linii.



Slajd 5

Slide 5 content: JavaScript if instruction example. The slide features logos for B2E (Business to Education), Szczeciński Park Naukowo-Technologiczny, and Daily Group. The title is 'Instrukcja if'. A code example shows a variable 'liczba' set to 21, followed by an if statement that checks if the number is not divisible by 2. If true, it prints 'Liczba nieparzysta'.

Instrukcja else rozszerza instrukcję if. Działa bardzo podobnie do if, ale zawarte w niej instrukcje są wykonywane tylko w przypadku, gdy warunek nie jest spełniony.


Slajd 6

Slide 6 content: JavaScript else instruction example. The slide features logos for B2E (Business to Education), Szczeciński Park Naukowo-Technologiczny, and Daily Group. The title is 'Instrukcja else'. A code example shows an if statement with a comment indicating it runs when the condition is met, followed by an else block with a comment indicating it runs when the condition is not met.


Instrukcja else rozszerza instrukcję if. Działa bardzo podobnie do if, ale zawarte w niej instrukcje są wykonywane tylko w przypadku gdy warunek nie jest spełniony.



Slajd 7



B2E
BUSINESS TO EDUCATION



SZCZECIŃSKI PARK
NAUKOWO-TECHNOLOGICZNY


JavaScript

Instrukcja *else*

Przykład:


```

liczba = 21;
if (liczba % 2 != 0) {
    document.writeln("Liczba nieparzysta");
}
else {
    document.writeln("Liczba parzysta");
}
    
```




Instrukcja *else* rozszerza instrukcję *if*. Działa bardzo podobnie do *if*, ale zawarte w niej instrukcje są wykonywane tylko w przypadku gdy warunek nie jest spełniony.

Slajd 8



B2E
BUSINESS TO EDUCATION




SZCZECIŃSKI PARK
NAUKOWO-TECHNOLOGICZNY

JavaScript

Instrukcja *else if*

```

if (warunek1) {
    instrukcje1;
}
else if (warunek2) {
    instrukcje2;
}
.....
else if (warunekN) {
    instrukcjeN;
}
else {
    instrukcjeM;
}
    
```




Kolejna wersja instrukcji *if* pozwala na badanie wielu warunków. Po *if* może wystąpić wiele dodatkowych bloków *else if*.


Taka konstrukcja oznacza, że jeżeli warunek1 jest spełniony to wykonane zostaną

instrukcje1, w przeciwnym wypadku gdy spełniony jest warunek2 to zostaną wykonane instrukcje2, itd. W przypadku, gdy wszystkie warunki są fałszywe, wykonywane są instrukcjeM z bloku else.

Slajd 9



B2E
BUSINESS TO EDUCATION




SZCZECIŃSKI PARK
NAUKOWO-TECHNOLOGICZNY

JavaScript

Instrukcja *switch*

```


switch (wyrażenie) {
  case wartość1:
    instrukcja1;
    break;
  case wartość2:
    instrukcja2;
    break;
  ...
  default : instrukcja;
}
```




Kolejna wersja instrukcji if pozwala na badanie wielu warunków. Po if może wystąpić wiele dodatkowych bloków else if.

Taka konstrukcja oznacza, że jeżeli warunek1 jest spełniony to wykonane zostaną instrukcje1, w przeciwnym wypadku gdy spełniony jest warunek2 to zostaną wykonane instrukcje2, itd. W przypadku, gdy wszystkie warunki są fałszywe, wykonywane są instrukcje z bloku else.

Slajd 10



B2E
BUSINESS TO EDUCATION




SZCZECIŃSKI PARK
NAUKOWO-TECHNOLOGICZNY

JavaScript

Instrukcja *switch*

```

switch (wyrażenie) {
  case wartość1:
    instrukcja1;
    break;
  case wartość2:
    instrukcja2;
    break;
  ...
  default : instrukcja;
}
```



Instrukcja wyboru switch (nazywana również instrukcją switch...case) pozwala w wygodny sposób sprawdzić ciąg warunków i wykonać różne instrukcje, w zależności od wyników porównywania. Najpierw program szuka wartość odpowiedniego wyrażenia i wykonuje załączoną instrukcję. Jeśli nie uda się znaleźć odpowiedniej wartości program szuka domyślnej instrukcji i ją wykonuje. Jeśli wartość została znaleziona to program kontynuuje



wykonywanie instrukcji aż do końca instrukcji **switch** lub do napotkania instrukcji **break**.

Slajd 11

Instrukcja *switch*

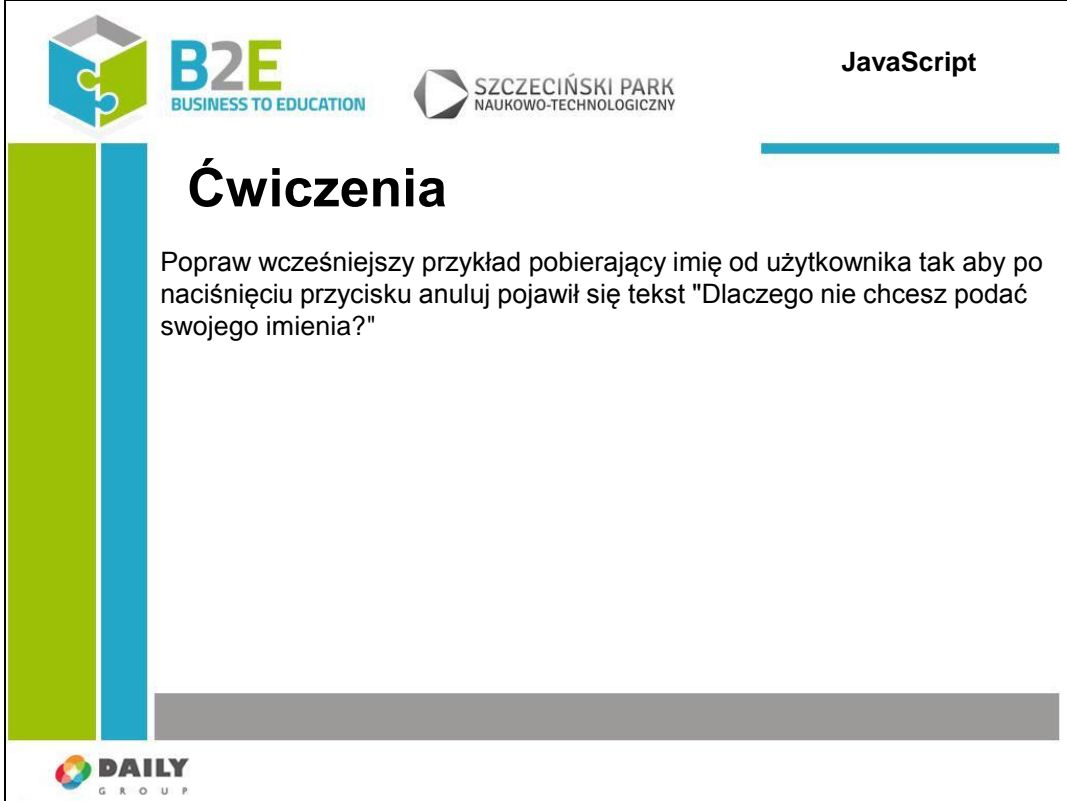
Przykład:

```
kolor = "Biały";
switch (kolor) {
  case "Zielony": document.writeln("Wybrano zielony kolor");
    break;
  case "Niebieski": document.writeln("Wybrano kolor niebieski");
    break;
  case "Biały": document.writeln("Wybrano kolor biały");
    break;
  default: document.writeln("Nieznany kolor");
}
```

W trakcie działania sprawdzane jest czy etykieta "Biały" znajduje się na zdefiniowanej liście etykiet instrukcji. Jeśli tak to wykonywanie programu przechodzi do tego miejsca. Jeśli pominiemy instrukcję break to oprócz napisu Wybrano kolor biały zobaczymy jeszcze napis Nieznany kolor.



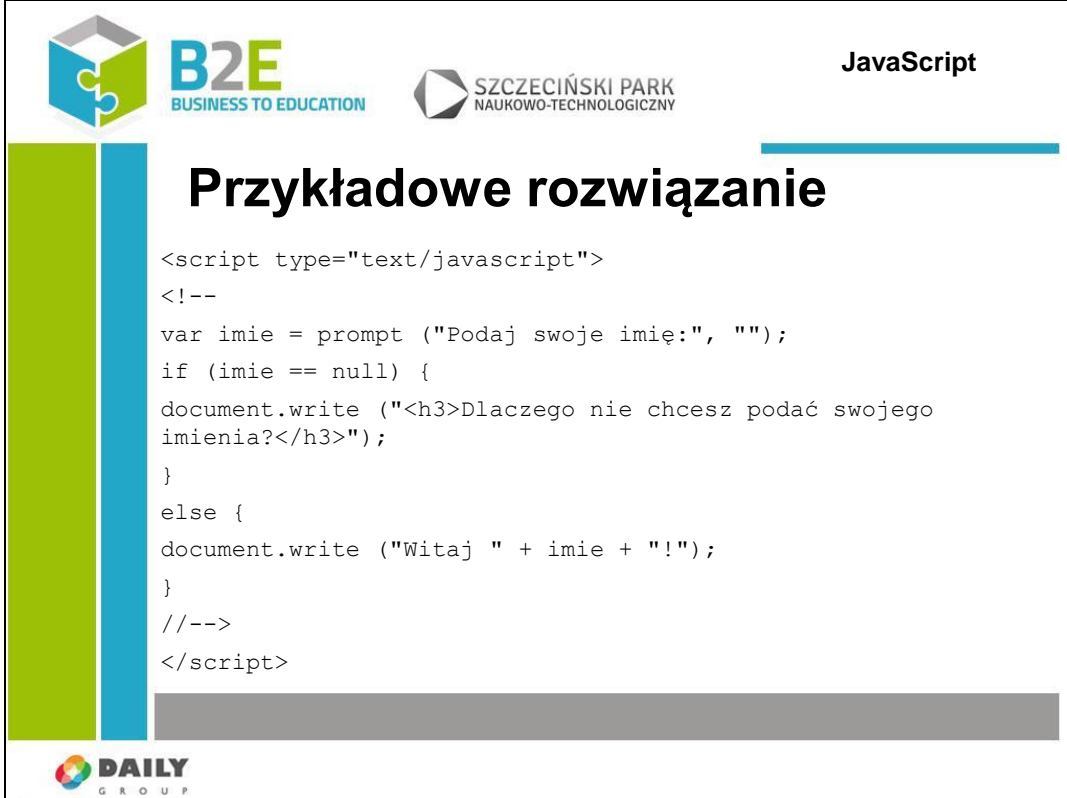
Slajd 12



Slide 12 content: The slide features a header with logos for B2E (Business to Education), Szczeciński Park Naukowo-Technologiczny, and JavaScript. The main title is "Ćwiczenia". The text asks to improve a previous example by adding a prompt for the user's name and a message "Dlaczego nie chcesz podać swojego imienia?" if the name is not provided. The slide is branded with the DAILY GROUP logo at the bottom left.

Popraw wcześniejszy przykład pobierający imię od użytkownika tak aby po naciśnięciu przycisku anuluj pojawił się tekst "Dlaczego nie chcesz podać swojego imienia?"

Slajd 13




Slide 13 content: The slide features a header with logos for B2E (Business to Education), Szczeciński Park Naukowo-Technologiczny, and JavaScript. The main title is "Przykładowe rozwiązanie". It displays a JavaScript code snippet that uses the prompt function to get the user's name and writes a message to the document based on whether the name is null. The slide is branded with the DAILY GROUP logo at the bottom left.


Naciśnięcie przycisku anuluj w oknie wyświetlanym przez funkcję prompt zwraca jako wynik wartość null



Slajd 14



B2E
BUSINESS TO EDUCATION




SZCZECIŃSKI PARK
NAUKOWO-TECHNOLOGICZNY

JavaScript


Ćwiczenie

Napisz skrypt pobierający od użytkownika 2 liczby i wyświetlający na ekranie większą z nich




DAILY
GROUP

Slajd 15



B2E
BUSINESS TO EDUCATION




SZCZECIŃSKI PARK
NAUKOWO-TECHNOLOGICZNY

JavaScript

Przykładowe rozwiązanie

```
<script type="text/javascript">
<!--
var a = prompt("Podaj pierwszą liczbę: ");
var b = prompt("Podaj drugą liczbę: ")

if (a>b)
    document.write("Liczba " + a " jest większa od liczby " +b);
else if (a == b)
    document.write("Liczby " + a +" i "+b+" są równe");
else
    document.write("Liczba " + b " jest większa od liczby " +a);
//-->
</script>
```



DAILY
GROUP




Slajd 16

The slide features a header with logos for B2E (Business to Education), Szczeciński Park Naukowo-Technologiczny, and JavaScript. The main title is 'Ćwiczenie' (Exercise). The text of the exercise asks for a script that takes two numbers from a user and displays the result of their division, ensuring proper handling of division by zero. The slide also includes a 'DAILY GROUP' logo at the bottom left.


Celem ćwiczenia jest praktyczne zapoznanie się z instrukcjami warunkowymi. Do tego celu wykorzystamy algorytm obliczania równania kwadratowego $Ax^2 + Bx + C$. Równanie ma rozwiązanie, jeśli parametr Δ (delta) równy $B^2 - 4 \cdot A \cdot C$ jest większy od 0 lub mu równy. Jeśli Δ równa się 0, mamy jedno rozwiązanie równe $-B / (2 \cdot A)$, jeśli Δ jest większa od 0, mamy dwa rozwiązania: $x_1 = (-B + \sqrt{\Delta}) / (2 \cdot A)$ i $x_2 = (-B - \sqrt{\Delta}) / (2 \cdot A)$.



Slajd 17



B2E
BUSINESS TO EDUCATION




SZCZECIŃSKI PARK
NAUKOWO-TECHNOLOGICZNY

JavaScript

Przykładowe rozwiązanie


```
<script type="text/javascript">
<!--
var a = prompt("Podaj pierwszą liczbę: ");
var b = prompt("Podaj drugą liczbę: ")

if (b == 0) {
    document.write("Dzielenie przez 0");
}
else {
    document.write("Wynik dzielenia to: "+ (a/b));
}
//-->
</script>
```




DAILY
GROUP

Slajd 18



B2E
BUSINESS TO EDUCATION




SZCZECIŃSKI PARK
NAUKOWO-TECHNOLOGICZNY

JavaScript

Ćwiczenia



Napisz skrypt pobierający od użytkownika dzień tygodnia.
Jeśli podana zostanie sobota, albo niedziela wyświetl komunikat "Miłego weekendu!", w przeciwnym wypadku wyświetl komunikat "Miłej pracy"



DAILY
GROUP

Napisz skrypt pobierający od użytkownika dzień tygodnia.
Jeśli podana zostanie sobota, albo niedziela wyświetl komunikat "Miłego weekendu!",
w przeciwnym wypadku wyświetl komunikat "Miłej pracy"


Slajd 19



JavaScript



Przykładowe rozwiązanie

```
<script type="text/javascript">
<!--
var dzien = prompt("Podaj dzień tygodnia: ");
switch(dzien) {
  case "sobota" :
  case "niedziela":
    document.write("Miłego wypoczynku");
    break;
  default :
    document.write("Miłej pracy");
}
//-->
</script>
```



Instrukcja switch pozwala na grupowanie możliwych wariantów (pomijamy instrukcję break po poszczególnych opcjach)


Slajd 20



JavaScript

Ćwiczenia


Napisz skrypt wyświetlający rozwiązania równania kwadratowego


$$Ax^2 + Bx + C.$$


Celem ćwiczenia jest praktyczne zapoznanie się z instrukcjami warunkowymi. Do tego celu wykorzystamy algorytm obliczania równania kwadratowego $Ax^2 + Bx + C$.

Równanie ma rozwiązanie, jeśli parametr Δ (delta) równy $B^2-4*A*C$ jest większy od 0 lub mu równy. Jeśli Δ równa się 0, mamy jedno rozwiązanie równe $-B/(2*A)$, jeśli Δ jest większa od 0, mamy dwa rozwiązania: $x_1 = (-B+\sqrt{\Delta})/(2*A)$ i $x_2 = (-B-\sqrt{\Delta})/(2*A)$.

Slajd 21






SZCZECIŃSKI PARK
NAUKOWO-TECHNOLOGICZNY

JavaScript

Przykładowe rozwiązanie

```

var A = 1, B = 1, C = -2, str = "";
if (A == 0) {
  str += "To nie jest równanie kwadratowe: A = 0!";
} else {
  delta = B * B - 4 * A * C;
  if (delta < 0) {
    str += "Delta < 0<br>";
    str += "To równanie nie ma rozwiązania w zbiorze liczb rzeczywistych!";
  } else if (delta == 0) {
    wynik = - B / 2 * A;
    str += "Rozwiązanie: x = " + wynik;
  } else {
    wynik = (- B + Math.sqrt(delta)) / 2 * A; str += "Rozwiązanie: x1 = " + wynik;
    wynik = (- B - Math.sqrt(delta)) / 2 * A; str += ", x2 = " + wynik;
  }
}
var element = document.getElementById("content");
element.innerHTML = str;
                    
```



Celem ćwiczenia jest praktyczne zapoznanie się z instrukcjami warunkowymi. Do tego celu wykorzystamy algorytm obliczania równania kwadratowego $Ax^2 + Bx + C$. Równanie ma rozwiązanie, jeśli parametr Δ (delta) równy $B^2-4*A*C$ jest większy od 0 lub mu równy. Jeśli Δ równa się 0, mamy jedno rozwiązanie równe $-B/(2*A)$, jeśli Δ jest większa od 0, mamy dwa rozwiązania: $x_1 = (-B+\sqrt{\Delta})/(2*A)$ i $x_2 = (-B-\sqrt{\Delta})/(2*A)$.

Slajd 22





SZCZECIŃSKI PARK
NAUKOWO-TECHNOLOGICZNY

JavaScript

JavaScript

Pętle



Instrukcje warunkowe nie są jedynymi sposobami na kontrolę przepływu działania programu. Jeśli konieczne jest powtórzenie dowolnego fragmentu kodu to wykorzystać możemy pętle.

Slajd 23



B2E BUSINESS TO EDUCATION

SZCZECIŃSKI PARK NAUKOWO-TECHNOLOGICZNY

JavaScript

Rodzaje pętli w JavaScript

Pętle są instrukcjami pozwalającymi na wielokrotne wykonywanie tego samego kodu. JavaScript obsługuje następujące rodzaje pętli:

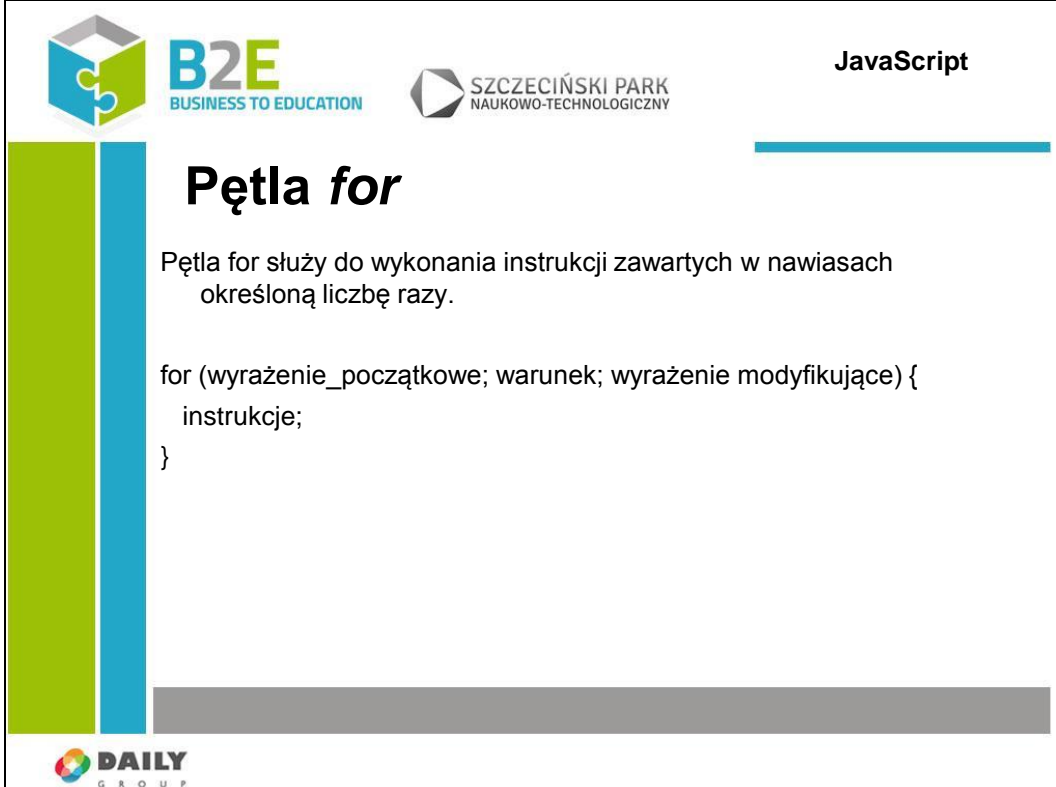
- for
- while
- do while

DAILY GROUP

Pętle są instrukcjami pozwalającymi na wielokrotne wykonywanie tego samego kodu. JavaScript obsługuje następujące rodzaje pętli:

- for
- while
- do while

Slajd 24



B2E BUSINESS TO EDUCATION

SZCZECIŃSKI PARK NAUKOWO-TECHNOLOGICZNY

JavaScript

Pętla for

Pętla for służy do wykonania instrukcji zawartych w nawiasach określoną liczbę razy.

```
for (wyrażenie_początkowe; warunek; wyrażenie modyfikujące) {
  instrukcje;
}
```

DAILY GROUP

Wyrażenie początkowe ($i = 0$) inicjuje zmienną używaną jako licznik pętli, warunek ($i < 5$) określa warunek którego spełnienie zakończy wykonywanie pętli, wyrażenie modyfikujące ($i++$) zwiększa lub zmniejsza licznik liczby wykonań).

Powyższy przykład oznacza, dla zmiennej i równej początkowo 0 sprawdź, czy jest mniejsza od 5 i jeśli tak to wypisz jej wartość na wyjście a następnie podnieś wartość zmiennej i o 1 i ponownie sprawdź warunek. Kontynuuj aż warunek przestanie być prawdziwy.

Każde wykonanie pętli nazywamy iteracją lub przebiegiem, a zmienną zawierającą licznik pętli - zmienną iteracyjną.



Slajd 25

JavaScript

Pętla *for*

Przykład

```
for (i = 0; i<5; i++) {  
    document.write(i + "<br />");  
}
```

DAILY GROUP

Dla zmiennej iteracyjnej *i* równej początkowo 0 sprawdź warunek – czy liczba jest mniejsza od 5. jeśli tak to wypisz na ekranie wartość tej liczby a następnie zwiększ ją o 1.

Slajd 26

JavaScript

Pętla *while*

Pętla *while* służy do wielokrotnego wykonywania powtarzających się fragmentów kodu (na etapie pisania funkcji nie wiemy ile razy kod zostanie wykonany).

```
while (warunek) {  
    instrukcje;  
}
```

DAILY GROUP

W przeciwieństwie do pętli *for* nie znamy z góry liczby iteracji w tej pętli. Pętla może wykonać dowolną ilość razy (dopóki warunek jest spełniony), instrukcje zawarte w pętli mogą również nie zostać wykonane, jeśli warunek nie został spełniony przed pierwszą



iteracją. Musimy też sami zadbać o zainicjalizowanie zmiennej iterującej i jej zwiększanie bądź zmniejszanie w każdej iteracji pętli.

Slajd 27

Slide 27 features a header with logos for B2E (Business to Education), Szczeciński Park Naukowo-Technologiczny, and the Daily Group. The title 'Pętla while' is prominently displayed. Below it, a code example shows a while loop that prints the value of 'i' from 0 to 4. The slide is decorated with a green and blue vertical bar on the left and a blue horizontal bar on the right.

JavaScript

Pętla *while*

Przykład

```
i = 0;
while (i <5) {
    document.writeln(i + "<br/>");
    i++;
}
```

DAILY GROUP

Do zmiennej 'i' przypisz wartość 0. Dopóki 'i' jest mniejsze od 5 wypisuj wartość 'i' na ekranie a następnie zwiększ i o 1.

Slajd 28

Slide 28 features the same header as slide 27. The title 'Pętla do ... while' is prominently displayed. Below it, a code example shows a do...while loop structure. The slide is decorated with a green and blue vertical bar on the left and a blue horizontal bar on the right.

JavaScript

Pętla *do ... while*

W pętli do ... while warunek sprawdzany jest po wykonaniu zestawu instrukcji.

```
do {
    instrukcje;
}
while (warunek);
```

DAILY GROUP



Specjalnym rodzajem pętli while jest pętla do while.
Różnicą w porównaniu do pętli while jest fakt, że instrukcje zawsze zostaną wykonane przynajmniej raz - nawet jeśli warunek początkowy nie jest spełniony.

Slajd 29

B2E BUSINESS TO EDUCATION

SZCZECIŃSKI PARK NAUKOWO-TECHNOLOGICZNY

JavaScript

Pętla do while

Przykład:

```
i = 0
do {
  document.writeln(i);
  i++;
} while (i<5)
```

DAILY GROUP

Na początku zmiennej "i" przypisz wartość "0". Wypisz wartość zmiennej "i" na ekranie i zwiększ ją o 1. Jeśli wartość zmiennej "i" jest mniejsza niż "5" wypisz jej wartość na ekranie i zwiększ o 1.

Slajd 30

B2E BUSINESS TO EDUCATION

SZCZECIŃSKI PARK NAUKOWO-TECHNOLOGICZNY

JavaScript

Zagnieżdżanie pętli

W języku JavaScript definicje pętli możemy umieszczać wewnątrz innych pętli. np.

```
for (i = 0; i < 5; i++) {
  document.writeln("jestem w petli
zewnetrznej" + i);
  for (j = 0; j<5; j++) {
    document.writeln("jestem w petli
wewnetrznej "+j);
  }
}
```

DAILY GROUP

Slajd 31

Zagnieżdżanie pętli

wynik:

Jestem w pętli zewnętrznej 1
Jestem w pętli wewnętrznej 1
Jestem w pętli wewnętrznej 2
Jestem w pętli zewnętrznej 2
Jestem w pętli wewnętrznej 1
Jestem w pętli wewnętrznej 2
Jestem w pętli zewnętrznej 3
Jestem w pętli wewnętrznej 1
Jestem w pętli wewnętrznej 2

Jak widzimy zagnieżdżanie pętli działa w następujący sposób:
 dla każdej iteracji pętli zewnętrznej wykonywane są wszystkie iteracje pętli wewnętrznej

Slajd 32

Przerwanie działania pętli

Do przerwania działania pętli wykorzystujemy 2 instrukcję:

- break – przerywa działanie całej pętli
- continue – przerywa działanie bieżącej iteracji

Do przerwania działania pętli wykorzystujemy 2 instrukcję:
 break – przerywa działanie całej pętli. Skrypt jest kontynuowany od kolejnej instrukcji poza pętlą.
 continue – przerywa działanie bieżącej iteracji. Przerwana jest bieżąca iteracja i uruchamiana jest kolejna. Należy pamiętać o ustawieniu zmiennej iteracyjnej (jeśli wykorzystujemy pętle while lub do).


Slajd 33

Ćwiczenie


Ćwiczenie 6
 Napisz skrypt wyświetlający na ekranie 20 razy tekst "Będę się pilnie uczył :)"



Slajd 34



B2E
BUSINESS TO EDUCATION




SZCZECIŃSKI PARK
NAUKOWO-TECHNOLOGICZNY

JavaScript


Przykładowe rozwiązania

```
i = 0;
while (i<20) {
    document.writeln("Będę się pilnie uczył");
    i++;
}
for (i=0; i<20; i++) {
    document.writeln("Będę się pilnie uczył");
}
i = 0;
do {
    document.write("Będę się pilnie uczył");
    i++;
} while (i<20)
```




DAILY
GROUP

Slajd 35



B2E
BUSINESS TO EDUCATION




SZCZECIŃSKI PARK
NAUKOWO-TECHNOLOGICZNY

JavaScript

Ćwiczenie


Napisz skrypt wyświetlający na ekranie tabliczkę mnożenia




DAILY
GROUP



Slajd 36



B2E
BUSINESS TO EDUCATION




SZCZECIŃSKI PARK
NAUKOWO-TECHNOLOGICZNY

JavaScript


Przykładowe rozwiązanie

```
for (i = 1; i <= 10; i++) {  
  for (j = 1; j <= 10; j++) {  
    document.write(i + " * " + j + " = " + i * j + "<br />");  
  }  
  document.write("<br />");  
}
```




DAILY
GROUP

Slajd 37



B2E
BUSINESS TO EDUCATION




SZCZECIŃSKI PARK
NAUKOWO-TECHNOLOGICZNY

JavaScript

Ćwiczenie


Napisz skrypt, który za pomocą pętli while i instrukcji continue wyświetli nieparzyste liczby z zakresu 10 – 30.




DAILY
GROUP



Slajd 38



B2E
BUSINESS TO EDUCATION



SZCZECIŃSKI PARK
NAUKOWO-TECHNOLOGICZNY


JavaScript

Przykładowe rozwiązanie

```

var str = "";
i = 10;
while(i < 30){
  if(i % 2 == 0){
    i++;
    continue;
  }
  str += i;
  str += " ";
  i++;
}
var element = document.getElementById("element");
element.innerHTML = str;

```



Rozwiązanie: w przypadku, gdy liczba jest parzysta (dzielenie modulo 2 daje wynik 0) pomijamy wypisywanie bieżącej wartości.

Slajd 39



B2E
BUSINESS TO EDUCATION



SZCZECIŃSKI PARK
NAUKOWO-TECHNOLOGICZNY

JavaScript

Dziękujemy za uwagę

Człowiek - najlepsza inwestycja



KAPITAŁ LUDZKI
NARODOWA STRATEGIA SPÓJNOŚCI



UNIA EUROPEJSKA
EUROPEJSKI
FUNDUSZ SPOŁECZNY

Projekt współfinansowany przez Unię Europejską w ramach Europejskiego Funduszu Społecznego



Ćwiczenia

Ćwiczenia zostały przedstawione na slajdach 12-21 oraz 33-38.

W pierwszym bloku ćwiczeń zwracana jest uwaga na wykorzystanie instrukcji warunkowych, drugi ćwiczenie umożliwia praktyczne rozpoznanie sposobu działania pętli – wypisujemy w nich na ekran m.in. wartości z tabliczki mnożenia.

Opis założonych osiągnięć ucznia

Uczestnik po zakończeniu tej lekcji zna instrukcje warunkowe i pętle i potrafi wykorzystać je w praktyce.

Lekcja 4 Tablice

Cel lekcji

Celem lekcji pokazanie, czym są i do czego można wykorzystywać zmienne tablicowe w języku JavaScript.

Sposoby osiągnięcia celów kształcenia

Czynności nauczyciela	Czynności ucznia
wprowadza nowe pojęcia związane z tematem zajęć (materiał ze slajdów) przygotowuje ćwiczenia z zaprezentowanego materiału	Wykonuje zadane przez nauczyciela ćwiczenia na komputerze
Prowadzi z uczniami dyskusję na temat wybranych przez nich rozwiązań ćwiczeń	Prezentuje rozwiązania ćwiczeń nauczycielowi i pozostałym uczniom, tłumaczy wybrane przez siebie rozwiązanie
Podsumowuje lekcję, ocenia pracę uczniów na zajęciach.	Dokonuje samooceny swojej pracy, omawia z nauczycielem rozwiązania ćwiczeń

Treść – slajdy z opisem

Slajd 1



W poprzednich lekcjach korzystaliśmy tylko z pojedynczych zmiennych. Często jednak potrzebujemy mechanizmu, który pozwoli nam na pogrupowanie zmiennych, albo nawet na zmienną do której możemy wpisać więcej niż jedną wartość. Do tego celu służą tablice.



Slajd 2

JavaScript

Tablice

struktury danych pozwalające na przechowywanie uporządkowanych elementów
pole danych w tabeli to *komórka*
każda komórka ma swój adres – nazywany indeksem
Indeksy komórek są numerowane od 0
pierwsza komórka ma adres 0, druga 1 itd

DAILY GROUP

W JavaScriptcie możemy zarządzać danymi w sposób bardziej zaawansowany niż za pomocą zmiennych. Jednym z przykładów są tablice – można powiedzieć, że są to kolekcje zmiennych uporządkowane i pogrupowane razem. Poszczególne elementy w tabeli nazywamy komórkami, adres każdej komórki indeksem. Prostą tablicę można sobie wyobrazić jako szafkę, w której każda szuflada jest komórką a numer tej szuflady indeksem komórki. Indeksy w tabelach JavaScript numerowane są od 0 – pierwsza komórka w tabeli ma indeks 0, druga 1 itd..

Slajd 3

JavaScript

Tablice

Tworzenie nazw tablic

- rozpoczyna się od małej lub wielkiej litery, znaku_ lub znaku \$
- nie może zawierać operatorów matematycznych (+ -), znaków punktacyjnych (takich jak !, &) albo spacji
- na drugim i dalszych miejscach może zawierać cyfry
- słowa kluczowe języka nie mogą być użyte jako nazwa

DAILY GROUP

Tworzenie nazw tablic rozpoczyna się od małej lub wielkiej litery, znaku_ lub znaku \$
nie może zawierać operatorów matematycznych (+ -), znaków punktacyjnych (takich jak !, &) albo spacji
na drugim i dalszych miejscach może zawierać cyfry
słowa kluczowe języka nie mogą być użyte jako nazwa

Slajd 4

Tworzenie tablic

Dwa sposoby

- przez deklaracje zmiennej w nawiasach kwadratowych
`var tablica = ["Janek", "Franek", "Broniek"];`
- przez wywołanie konstruktora tablicy
`var tablica = new Array("Janek", "Franek", "Broniek");`

Tablicę możemy utworzyć na dwa sposoby – za pomocą nawiasów kwadratowych podając wewnątrz nawiasów komórki tablicy, lub wywołując konstruktor obiektu Array, który jest reprezentacją tablicy w JavaScript (o obiektach i konstruktorach powiemy sobie później). Efektem działania obu instrukcji będzie zmienna tablica zawierająca 3 imiona – Janek, Franek i Broniek.

Slajd 5

Przykłady tworzenia tablic

```
var tablica = [1,2,3,4,5,6,7,8,9,10];
var tablica = [1, "Janek", 2, "Franek"];
var tablica = ["Janek", "Franek", "Broniek"];
var tablica = [];
```

Wartość każdej komórki może być dowolnego typu.

Przykładowo możemy utworzyć tablicę zawierającą liczby całkowite od 1 do 10, wcześniej widzieliśmy przykład deklaracji tablicy zawierającej ciągi znaków. W tablicy możemy też umieścić elementy różnego typu – np. liczby i ciągi znaków. Notacja z wykorzystaniem nawiasów kwadratowych umożliwia pomijanie elementów w tablicy – komórki pominięte będą miały wartość undefined. Możemy także utworzyć pustą tablicę i dodawać do niej elementy w trakcie działania programu.



Slajd 6

JavaScript

Przykłady tworzenia tablic

```
var tablica = new Array();
var tablica = new Array(1,2,3,4,5,6,7,8,9,10);
var tablica = new Array(5);
```

DAILY GROUP

W konstruktorze tablicy możemy nie podawać żadnych argumentów – zostanie wtedy utworzona pusta tablica, możemy podać argumenty oddzielone przecinkami – zostaną one wtedy wpisane jako wartości komórek tablicy.

Trzecia przykładowa konstrukcja ma trochę inne działanie. Zamiast spodziewanego utworzenia tablicy z jedną komórką o wartości 5, zostanie utworzona tablica pięcioelementowa w której komórki będą miały wartość undefined

Slajd 7

JavaScript

Odczyt i zapis danych

```
var zmienna = tablica[5];
document.write(tablica[5]);
if (tablica[5] == wartość) {
    instrukcje;
}
tablica[5] = wartość;
var indeks = 6;
tablica[indeks] = nowa_wartość;
```



DAILY GROUP

Odczyt danych z tablicy jest prosty. Aby przypisać wartość z danej komórki tablicy wystarczy podać jej indeks. Nie musimy oczywiście korzystać z dodatkowych zmiennych aby wyświetlić lub wykorzystać wartość komórki. Możliwe jest bezpośrednie operowanie na komórkach tabeli. Aby zapisać wartość do komórki tablicy wykonujemy operację przypisania na tablicy z wykorzystaniem indeksu komórki, którą chcemy zmodyfikować.

Wartości indeksu nie musimy podawać w trakcie pisania kodu – możemy wykorzystać do tego zmienną i ustawić wartość w trakcie działania programu.

Możemy jako indeks tablicy wykorzystać komórkę innej tablicy

Slajd 8




JavaScript

Tablice asocjacyjne

Rozważmy taką tablicę:

```
var wypłaty = new Array("Janek", 2000, "Franek", 3000, "Bronek", 2500);  
  
var wypłaty = new Array();  
wypłaty["Janek"] = 2000;  
wypłaty["Franek"] = 3000;  
wypłaty["Bronek"] = 2500;
```




Używanie indeksów numerycznych jest dobre, jeśli operujemy na niewielkiej liczbie danych. Jeśli danych jest więcej lub jeśli dane mają dodatkowe znaczenie, użycie numerów aby pobrać wartość z tabeli zaczyna robić się kłopotliwe.

Rozważmy tablicę z wypłatami w następującej postaci: w pierwszej komórce jest identyfikator osoby (imię) a w kolejnej jego zarobki. Możliwe jest wyszukanie zarobków Franka w następujący sposób – sprawdź, czy wartość pierwszej komórki to „Franek” – jeśli tak to pobierz i wyświetl wartość kolejnej komórki. W przeciwnym wypadku przeskocz o 2 komórki i wykonaj sprawdzenie ponownie.

Na szczęście javascript wspiera przypisywanie nazw elementom komórek. Możemy zatem utworzyć następującą tablicę, gdzie jako indeksy pól podamy identyfikatory osób a jako wartość ich wypłatę. Wyświetlenie dochodów Franka to w tym przypadku proste zapytanie `wypłaty[„Franek”]`.

JavaScript nie tworzy w tym przypadku nowych elementów tablicy tylko dodaje nowe właściwości (pary typu klucz – wartość) do obiektu przechowującego tablicę. Zapytanie o długość tablicy zwróci w tym przypadku 0 – bo utworzyliśmy pustą tablicę i dodaliśmy do niej 3 właściwości

Slajd 9



B2E BUSINESS TO EDUCATION

SZCZECIŃSKI PARK NAUKOWO-TECHNOLOGICZNY

JavaScript

Tablice asocjacyjne

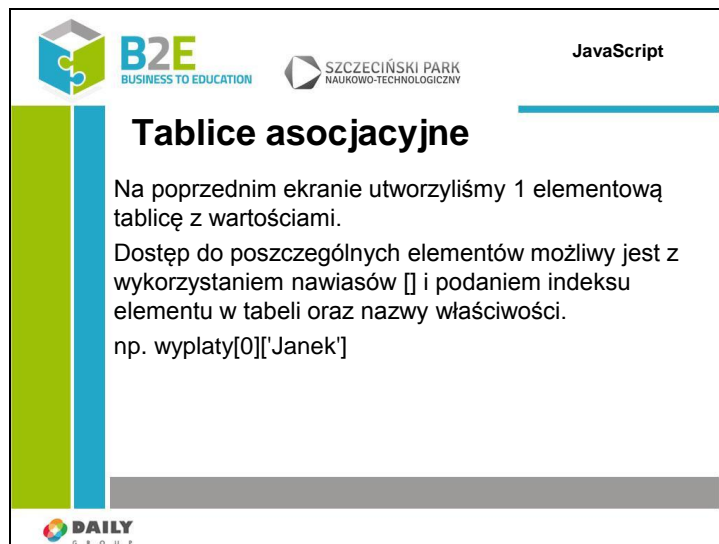
Tworzenie tablicy asocjacyjnej z wypełnieniem danymi

```
var wypłaty = new Array(
{
'Janek' : 2000,
'Franek' : 3000,
'Broniek' : 2500
})
```

DAILY GROUP

Tworząc tabelę asocjacyjną zainicjowaną początkowym zestawem danych do konstruktora obiektu przekazujemy definicję wartości, które chcemy utworzyć. Definicja zawiera pary klucz wartość (oddzielone dwukropkiem, a nie znakiem równości) umieszczone w nawiasach klamrowych

Slajd 10



B2E BUSINESS TO EDUCATION

SZCZECIŃSKI PARK NAUKOWO-TECHNOLOGICZNY

JavaScript

Tablice asocjacyjne

Na poprzednim ekranie utworzyliśmy 1 elementową tablicę z wartościami.

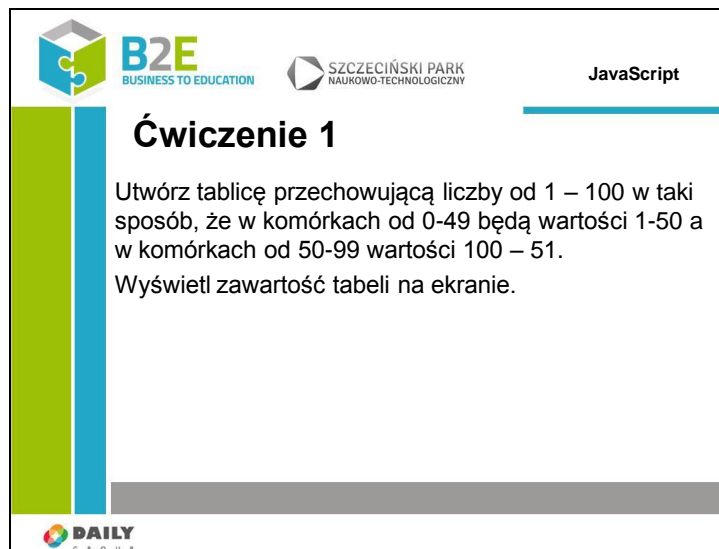
Dostęp do poszczególnych elementów możliwy jest z wykorzystaniem nawiasów [] i podaniem indeksu elementu w tabeli oraz nazwy właściwości.

np. wypłaty[0]['Janek']

DAILY GROUP

Na poprzednim ekranie utworzyliśmy 1 elementową tablicę z wartościami. Dostęp do poszczególnych elementów możliwy jest z wykorzystaniem nawiasów [] i podaniem indeksu elementu w tabeli oraz nazwy właściwości. np. wypłaty[0]['Janek'], gdzie 0 oznacza 1 element w tablicy.

Slajd 11



B2E BUSINESS TO EDUCATION

SZCZECIŃSKI PARK NAUKOWO-TECHNOLOGICZNY

JavaScript

Ćwiczenie 1

Utwórz tablicę przechowującą liczby od 1 – 100 w taki sposób, że w komórkach od 0-49 będą wartości 1-50 a w komórkach od 50-99 wartości 100 – 51.


Wyświetl zawartość tabeli na ekranie.

DAILY GROUP


Ćwiczenie: Utwórz tablicę przechowującą liczby od 1 – 100 w taki sposób, że w komórkach od 0-49 będą wartości 1-50 a w komórkach od 50-99 wartości 100 – 51. Wyświetl zawartość tabeli na ekranie.



Slajd 12



B2E
BUSINESS TO EDUCATION




SZCZECIŃSKI PARK
NAUKOWO-TECHNOLOGICZNY

JavaScript

Ćwiczenie 1

```
for (i = 0; i<100; i++) {  
  if (i<50) {  
    tablica[i] = i+1;  
  }  
  else {  
    tablica[i] = 100 - (i - 50);  
  }  
}  
for(i = 0; i<100; i++) {  
  document.write(tablica[i] + "<br />");  
}
```



Slajd 13



B2E
BUSINESS TO EDUCATION



SZCZECIŃSKI PARK
NAUKOWO-TECHNOLOGICZNY


JavaScript


JavaScript

Tablice wielowymiarowe



Slajd 14





SZCZECIŃSKI PARK
NAUKOWO-TECHNOLOGICZNY

JavaScript

Tablice wielowymiarowe


```

var tablica = [1,2,3];
tablica[x]

var tablica = [
  ["A1", "B1", "C1"],
  ["A2", "B2", "C2"]
];
tablica[x][y]
```

1	2	3
---	---	---

A1	B1	C1
A2	B2	C2



Do tej pory wszystkie pokazane tablice miały tylko 1 wymiar. Można je było przedstawić za pomocą wektora z wartościami komórek tabeli.


Do pojedynczej wartości można było dostać się podając jej indeks.


Czy możliwe jest zatem podanie więcej niż jednego indeksu, aby wskazać element tablicy? JavaScript nie wspiera niestety wielowymiarowych tablic, ale jest sposób na zasymulowanie takiej funkcjonalności.

W języku JavaScript wartością komórek tablicy mogą być inne tablice. Możemy utworzyć tabele zawierające w komórkach inne tabele, które mogą zawierać kolejne tabele itd.

Dostęp do elementów w takich tabelach jest następujący: `tablica[x][y]` – Z tablicy pobierz wartość komórki oznaczonej indeksem x a z niej wartość komórki oznaczonej Y

Slajd 15





SZCZECIŃSKI PARK
NAUKOWO-TECHNOLOGICZNY


JavaScript

Użycie pętli for

```

var tablica = [];
for (var i = 0; i < 100; i++) {
  tablica[i] = "wartość "+i;
}

var dlugosc_tablicy = tablica.length;
for (var i = 0; i < dlugosc_tablicy; i++) {
  document.write(tablica[i] + "<br />");
}
```



Bezpośrednie odwołania do elementów tablicy za pomocą indeksu są wystarczające tylko w przypadku, gdy tablica zawiera niewiele elementów. Jeśli danych jest dużo, lub z góry nie wiemy ile (częsta sytuacja, gdy nasz program otrzyma dane z zewnątrz) wymagane będzie wprowadzenie automatycznego przetwarzania tablicy. W takim celu możemy wykorzystać pętle. Pierwsza przedstawiona pętla dodaje 100 elementów do nowo utworzonej tablicy. Odczyt danych z tablicy możliwy jest w następujący sposób.

Pętla for może zostać wykorzystana w przypadku gdy znamy długość tablicy. W przypadku tablic z indeksem numerycznym długość tablicy możemy pobrać za pomocą właściwości `length`. Właściwość ta przyjmuje wartość najwyższego indeksu w tablicy + 1.

Tworząc pętlę warto obliczyć długość tablicy przed wywołaniem pętli. Przypisując liczbę elementów w tablicy do zmiennej operacja obliczenia wartości właściwości `tablica.length` wykonywana jest tylko raz, gdybyśmy w definicji pętli ustawili warunek `i < tablica.length` długość tablicy sprawdzana byłaby za każdą iteracją.

Slajd 16

JavaScript

Pętla for in

```
for (var indeks in tablica) {
    document.write(tablica[indeks] +
"<br/>");
}
```

Jeśli nie znamy długości tablicy, lub jeśli wiemy że zawiera ona komórki niezdefiniowane możemy wykorzystać pętlę specjalną `for in`. Definicja pętli jest następująca:

```
for (var indeks in tablica) {
    instrukcje
}
```

Wynikiem działania funkcji jest przejście po wszystkich zdefiniowanych indeksach (funkcja wyświetli tylko komórki mające wartość) oraz dodanych do obiektu właściwościach.

Slajd 17

JavaScript

Operacje na tablicach

Tablice są obiektami w JavaScript, zawierającymi kilka predefiniowanych metod umożliwiających m.in.:

- Łączenie tablic (`concat`)
- Odwrócenie kolejności elementów (`reverse`)
- Dodawanie i usuwanie elementów (`pop`, `push`, `shift`, `unshift`)
- sortowanie (`sort`)
- sklejanie (`join`)

Tablice są obiektami w JavaScript, zawierającymi kilka predefiniowanych metod umożliwiających m.in.:

-`concat`. Tablice zostaną dodane do siebie a powstają wynik zwrócony w wyniku działania funkcji

-`reverse` – odwraca kolejność elementów w tablicy

-`pop` – pobiera i usuwa z tablicy ostatni element

-`push` - dodaje na końcu tablicy nowy element

- shift – pobiera i usuwa pierwszy element z tablicy
- unshift – wstawia element na początku tablicy (odwrotność shift)
- sort – sortuje tablicę. Jako element należy podać nazwę funkcji porównującej 2 elementy
- join – zwraca łańcuch znaków z zawartością wszystkich komórek oddzielonych znakiem separatora

Slajd 18

JavaScript

Łączenie tablic

```

tablica2 = tablica.concat( lista elementów)

var dni_tygodnia = ["Poniedziałek", "Wtorek", "Środa",
"Czwartek", "Piątek", "Sobota", "Niedziela"];
var miesiace = ["Styczeń", "Luty", "Marzec", "Kwiecień"];
var wynik = dni_tygodnia.concat(miesiace);
    
```

DAILY GROUP

Wynikiem połączenia tablic jest nowa tablica składająca się z wszystkich elementów tablicy dni_tygodnia oraz wszystkich elementów tablicy miesiace.

Slajd 19

JavaScript

Odwracanie tablic

```

var tablica = [1,2,3,4,5];
tablica.reverse();
wynik: [5,4,3,2,1]
    
```

DAILY GROUP

Do odwracania tablic służy funkcja reverse. Nie przyjmuje ona żadnych argumentów i w działa na tablicy, na rzecz której została wywołana (nie jest tworzona tablica wynikowa).

Slajd 20

Dodawanie i usuwanie elementów

- pop - `tablica.pop()`
- push – `tablica.push(lista elementów)`
- shift – `tablica.shift()`
- unshift – `tablica.unshift(lista elementów)`

Slajd 21

Sortowanie tablicy

Aby posortować tablicę należy wywołać na niej funkcję `sort()`

```
var liczby = [5,7,3,5,2,7,5];
liczby.sort();
wynik: [2,3,5,5,5,7,7]
```

Sortowanie tablicy to uporządkowanie jej elementów. Funkcja zmienia porządek elementów w tablicy nadpisując oryginalną wersję. Tablice zawierające liczby zostaną uporządkowane rosnąco, natomiast tablice zawierające ciągi znaków – alfabetycznie.

Slajd 22

Sortowanie cd.

```
tablica.sort(nazwa_funkcji);
```

Funkcja musi zwracać:

- wartość mniejszą od 0, jeśli pierwszy argument jest mniejszy od drugiego
- wartość 0, jeśli argumenty są równe,
- wartość większą od 0, jeśli pierwszy argument jest większy od drugiego

Jeśli będziemy chcieli posortować tablice w sposób odmienny od domyślnego do wywołania metody `sort`, należy przekazać nazwę funkcji porównującej 2 elementy. Taka funkcja będzie otrzymywać 2 elementy tablicy, jako wynik musi zwracać:

- wartość mniejszą od 0, jeżeli pierwszy argument jest mniejszy od drugiego
 - wartość równą 0, jeżeli argumenty są równe
 - wartość większą od 0, jeżeli pierwszy argument jest większy od drugiego
- Aby odwrócić kolejność sortowania możemy albo zmodyfikować funkcję porównującą, albo na wyniku sortowania zastosować metodę reverse()

Slajd 23

JavaScript

Sklejanie wartości (join)

```
var tablica = ["Janek", "Franek", "Broniek"];
document.write(tablica.join(" i "));
```

wynik: Janek i Franek i Broniek
bez separatora: Janek,Franek,Broniek

Czasem przydatna jest możliwość zmiany wszystkich elementów tablicy w jeden łańcuch znaków. Można do tego wykorzystać metodę join() jako argument podając separator rozdzielający poszczególne elementy tablicy. W podanym przykładzie jako separator wykorzystaliśmy łańcuch składający się ze znaku spacji na początku i końcu oraz spójnika i. Wynikiem działania skryptu będzie tekst: Janek i Franek i Broniek. Jeśli nie podamy żadnego separatora domyślnie zostanie przyjęty znak,

Slajd 24

JavaScript

Dzielenie ciągu na elementy



```
var tekst= "Janek i Franek i Broniek";
document.write(tekst.split(" i "));
```

wynik: Janek,Franek,Broniek

Odwrotną metodą jest metoda split() – operująca na ciągu znaków. Wynikiem działania jest tablica elementów powstałych z podzielenia ciągu wejściowego na fragmenty oddzielone znakiem separatora.




Slajd 25



JavaScript

Ćwiczenie

Utwórz tablicę przechowującą 50 kolejnych wartości całkowitych i wyświetl jej zawartość na ekranie. W obu operacjach użyj pętli typu while.




Slajd 26

JavaScript


Przykładowe rozwiązanie

```
var tablica = new Array();  
i = 0;  
while (i<50) {  
  tablica[i] = i+1;  
  i++;  
}  
j = 0;  
while (j<tablica.length) {  
  document.writeln(tablica[j]);  
  j++;  
}
```





Slajd 27



B2E
BUSINESS TO EDUCATION




SZCZECIŃSKI PARK
NAUKOWO-TECHNOLOGICZNY

JavaScript


Ćwiczenia

Wykonaj sortowanie tablicy zawierającej wartości całkowite, tak aby najpierw znalazły się wartości niepodzielne przez 3 w porządku malejącym, a następnie wartości podzielne przez 3 w porządku rosnącym.




DAILY
GROUP

Slajd 28



B2E
BUSINESS TO EDUCATION




SZCZECIŃSKI PARK
NAUKOWO-TECHNOLOGICZNY

JavaScript

Przykładowe rozwiązanie

```
function porownaj(e1, e2) {  
  if(e1 % 3 != 0){  
    if(e2 % 3 != 0){  
      return e2 - e1;  
    } else {  
      return -1;  
    }  
  } else{  
    if(e2 % 3 != 0){  
      return 1;  
    } else {  
      return e1 - e2;  
    }  
  }  
}
```


```
var str = "";  
tab1 = Array(5, 7, 3, 1, 8, 2, 0, 4, 9, 6);
```




DAILY
GROUP



Slajd 29



B2E
BUSINESS TO EDUCATION




SZCZECIŃSKI PARK
NAUKOWO-TECHNOLOGICZNY


JavaScript

Przykładowe rozwiązanie


```
var str = "";  
var tab1 = Array(5, 7, 3, 1, 8, 2, 0, 4, 9, 6);  
  
str += "Zawartość tablicy przed sortowaniem: <br />";  
for(var i in tab1) str += tab1[i] + " ";  
  
tab1.sort(porownaj);  
  
str += "<br /><br />Zawartość tablicy po sortowaniu: <br />";  
for(var i in tab1) str += tab1[i] + " ";  
  
var element = document.getElementById("content");  
element.innerHTML = str;
```



Slajd 30



B2E
BUSINESS TO EDUCATION




SZCZECIŃSKI PARK
NAUKOWO-TECHNOLOGICZNY

JavaScript


Ćwiczenia

Z tablicy zawierającej dni tygodnia przesuń niedzielę na początek.






Slajd 31



B2E
BUSINESS TO EDUCATION



SZCZECIŃSKI PARK
NAUKOWO-TECHNOLOGICZNY

JavaScript


Przykładowe rozwiązanie

```
var dni = ["Poniedziałek", "Wtorek", "Środa", "Czwartek", "Piątek", "Sobota", "Niedziela"];
var niedziela = dni.pop();
dni.unshift(niedziela);
```



KAPITAŁ LUDZKI
NARODOWA STRATEGIA SPOJNOŚCI

Człowiek - najlepsza inwestycja



UNIA EUROPEJSKA
EUROPEJSKI
FUNDUSZ SPOŁECZNY

Projekt współfinansowany przez Unię Europejską w ramach Europejskiego Funduszu Społecznego



DAILY
GROUP

Ćwiczenia

Ćwiczenie zostało zaprezentowane na slajdzie 10 oraz na slajdach 25 - 31.

Opis założonych osiągnięć ucznia

Po ukończeniu lekcji uczestnik będzie znał sposoby tworzenia tablic w języku JavaScript, a także podstawowe metody służące do pracy z tablicami

Lekcja 5 Obiekty i funkcje

Cel lekcji

Język JavaScript jest językiem obiektowym – w tej lekcji poznamy definicję obiektu oraz funkcji.

Sposoby osiągnięcia celów kształcenia

Czynności nauczyciela	Czynności ucznia
wprowadza nowe pojęcia związane z tematem zajęć (materiał ze slajdów) przygotowuje ćwiczenia z zaprezentowanego materiału	Wykonuje zadane przez nauczyciela ćwiczenia na komputerze
Prowadzi z uczniami dyskusję na temat wybranych przez nich rozwiązań ćwiczeń	Prezentuje rozwiązania ćwiczeń nauczycielowi i pozostałym uczniom, tłumaczy wybrane przez siebie rozwiązanie
Podsumowuje lekcję, ocenia pracę uczniów na zajęciach.	Dokonyuje samooceny swojej pracy, omawia z nauczycielem rozwiązania ćwiczeń

Treść – slajdy z opisem

Slajd 1

We wcześniejszych lekcjach wprowadziliśmy pojęcia zmiennej, pokazaliśmy istniejące typy danych w języku JavaScript i omówiliśmy tablice jako specjalny rodzaj zmiennych. JavaScript pozwala nam na tworzenie własnych typów danych zwanych obiektami i zbiorów instrukcji, które mogą wykonywać jakąś część programu zwanych funkcjami. W tej lekcji przybliżymy te pojęcia.

Slajd 2

Funkcja to jeden z podstawowych bloków w JavaScript. Funkcja jest zestawem instrukcji, które wykonują określone zadanie lub obliczają wartość. Funkcja wykonuje zdefiniowane w ciele funkcji instrukcje a następnie zwraca wynik do miejsca, z którego została wywołana. Aby wykorzystywać funkcje trzeba ją wcześniej zdefiniować.

Slajd 3

Definicja funkcji (zwana też deklaracją funkcji) składa się ze słowa kluczowego function i następujących elementów:

- nazwy funkcji
- listy argumentów w nawiasach, oddzielonych przecinkami)

- instrukcji definiujących funkcję umieszczonych w nawiasach klamrowych. Nazwa może się składać z liter, cyfr oraz znaków podkreślenia i dolara, nie może jednak zaczynać się od cyfry. We wszystkich współczesnych implementacjach JavaScriptu może też zawierać znaki spoza alfabetu łacińskiego (np. polskie znaki diakrytyczne), choć z reguły się ich nie stosuje. Funkcja może, ale nie musi zawierać słowa kluczowego return (zwracającego wartość funkcji). Gdy w definicji funkcji brakuje instrukcji return wartością zwracaną przez funkcję jest "undefined"

Slajd 4

Definiowanie funkcji

```
function nazwa(argument1, argument2, ... , argumentN)
{
  //instrukcje
}

function kwadrat(liczba) {
  return liczba * liczba;
}
```

Funkcja posiada 1 argument nazwany liczba i zwraca za pomocą instrukcji return wartość argumentu pomnożonego przez samego siebie. Typy podstawowe (np, number) są przekazywane do funkcji za pomocą wartości. Oznacza to, że jeśli funkcja zmodyfikuje wartość parametru to zmiana ta nie będzie widoczna poza funkcją, przekazanie do funkcji typu złożonego (np. tablicy, lub zdefiniowanego przez użytkownika obiektu) spowoduje przekazanie tylko referencji do obiektu. Wszystkie modyfikacje wykonane przez funkcję będą widoczne także poza nią.

Slajd 5

Argumenty funkcji

- brak z góry zdefiniowanej liczby
- służą do przekazania wartości do funkcji w celu dowolnego przetwarzania i wykonywania na nich instrukcji

Argumenty funkcji

- brak z góry zdefiniowanej liczby
- służą do przekazania wartości do funkcji w celu dowolnego przetwarzania i wykonywania na nich instrukcji

Slajd 6

JavaScript

Zwracanie wartości przez funkcję

- słowo kluczowe **return**

function nazwa (argumenty)
{
 //instrukcje
 return wartość;
}

DAILY GROUP

Jeśli w ciele funkcji wystąpi instrukcja return to przetwarzanie funkcji zostaje przerwane i do miejsca wywołania zwracana jest wartość występująca po słowie return.

Slajd 7

JavaScript

Zasięg (widoczność) zmiennych

Zasięg zmiennej to miejsce w którym zmienna jest ważna i można się do niej bezpośrednio odwoływać.

Zmienne mogą być:

- globalne,
- lokalne

DAILY GROUP

Kiedy zadeklarujemy zmienną poza ciałem funkcji jest ona nazywana zmienną globalną, ponieważ jest dostępna z poziomu każdego innego fragmentu kodu w bieżącym dokumencie,

Jeśli zadeklarujemy zmienną w ciele funkcji jest ona nazywana zmienną lokalną - widoczną tylko w obrębie tej funkcji (i ewentualnych podfunkcji w niej zdefiniowanych)

Aby zadeklarować zmienną lokalną należy poprzedzić ją słowem kluczowym var. Wartość takiej zmiennej zostanie zapomniana po opuszczeniu funkcji, a jeśli istniała wcześniej zmienna globalna o takiej samej nazwie, zachowa ona swoją wartość sprzed wywołania funkcji.

Zmienna zadeklarowana w funkcji bez wykorzystania słowa kluczowego var jest zmienną globalną.



Slajd 8

JavaScript

JavaScript

Funkcje predefiniowane

B2E BUSINESS TO EDUCATION | SZCZECIŃSKI PARK NAUKOWO-TECHNOLOGICZNY

DAILY GROUP

Slajd 9

JavaScript

Funkcje predefiniowane

JavaScript zawiera kilka predefiniowanych funkcji o zasięgu globalnym

- eval
- isNaN
- parseInt oraz parseFloat
- Number i String

B2E BUSINESS TO EDUCATION | SZCZECIŃSKI PARK NAUKOWO-TECHNOLOGICZNY

DAILY GROUP

JavaScript zawiera kilka predefiniowanych funkcji o zasięgu globalnym

- eval
- isNaN
- parseInt oraz parseFloat
- Number i String

Slajd 10

JavaScript

Funkcja eval

Funkcja eval przekazuje łańcuch znaków do kompilatora i wykonuje rezultat

eval(wyrażenie);

B2E BUSINESS TO EDUCATION | SZCZECIŃSKI PARK NAUKOWO-TECHNOLOGICZNY

DAILY GROUP

Funkcja eval przekazuje łańcuch znaków do kompilatora i wykonuje rezultat eval(wyrażenie); Nie zaleca się nadmiernie wykorzystywać tej funkcji, ponieważ utrudnia, a czasem nawet uniemożliwia sprawdzanie poprawności kodu z wykorzystaniem narzędzi typu Jslint. Dodatkowo funkcja eval zużywa czas na przekształcenie wyrażenia.



Slajd 11

JavaScript

Funkcja eval - przykłady

```
eval("x=10;y=20;document.write(x*y)");  
document.write("<br>" + eval("2+2"));  
document.write("<br>" + eval(x+17));
```

wynikiem będą:

200
4
27

DAILY GROUP

```
eval("x=10;y=20;document.write(x*y)");  
document.write("<br>" + eval("2+2"));  
document.write("<br>" + eval(x+17));
```

wynikiem będą:

200
4
27

Slajd 12

JavaScript

Funkcja isNaN

Sprawdza czy wartość jest liczbą.
isNaN(wartość);



Zwraca true jeśli przekazana wartość nie jest liczbą

DAILY GROUP

Funkcje parseFloat i parseInt zwracają NaN, gdy podana jako argument wartość nie jest liczbą. Wykorzystując funkcję sprawdzającą isNaN możemy w kodzie programu sprawdzić czy konwersja się udała i w zależności od wyniku odpowiednio zareagować.



Slajd 13




JavaScript

Funkcja isNaN - przykłady



```
document.write(isNaN(5-2)+ "<br>");
document.write(isNaN(0)+ "<br>");
document.write(isNaN("Hello")+ "<br>");
document.write(isNaN("2005/12/12")+ "<br>");
```

wynikiem będzie:

```
false
false
true
true
```




Slajd 14



JavaScript

Funkcje parseInt i parseFloat

Funkcje konwertujące łańcuch znaków na liczby całkowite i zmiennoprzecinkowe.

```
parseFloat(str);
parseInt(str [, base]);
```




Funkcja `parseFloat` próbuje zwrócić liczbę zmiennoprzecinkową z podanego łańcucha znaków. Jeśli natrafi na znak inny niż `+` lub `-`, cyfrę(0-9) lub `.` zwraca wartość uzyskaną do danego momentu i ignoruje resztę ciągu. Jeśli pierwszy znak w łańcuchu nie może być skonwertowany do liczby funkcja zwraca `NaN`.

Funkcja `parseInt` konwertuje pierwszy argument (`str`) na liczbę całkowitą zgodną z wybraną bazą. Np. baza 10 zwróci liczbę jako dziesiętną 8 jako ósemkową, 16 - szesnastkową itd.


Jeśli podczas przetwarzania funkcja trafi na znak nie znajdujący się w podanej bazie przerwie wykonywanie i zwróci wartość uzyskaną do tego momentu. Jeśli pierwszy znak w ciągu nie może być zmieniony na liczbę funkcja zwraca `NaN`



Slajd 15



B2E
BUSINESS TO EDUCATION




SZCZECIŃSKI PARK
NAUKOWO-TECHNOLOGICZNY

JavaScript


parseFloat - przykłady

```
document.write(parseFloat("10") + "<br>");  
document.write(parseFloat("10.33") + "<br>");  
document.write(parseFloat("34 45 66") + "<br>");  
document.write(parseFloat(" 60 ") + "<br>");  
document.write(parseFloat("40 lat") + "<br>");  
document.write(parseFloat("On ma 40 lat") + "<br>");
```




DAILY
GROUP

Slajd 16



B2E
BUSINESS TO EDUCATION




SZCZECIŃSKI PARK
NAUKOWO-TECHNOLOGICZNY

JavaScript

parseFloat - przykłady cd.

wynikiem będą:


```
10  
10.33  
34  
60  
40 (!!!!)  
NaN
```




DAILY
GROUP



Slajd 17






SZCZECIŃSKI PARK
NAUKOWO-TECHNOLOGICZNY

JavaScript

parseFloat - przykłady

```

document.write(parseFloat("10") + "<br>");
document.write(parseFloat("10.33") + "<br>");
document.write(parseFloat("34 45 66") + "<br>");
document.write(parseFloat("40 lat") + "<br>");
document.write(parseFloat("On ma 40 lat") + "<br>");
document.write(parseFloat("10",10)+ "<br>");
document.write(parseFloat("010")+ "<br>");
document.write(parseFloat("10",8)+ "<br>");
document.write(parseFloat("0x10")+ "<br>");
document.write(parseFloat("10",16)+ "<br>");
    
```



Tylko pierwsza liczba w łańcuchu jest zwracana,

spacje na początku i końcu wyrażenia są dopuszczalne

Starsze przeglądarki ustawiają domyślną bazę na ósemkową jeśli łańcuch będzie rozpoczynał się od 0.

W wersji ECMAScript 5 domyślną bazą są liczby dziesiętne

Slajd 18





SZCZECIŃSKI PARK
NAUKOWO-TECHNOLOGICZNY

JavaScript

parseFloat - przykłady cd.


wyniki:


```

10
10
34
40 (!!!)
NaN
10
8 (!!!)
8
16
16
    
```



Slajd 19

 **B2E**
BUSINESS TO EDUCATION


 SZCZECIŃSKI PARK
NAUKOWO-TECHNOLOGICZNY

JavaScript

Funkcja Number i String


Funkcje pozwalające na konwersję obiektu do liczby lub łańcucha znaków

```
var obiekt;  
obiekt = Number(obiekt);  
obiekt = String(obiekt);
```

 DAILY
GROUP

Funkcja Number wykorzystuje metodę `valueOf()` obiektu a funkcja String metodę `toString()`.

Slajd 20

 **B2E**
BUSINESS TO EDUCATION

 SZCZECIŃSKI PARK
NAUKOWO-TECHNOLOGICZNY

JavaScript

Ćwiczenia


Ćwiczenie 1

Napisz funkcję, której 2 pierwsze argumenty będą operandami, a trzeci argument będzie oznaczał rodzaj działania. W zależności od rodzaju działania funkcja powinna zwracać sumę, różnicę, iloraz lub iloczyn dwóch pierwszych argumentów


 DAILY
GROUP



Slajd 21



B2E
BUSINESS TO EDUCATION




SZCZECIŃSKI PARK
NAUKOWO-TECHNOLOGICZNY


JavaScript

Przykładowe rozwiązanie


```
function math(a,b, op) {  
  switch (op) {  
    case "+": return a+b; break;  
    case "-": return a-b; break;  
    case "*": return a*b; break;  
    case "/": return a/b; break;  
    default: return "Nieznana operacja";break; }  
}  
  
var a = 5, b = 10;  
document.writeln(math(a,b, "+"));  
document.writeln(math(a,b, "-"));  
document.writeln(math(a,b, "*"));  
document.writeln(math(a,b, "/"));  
document.writeln(math(a,b, "^"));
```



Slajd 22



B2E
BUSINESS TO EDUCATION




SZCZECIŃSKI PARK
NAUKOWO-TECHNOLOGICZNY

JavaScript


Ćwiczenie

Napisz funkcję, która przyjmuje 2 parametry liczbowe i zwraca większy z nich.






Slajd 23



B2E
BUSINESS TO EDUCATION




SZCZECIŃSKI PARK
NAUKOWO-TECHNOLOGICZNY


JavaScript

Przykładowe rozwiązanie


```
function wiekszy(a,b) {  
  if (a > b) return a;  
  else return b;  
}  
  
var a = 10;  
var b = 20;  
document.writeln("Wiekszy z pary " + a + " i "+ b + " to "+wiekszy(a,b));
```



Slajd 24



B2E
BUSINESS TO EDUCATION




SZCZECIŃSKI PARK
NAUKOWO-TECHNOLOGICZNY

JavaScript

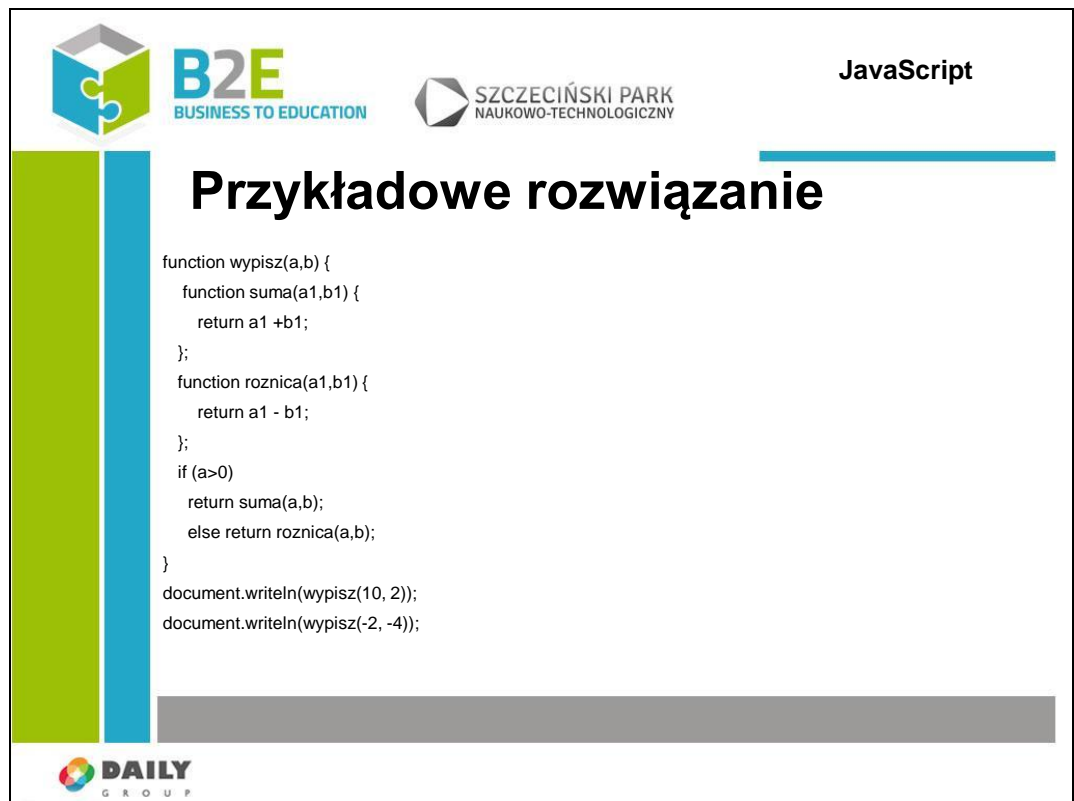
Ćwiczenia

Napisz funkcję przyjmującą dwa argumenty i zwracającą ich różnicę, gdy pierwszy jest nieujemny, lub ich sumę, gdy pierwszy jest ujemny. Obliczenie sumy i różnicy powinno zostać wykonane przez pomocnicze funkcje wewnętrzne.





Slajd 25



Slide 25 content: The slide features a header with logos for B2E (Business to Education), Szczeciński Park Naukowo-Technologiczny, and JavaScript. The main title is "Przykładowe rozwiązanie". Below the title is a JavaScript code block. At the bottom left is the DAILY GROUP logo.

```
function wypisz(a,b) {  
  function suma(a1,b1) {  
    return a1 +b1;  
  };  
  function roznica(a1,b1) {  
    return a1 - b1;  
  };  
  if (a>0)  
    return suma(a,b);  
  else return roznica(a,b);  
}  
document.writeln(wypisz(10, 2));  
document.writeln(wypisz(-2, -4));
```

Slajd 26



Slide 26 content: The slide features a header with logos for B2E (Business to Education), Szczeciński Park Naukowo-Technologiczny, and JavaScript. The main title is "JavaScript" with the subtitle "Obiekty". At the bottom left is the DAILY GROUP logo.

Slajd 27

JavaScript

Czym jest obiekt?

Obiekt:

- dowolny byt, który chcemy zapisać w pamięci komputera
- posiada właściwości, będące parami klucz i wartość
- może posiadać metody - funkcje operujące na obiekcie

DAILY GROUP

Obiekty są kolekcjami właściwości, gdzie każda właściwość ma nazwę oraz wartość. Nazwa może być dowolnym łańcuchem, również pustym. Wartość właściwości może być dowolną, dozwoloną wartością JavaScriptu, z wyjątkiem undefined. Obiekty JavaScriptu są bezklasowe. Nie ma żadnych ograniczeń co do nazw nowych właściwości lub ich wartości. Obiekty są wygodne do przechowywania i organizowania danych. Obiekty mogą zawierać inne obiekty.

Slajd 28

JavaScript

Obiekty w JavaScript

- zmienne
- funkcje
- tablice
- obiekty zdefiniowane przez użytkownika

DAILY GROUP

W języku JavaScript proste typy to liczby, łańcuchy, typy logiczne oraz typy specjalne (null i undefined). Wszystkie pozostałe elementy, wliczając funkcje, zmienne, tablice to obiekty

Slajd 29

JavaScript

Tworzenie prostych obiektów

- JSON - JavaScript Object Notation
- obiekt należy umieścić między nawiasami klamrowymi

```
{
  definicja obiektu
}
```

DAILY GROUP

Do tworzenia prostych obiektów można wykorzystać notację JSON (JavaScript Object Notation)

- notacja obiektów JavaScript. Definicję obiektu należy umieścić w nawiasach klamrowych. Definicja składa się z właściwości i przypisanych im wartości. Nazwa właściwości powinna być ciągiem znaków w cudzysłowie, natomiast wartość może być

łańcuchem znaków, liczbą, obiektem, tablicą itd. Kolejne właściwości oddzielamy od siebie przecinkiem.

Slajd 30

```
{  
  "producent": "Fiat",  
  "model": "Punto",  
  "rocznik": 2010  
}
```

Powyższa definicja opisuje przykładowy obiekt samochód. Zawiera 3 właściwości: producent, model i rocznik.

Slajd 31

```
var obiekt = {  
  //definicja  
}  
np.  
var samochod = {  
  "producent": "Fiat",  
  "model": "Punto"  
}
```

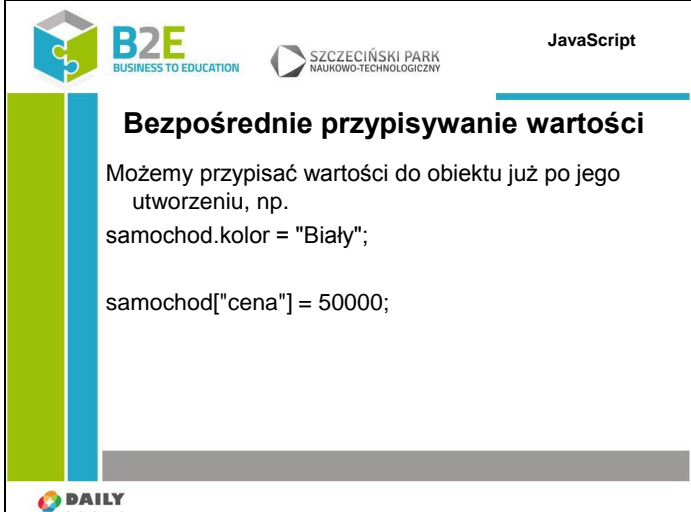
Obiekt JSON można przypisać do zmiennej albo wykorzystać funkcję eval.

Slajd 32

```
var zmienna = nazwaObiektu.nazwaWłaściwości  
np.  
var model_samochodu = samochod.model;
```

Po utworzeniu obiektu możemy uzyskać dostęp do jego właściwości. W tym celu korzystamy z operatora . (znak kropki)

Slajd 33



JavaScript

B2E
BUSINESS TO EDUCATION

SZCZECIŃSKI PARK
NAUKOWO-TECHNOLOGICZNY

Bezpośrednie przypisywanie wartości

Możemy przypisać wartości do obiektu już po jego utworzeniu, np.

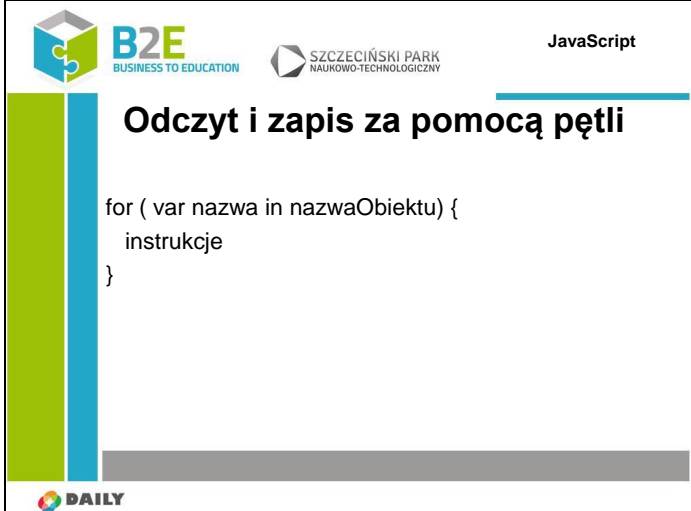
```
samochod.kolor = "Biały";
```

```
samochod["cena"] = 50000;
```

DAILY
GROUP

Jeśli przy operacji przypisania odwołamy się do nieistniejącej właściwości zostanie ona utworzona. Dostęp do właściwości obiektu możliwy jest za pomocą `.` lub nawiasu klamrowego. W nawiasie podajemy nazwę właściwości, której wartość chcemy zmodyfikować.

Slajd 34



JavaScript

B2E
BUSINESS TO EDUCATION

SZCZECIŃSKI PARK
NAUKOWO-TECHNOLOGICZNY

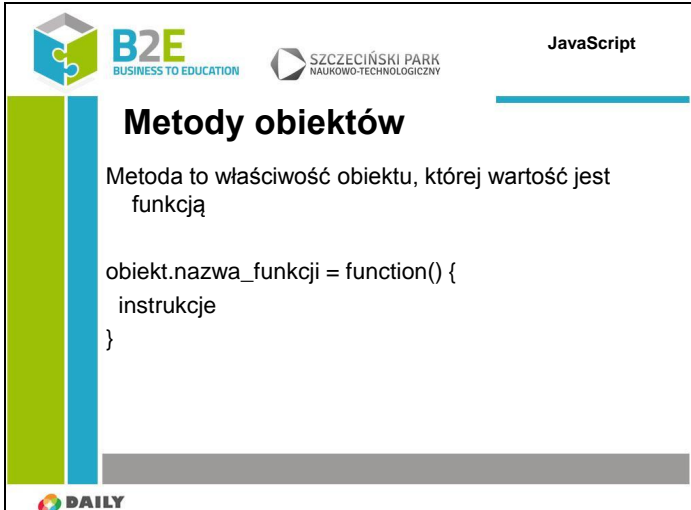
Odczyt i zapis za pomocą pętli

```
for ( var nazwa in nazwaObiektu) {  
  instrukcje  
}
```

DAILY
GROUP

Specjalna wersja pętli for działająca na właściwościach obiektu. Za każdą iteracją pod zmienną nazwa zostanie podstawiona kolejna właściwość obiektu nazwaObiektu. Pętla będzie wykonywana aż odczytane zostaną wszystkie właściwości, lub zostanie przerwana za pomocą instrukcji `return` lub `break`; Pętla pozwala w prosty sposób przejść po właściwościach obiektu bez konieczności posiadania informacji o ich nazwach i liczbie.

Slajd 35



JavaScript

B2E
BUSINESS TO EDUCATION

SZCZECIŃSKI PARK
NAUKOWO-TECHNOLOGICZNY

Metody obiektów

Metoda to właściwość obiektu, której wartość jest funkcją

```
obiekt.nazwa_funkcji = function() {  
  instrukcje  
}
```

DAILY
GROUP

Metody wykonują zwykle operacje na danych zawartych w danym obiekcie, więc muszą mieć dostęp do jego właściwości. Służy do tego słowo kluczowe `this`. Odróżnia się w ten sposób właściwość obiektu od innych zmiennych.

Slajd 36



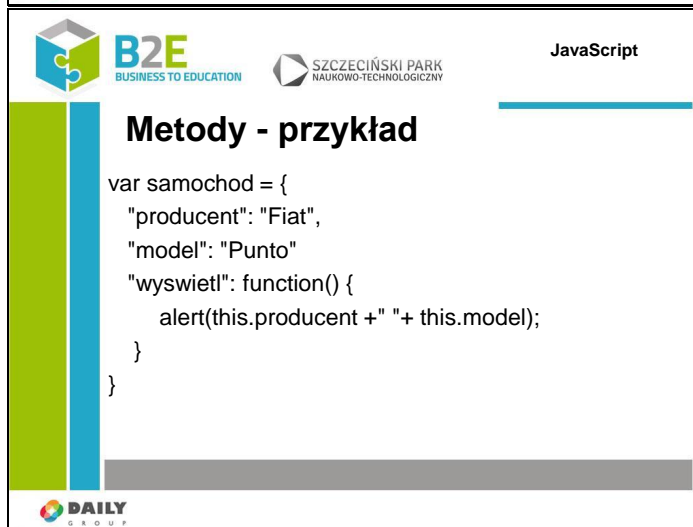
Slide 36 content: JavaScript logo, B2E logo, SZCZECIŃSKI PARK NAUKOWO-TECHNOLOGICZNY logo, and a code block for defining a method in JavaScript.

```

var obiekt = {
  //definicje właściwości
  nazwa_funkcji : function () {
    instrukcje
  }
}
    
```

Definicje metody można umieścić w notacji JSON, tak samo jak każdą inną właściwość obiektu. W ten sposób możemy utworzyć obiekt zawierający zarówno właściwości jak i funkcje. Metody obiektu mogą przyjmować argumenty, definiujemy je tak samo jak argumenty w funkcji./

Slajd 37



Slide 37 content: JavaScript logo, B2E logo, SZCZECIŃSKI PARK NAUKOWO-TECHNOLOGICZNY logo, and a code block for creating a car object with properties and a method.

```

var samochod = {
  "producent": "Fiat",
  "model": "Punto"
  "wyswietl": function() {
    alert(this.producent + " " + this.model);
  }
}
    
```

Tworzymy obiekt samochód zawierający właściwości producent i model oraz metodę wyświetlającą komunikat z informacją o wartościach właściwości obiektu.

Slajd 38



Slide 38 content: JavaScript logo, B2E logo, SZCZECIŃSKI PARK NAUKOWO-TECHNOLOGICZNY logo, and the title 'JavaScript Konstruktory i prototypy'.

Slajd 39



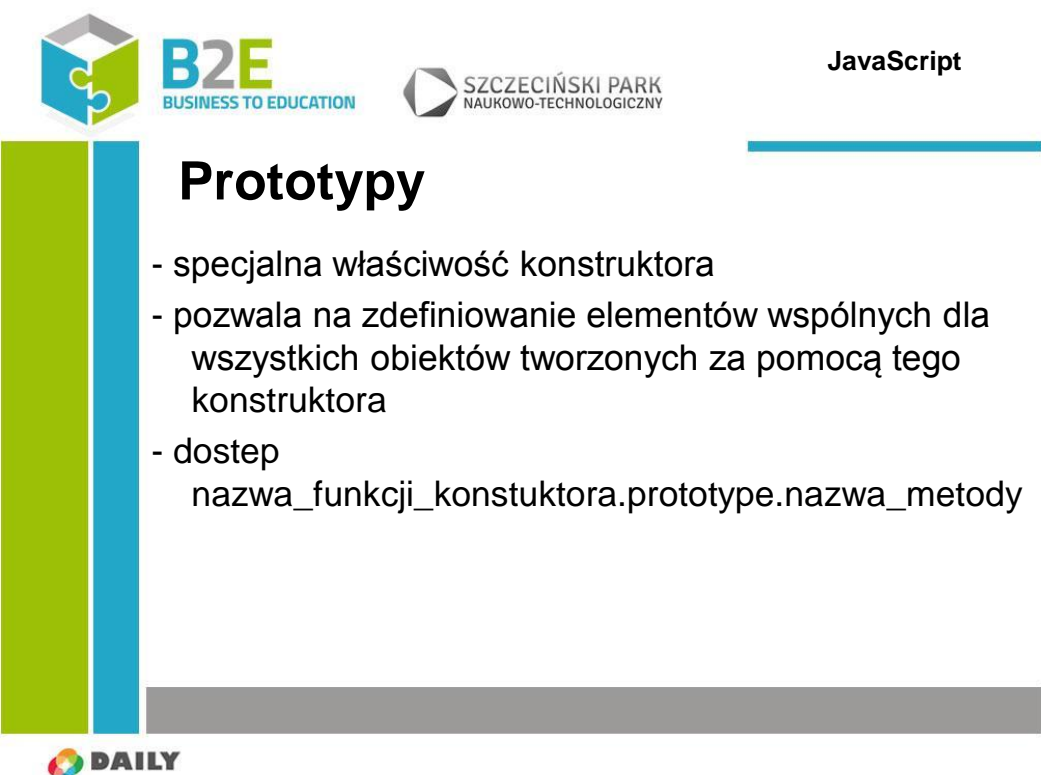
JavaScript

Konstruktory

```
function Samochod(producent, model, rocznik) {
  this.producent = producent;
  this.model = model;
  this.rocznik = rocznik;
}
var samochod = new Samochod("Fiat", "Punto", 2010);
```

Tworzenie obiektów za pomocą JSON nie jest jedyną dostępną możliwością. Możemy również utworzyć obiekt za pomocą operatora new, której jako parametr podamy specjalną funkcję zwaną konstruktorem. Operator new tworzy nowy pusty obiekt a następnie przekazuje go funkcji będącej jego argumentem. Zadaniem tej funkcji jest zainicjalizowanie obiektu domyślnymi wartościami.

Slajd 40



JavaScript

Prototypy

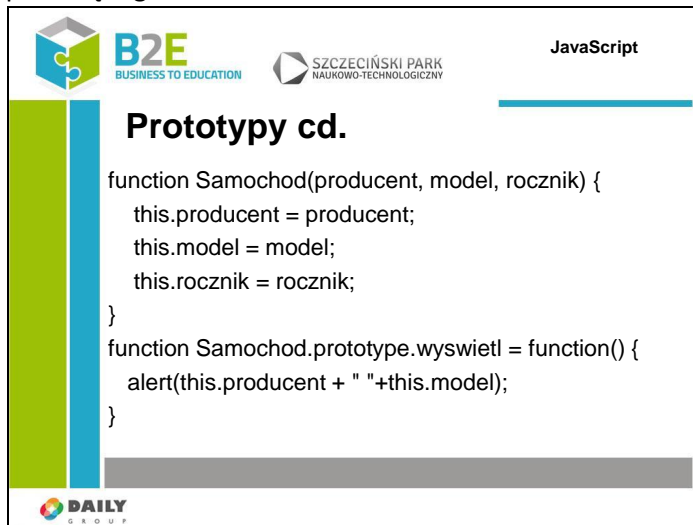
- specjalna właściwość konstruktora
- pozwala na zdefiniowanie elementów wspólnych dla wszystkich obiektów tworzonych za pomocą tego konstruktora
- dostęp `nazwa_funkcji_konstruktora.prototype.nazwa_metody`

Na jednym z wcześniejszych slajdów utworzyliśmy prosty obiekt zawierający 2 właściwości i metodę. Jednak definiowanie metod przy każdorazowym tworzeniu obiektu jest niewygodne i nieefektywne. Poznaliśmy też konstruktory obiektu, więc moglibyśmy umieścić definicję metod w konstruktorze, aby nie być zmuszonym do ich ciągłego wpisywania. Takie rozwiązanie mimo, że wygląda dobrze nadal jest nieefektywne. Każda instancja obiektu będzie zawierała własne właściwości oraz definicje metod. Właściwości zwykle są różne dla każdego obiektu danego typu, ale definicje metod są takie same, więc mogłyby być utworzone tylko raz.

W języku JavaScript w tym celu wykorzystuje się tzw. prototyp obiektu. Użycie operatora new powoduje nie tylko stworzenie pustego obiektu i przekazanie go funkcji będącej konstruktorem, ale także przygotowanie prototypu obiektu. Wartością prototypu jest wartość właściwości funkcji o nazwie prototype. Kiedy do prototypu dodamy nową funkcję lub właściwość będzie ona dostępna dla wszystkich obiektów tworzonych za

pomocą tego konstruktora.

Slajd 41



JavaScript

Prototypy cd.

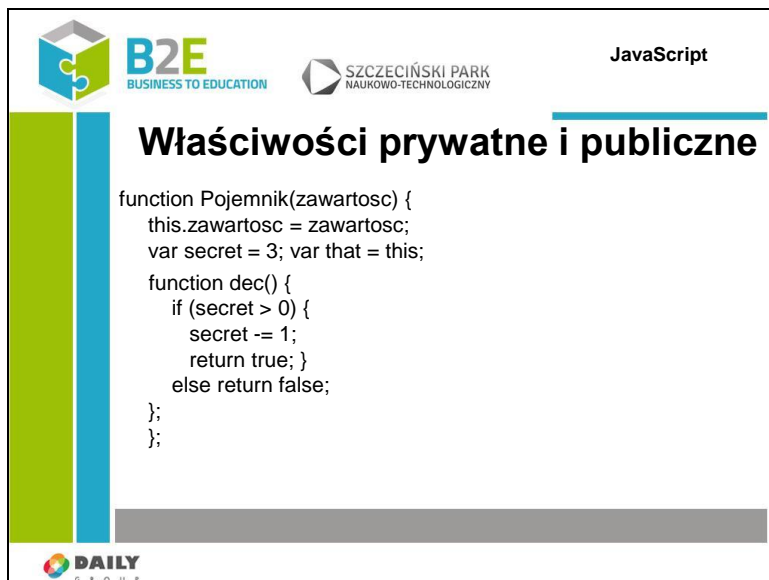
```
function Samochod(producent, model, rocznik) {
  this.producent = producent;
  this.model = model;
  this.rocznik = rocznik;
}
function Samochod.prototype.wyswietl = function() {
  alert(this.producent + " "+this.model);
}
```

DAILY GROUP

Na jednym z wcześniejszych slajdów utworzyliśmy prosty obiekt zawierający 2 właściwości i metodę. Jednak definiowanie metod przy każdorazowym tworzeniu obiektu jest niewygodne i nieefektywne. Poznaliśmy też konstruktory obiektu, więc moglibyśmy umieścić definicję metod w konstruktorze, aby nie być zmuszonym do ich ciągłego wpisywania. Takie rozwiązanie mimo, że wygląda dobrze nadal jest nieefektywne. Każda instancja obiektu będzie zawierała własne właściwości oraz definicje metod. Właściwości zwykle są różne dla każdego obiektu danego typu, ale definicje metod są takie same, więc mogłyby być utworzone tylko raz.

W języku JavaScript w tym celu wykorzystuje się tzw. prototyp obiektu. Użycie operatora new powoduje nie tylko stworzenie pustego obiektu i przekazanie go funkcji będącej konstruktorem, ale także przygotowanie prototypu obiektu. Wartością prototypu jest wartość właściwości funkcji o nazwie prototype. Kiedy do prototypu dodamy nową funkcję lub właściwość będzie ona dostępna dla wszystkich obiektów tworzonych za pomocą tego konstruktora.

Slajd 42



JavaScript

Właściwości prywatne i publiczne

```
function Pojemnik(zawartosc) {
  this.zawartosc = zawartosc;
  var secret = 3; var that = this;
  function dec() {
    if (secret > 0) {
      secret -= 1;
      return true; }
    else return false;
  };
};
```

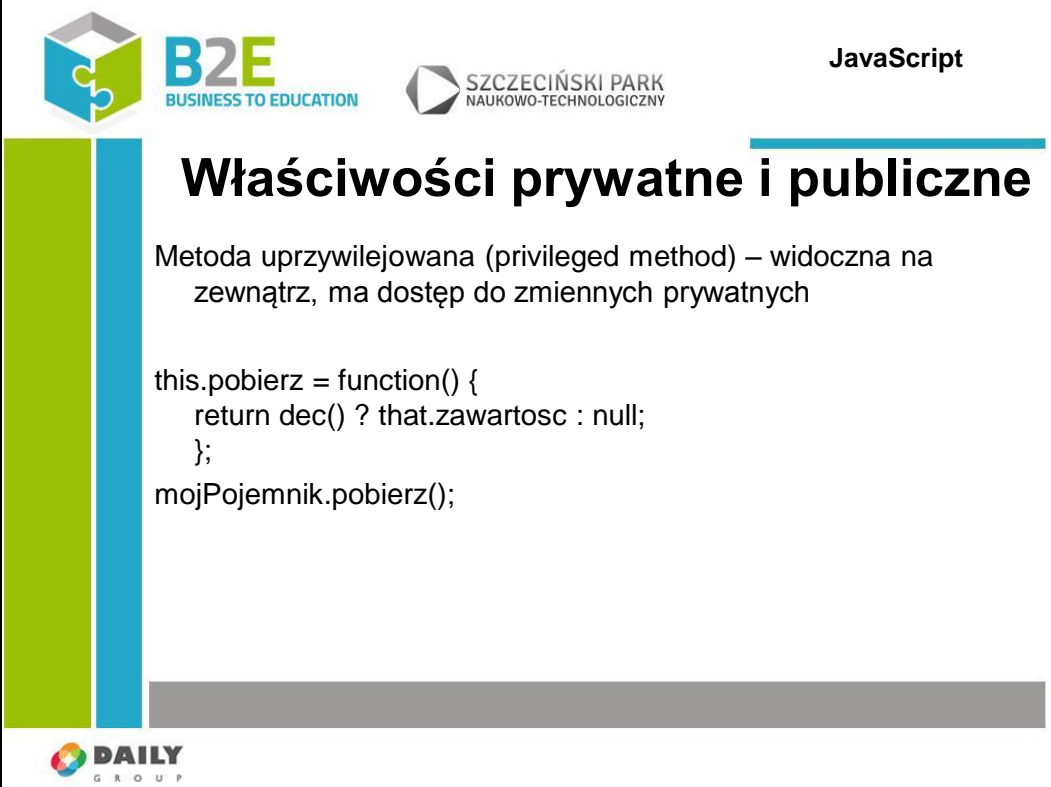
DAILY GROUP

Wszystkie właściwości obiektu są publiczne, tzn. można je odczytać, modyfikować i usuwać w dowolnym miejscu skryptu.

Parametry i zmienne utworzone w konstruktorze są prywatne. Są one dołączone do obiektu, ale nie są dostępne na zewnątrz funkcji, nie są również dostępne dla publicznych

metod obiektu. Aby z nich skorzystać należy wykorzystać prywatne metody – funkcje zdefiniowane w konstruktorze obiektu. Zmienna `that`, jest wykorzystywana w celu udostępnienia publicznych właściwości obiektu dla prywatnych metod – `this` w prywatnej metodzie będzie oznaczało funkcję konstruktora i zmienne w nim zdefiniowane.

Slajd 43



The slide features several logos at the top: a stylized 'C' logo, 'B2E BUSINESS TO EDUCATION', 'SZCZECIŃSKI PARK NAUKOWO-TECHNOLOGICZNY', and 'JavaScript'. The main title is 'Właściwości prywatne i publiczne'. Below the title is a description of a privileged method and a code snippet.

Właściwości prywatne i publiczne

Metoda uprzywilejowana (privileged method) – widoczna na zewnątrz, ma dostęp do zmiennych prywatnych

```
this.pobierz = function() {  
    return dec() ? that.zawartosc : null;  
};  
mojPojemnik.pobierz();
```



DAILY GROUP logo is visible at the bottom left of the slide content.

Metoda uprzywilejowana ma dostęp do prywatnych zmiennych i metod i jest dostępna dla publicznych metod i innych funkcji. Metody te są dodawane w konstruktorze i podłączone do obiektu `this`.

Wywołanie funkcji `pobierz` zwróci zawartość obiektu pierwsze 3 razy kiedy zostanie wywołana. Po tym zawsze będzie zwracać `null`. Metoda wywołuje prywatną metodę `dec`, która działa na prywatnej zmiennej `secret`. Metoda `pobierz` jest dostępna dla innych publicznych metod i funkcji, ale nie pozwala na bezpośredni dostęp do prywatnych zmiennych.



Slajd 44

JavaScript

Właściwości prywatne i publiczne

Publiczne:


```
function Konstruktor(...) {  
  this.nazwaWlasciwosci = wartosc;  
}  
Konstruktor.prototype.nazwaWlasciwosci = wartosc;
```

Prywatne:



```
function Konstruktor(...) {  
  var that = this; var nazwaWlasciwosci = wartosc;  
  function nazwaFunkcji(...) {.....}  
}
```

Uprzywilejowane

```
function Konstruktor(...) {  
  this.nazwaFunkcji = function (...) { ..... }  
}
```




Slajd 45

JavaScript

Ćwiczenie 1


Napisz kod konstruktora obiektu, będącego reprezentacją koła na płaszczyźnie. W konstruktorze umieść metody zwracające pole i obwód koła.




Napisz kod konstruktora obiektu, będącego reprezentacją koła na płaszczyźnie. W konstruktorze umieść metody zwracające pole i obwód koła.



Slajd 46



B2E
BUSINESS TO EDUCATION




SZCZECIŃSKI PARK
NAUKOWO-TECHNOLOGICZNY

JavaScript


Ćwiczenie 1 - rozwiązanie

```
function Kolo(x, y, r) {  
  this.x = x;  
  this.y = y;  
  this.r = r;  
  this.pole = function() {  
    return 3,14 * r * r;  
  }  
  this.obwod = function() {  
    return 2 * 3,14 * r;  
  }  
}
```




DAILY GROUP

Slajd 47



B2E
BUSINESS TO EDUCATION




SZCZECIŃSKI PARK
NAUKOWO-TECHNOLOGICZNY

JavaScript

Ćwiczenie 2

Napisz kod konstruktora obiektu reprezentującego prostokąt. W prototypie obiektu umieść metody zwracające pole i obwód prostokąta.




DAILY GROUP


Napisz kod konstruktora obiektu reprezentującego prostokąt. W prototypie obiektu umieść metody zwracające pole i obwód prostokąta.



Slajd 48



B2E
BUSINESS TO EDUCATION




SZCZECIŃSKI PARK
NAUKOWO-TECHNOLOGICZNY

JavaScript

Ćwiczenie 2 - rozwiązanie

```

function Prostokat (a, b) {
  this.a = a;
  this.b = b;
}
Prostokat.prototype.pole = function() {
  return a* b;
}
Prostokat.prototype.obwod = function() {
  return 2*a + 2*b;
}
    
```



DAILY
GROUP

Ćwiczenia

Ćwiczenia do lekcji zostały zaprezentowane na slajdach w poprzednim punkcie.

Wymagają one od uczestnika umiejętności definiowania obiektów w JavaScript, a także ich konstruktorów, prototypów i funkcji, które mogą być wykorzystane jako metody obiektu.

Opis założonych osiągnięć ucznia

Po ukończeniu tej lekcji uczestnik wie czym są obiekty w języku JavaScript, potrafi je tworzyć. Wie także jak grupować instrukcje w funkcje i w jaki sposób można je wykorzystywać.

Lekcja 6 Model dokumentu – Document Object Model

Cel lekcji

Celem lekcji jest wprowadzenie definicji modelu dokumentu, jako reprezentacji strony internetowej, która może być przetwarzana przez skrypty i programy.

Sposoby osiągnięcia celów kształcenia

Czynności nauczyciela	Czynności ucznia
wprowadza nowe pojęcia związane z tematem zajęć (materiał ze slajdów) przygotowuje ćwiczenia z zaprezentowanego materiału	Wykonuje zadane przez nauczyciela ćwiczenia na komputerze
Prowadzi z uczniami dyskusję na temat wybranych przez nich rozwiązań ćwiczeń	Prezentuje rozwiązania ćwiczeń nauczycielowi i pozostałym uczniom, tłumaczy wybrane przez siebie rozwiązanie
Podsumowuje lekcję, ocenia pracę uczniów na zajęciach.	Dokonuje samooceny swojej pracy, omawia z nauczycielem rozwiązania ćwiczeń

Treść – slajdy z opisem

Slajd 1

Każda strona www składa się z wielu elementów (linków, paragrafów, obrazów itp.) Każdy z tych obiektów może mieć swoje właściwości (jak rozmiar, kolor) i można wykonywać na nich różne operacje. Dodatkowo dostępne są jeszcze obiekty opisujące okno przeglądarki i dokumentu załadowanego w bieżącym oknie. Do wygodnego zarządzania wszystkimi obiektami stworzono model dokumentu – Dokument Object Model, który określa hierarchię obiektów na stronie, ich właściwości oraz sposób zachowania.

Slajd 2

```

graph TD
    window[window] --- document[document]
    window --- history[history]
    window --- location[location]
    window --- navigator[navigator]
    window --- screen[screen]
    
```

Obsługą standardu zajmuje się organizacja W3C (World Wide Web Consortium – <http://www.w3.org>), ale poszczególne przeglądarki w różnym stopniu respektują zalecenia standardu.

Document Object Model zmienia każdy element witryny www w hierarchie obiektów, z których każdy posiada swoje własne właściwości. Właściwości opisują wszystkie atrybuty



obiektu takie jak HTML, który zawiera, jego wysokość, szerokość itd. Najbardziej zewnętrznym elementem hierarchii (jej korzeniem) jest obiekt window, który jest bieżącym oknem przeglądarki, zakładką, ramką pływającą (iframe) lub oknem typu popup.

Poniżej znajdują się następujące obiekty:

- dokument, który jest reprezentacją dokumentu (X)HTML,
- obiekt history zawiera historię odwiedzin stron podczas danej sesji przeglądarki,
- obiekt location reprezentuje adres URL aktualnego dokumentu,
- obiekt navigator przechowuje informacje o przeglądarce, jej nazwę, wersje itp.,
- obiekt screen.

Slajd 3

JavaScript

Obiekt window

- główny i domyślny obiekt w DOM
- reprezentuje okno lub zakładkę w przeglądarce
- zmienne globalne w JavaScript są w praktyce właściwościami tego obiektu

DAILY GROUP

Obiekt window reprezentuje okno przeglądarki (lub jeśli korzystamy z przeglądarek typu Firefox – jego zakładkę). Jest obiektem domyślnym tzn. że jego metody i właściwości możemy wywoływać bezpośrednio – bez konieczności podawania jego nazwy.

Slajd 4

JavaScript

Przykładowe metody obiektu window

- alert(treść) – wyświetla komunikat zawarty w argumencie treść
- confirm(treść) – wyświetla komunikat zawarty w argumencie treść oraz przyciski OK i Anuluj
- prompt(treść [,opis]) – wyświetla okno dialogowe z polem pozwalającym wprowadzić odpowiedź
- back() – odpowiednik przycisku wstecz w przeglądarce
- close() – zamyka okno
- open(URL, nazwa [właściwości, [,zamiana]])- wyświetla nowe okno o określonej nazwie.

DAILY GROUP

Metoda alert wyświetla okienko z komunikatem i przyciskiem OK, metoda confirm pozwala nam uzyskanie potwierdzenia od użytkownika – wyświetlone

zostanie okno z komunikatem i przyciski OK, i Anuluj.

Wynikiem wywołania metody będzie true – jeśli wciśnięto OK lub false jeśli wciśnięto Anuluj.

Metoda prompt wyświetla komunikat zawarty w parametrze treść i pole tekstowe na wprowadzenie odpowiedzi. W polu tekstowym zostanie wyświetlony tekst wprowadzony jako opis, wynikiem działania jest treść wpisana przez użytkownika

Metoda back() cofnie nas o 1 stronę w historii przeglądarki

Metoda close() zamyka okno przeglądarki

Metoda open() otwiera adres Url w nowym oknie przeglądarki o podanej nazwie i właściwościach (np. rozmiarze)

Slajd 5

Obiekt document

zawiera dokument html wczytany do przeglądarki
pozwala na dostęp i manipulacje praktycznie każdym elementem strony

Obiekt document reprezentuje dokument html wczytany do okna przeglądarki oraz zawiera szereg właściwości i metod pozwalających na jego modyfikację. Poprzez ten obiekt można otrzymać dostęp praktycznie do każdego elementu strony i za jego pomocą można tymi elementami manipulować.

Slajd 6

Podstawowe metody obiektu document

- write(tekst)
- writeln (tekst)
- getElementById(id)
- getElementsByName(nazwa)
- getElementByTagName(tag)

•write(tekst) – wpisuje w dokumencie tekst zawarty w argumencie tekst
•writeln (tekst) – tak samo jak write, ale na końcu dokleja znak nowej linii
•getElementById(id) – zwraca odnośnik do obiektu o identyfikatorze id
•getElementsByName(nazwa) – zwraca listę obiektów o atrybucie nazwa równym przekazanej nazwie
•getElementByTagName(tag) – zwraca listę obiektów o danym znaczniku

Slajd 7

JavaScript

Obiekt history

Właściwości

- current
- length
- next
- previous

Metody:

- back()
- forward()
- go(parametr)

DAILY GROUP

Przechowuje historię stron odwiedzanych przez użytkownika w danej sesji przeglądarki. Zawiera metody pozwalające na przemieszczanie się między odwiedzanymi stronami history.current – zawiera adres URL bieżącej strony, history.length – liczba stron przechowywanych w historii właściwości next i previous zawierają adresy stron dostępnych pod przyciskami wstecz i dalej.

Podstawowe metody obiektu to:

back() i forward() – wczytują odpowiednio poprzednią i następną stronę w historii go- wczytuje stronę podaną jako parametr. Jeśli parametr jest liczbą to jest traktowany jako pozycja względem aktualnej strony w historii przeglądania. Jeśli parametr jest ciągiem znaków to traktowany jest jako adres URL

Slajd 8

JavaScript

Obiekt location

Zawiera adres URL aktualnego dokumentu

Metody:

- assign(url)
- reload(force)
- replace(url)

DAILY GROUP

Zawiera adres URL aktualnego dokumentu oraz metody pozwalające na manipulacje tym adresem. We właściwościach obiektu znajdują się informacje składowe adresu (host,

nazwa hosta, port).

Podstawowe metody:

- `assign(url)` – powoduje wczytanie dokumentu o adresie wskazywanym w argumencie URL
- `reload(force)` – wymusza ponowne wczytanie strony. Jeśli parametr `force` jest ustawiony na `true` to strona jest ponownie wczytywana z serwera, w przeciwnym wypadku może zostać wczytana z pamięci cache
- `replace(url)` – powoduje wczytanie dokumentu o adresie wskazywanym w argumencie URL, bieżąca strona nie zostanie zapamiętana w historii.

Slajd 9

JavaScript

Obiekt navigator

Zawiera informacje o przeglądarce i systemie operacyjnym na którym działa

Zawiera właściwość `userAgent`, którą można wykorzystać do określenia typu przeglądarki

```
document.write(navigator.userAgent)
```

Przykładowe skrypty określające typ przeglądarki to:

- `whichbrowser` (www.whichbrowser.net)
- `UAParser.js` (<http://faisalman.github.io/ua-parser-js>)

Za pomocą właściwości `userAgent` obiektu `navigator` możemy określić typ przeglądarki i w zależności od wyniku sterować naszym skryptem. Niestety właściwość ta zawiera oprócz nazwy przeglądarki także jej wersję oraz system na którym jest uruchomiona. Utrzymywanie takiego skryptu wymaga czasu – dlatego w internecie dostępne są gotowe skrypty, które możemy wykorzystać. Posiadają one zwykle wsparcie społeczności programistów i jeśli pojawia się jakiś błąd jest on szybko usuwany.



Slajd 10

Slide 10 content: The slide features a header with logos for B2E (Business to Education), Szczeciński Park Naukowo-Technologiczny, and JavaScript. The main title is "Dostęp do elementów strony". Below the title is a block of HTML code:


```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
"http://www.w3.org/TR/html4/strict.dtd">
<html>
<head>
<title>Tester JavaScript</title>
<script type="text/javascript">
</script>
<body>
  <div id="content">
    Witaj!
  </div>
</body>
</html>
```

 At the bottom left is the DAILY GROUP logo.

Strona www jest wczytywana w przeglądarce jako struktura hierarchiczna (drzewo), której elementy są zbudowane ze znaczników html. Wykorzystując JavaScript możemy wyszukiwać i modyfikować elementy, usuwać i tworzyć nowe.

Slajd 11

Slide 11 content: The slide features the same header as slide 10. The main title is "Dostęp do elementów strony". Below the title is a block of JavaScript code:


```
metoda getElementById()
var content = document.getElementById("content");
var tresc = content.innerHTML;
content.innerHTML = tresc;
```

 At the bottom left is the DAILY GROUP logo.

Strona www jest wczytywana w przeglądarce jako struktura hierarchiczna (drzewo), której elementy są zbudowane ze znaczników html. Wykorzystując JavaScript możemy wyszukiwać i modyfikować elementy, usuwać i tworzyć nowe.

Do znalezienia danego obiektu najlepiej wykorzystać metodę getElementById obiektu document. Zwraca ona referencje do obiektu w drzewie dokumentu o podanym id. Taką referencje najlepiej przypisać do zmiennej i traktować jako obiekt.

Do modyfikacji elementu można wykorzystać właściwość innerHtml. Większość elementów drzewa DOM posiada taką właściwość. innerHTML to zawartość obiektu w postaci HTML.

Możemy odczytać zawartość wyszukanego przez nas elementu (w naszym przypadku

Slajd 12

będzie to napis „Witaj!”) lub zmodyfikować.

The slide content includes logos for B2E (Business to Education), Szczeciński Park Naukowo-Technologiczny, and Daily Group. The title is "Bezpośrednia manipulacja węzłami dokumentu". The text explains that non-text elements contain a collection of child nodes (childNodes) and provides a JavaScript code snippet: `var zmienna = content.childNodes[0]; zmienna.nodeValue = "Nowa treść";`. The slide is titled "JavaScript" in the top right corner.

W innerHTML możemy umieścić treść będącą kodem HTML, po przypisaniu tej wartości przeglądarka sama przebuduje drzewo DOM dokumentu. Czasami jednak wymagana jest większa kontrola nad węzłami dokumentu.

W DOM każdy węzeł nie będący tekstem zawiera kolekcję węzłów potomnych. Np. węzeł `<body>` zawiera węzeł potomny `<div>`, który zawiera węzeł potomny z tekstem Witaj.

Dostęp do elementów potomnych realizowany jest za pomocą tablicy `childNodes`. W zmiennej `zmienna` uzyskamy 1 element potomny węzła `content`. Dostęp do wartości tego obiektu możliwy jest przez właściwość `nodeValue`.

Slajd 13

The slide content includes logos for B2E (Business to Education), Szczeciński Park Naukowo-Technologiczny, and Daily Group. The title is "Tworzenie elementów przez skrypt". The text explains that nodes are linked to HTML elements and provides JavaScript code: `var div = document.createElement("div");` for creating a `div` element, `var tekst = document.createTextNode("Treść");` for creating a text node, and `div.appendChild(tekst); content.appendChild(div);` for attaching them. The slide is titled "JavaScript" in the top right corner.

Za pomocą JavaScriptu możemy nie tylko modyfikować istniejące ale także dodawać nowe elementy do strony. Aby tego dokonać trzeba wyróżnić kilka typów węzłów:

węzły powiązane z elementami html tworzymy za pomocą metody `createElement` obiektu `document` – jako argument podajemy nazwę tworzonego znacznika.

Natomiast węzły zawierające tylko tekst tworzymy za pomocą innej metody obiektu `document` - `createTextNode`. Tak utworzone węzły łączymy ze sobą za pomocą metody `appendChild`. Żeby taki połączony węzeł był widoczny musimy połączyć go z innym elementem naszej strony.



Slajd 14

JavaScript

Usuwanie elementów

```
element.removeChild(obiekt);  
np. content.removeChild(div);
```

DAILY GROUP

Skoro możemy modyfikować i dodawać elementy to pewnie istnieje też sposób na usuwanie elementów. Służy do tego metoda `removeChild`, której jako parametr przekazujemy obiekt do usunięcia. `element` jest węzłem z którego usuwamy węzeł potomny. Usunięcie obiektu jest tylko odłączeniem go od elementu nadrzędnego, a nie usunięciem z pamięci. Węzeł nadal istnieje a referencje do niego otrzymaliśmy w wyniku działania metody `removeChild`.

Slajd 15

JavaScript



Ćwiczenia

Umieść w kodzie HTML warstwę zdefiniowaną za pomocą znacznika `<div>`. Napisz skrypt, który doda do tej warstwy elementy, jakie w postaci HTML miałyby postać: `<i>Szkolenie JavaScript</i>`. Nie używaj właściwości `innerHTML`.

DAILY GROUP




Slajd 16





JavaScript

Przykładowe rozwiązanie

```
var str = "Szkolenie JavaScript";  
var element = document.getElementById("content");  
var bEl = document.createElement("b");  
var iEl = document.createElement("i");  
var iElTextNode = document.createTextNode(str);  
bEl.appendChild(iEl);  
iEl.appendChild(iElTextNode);  
element.appendChild(bEl);
```




Slajd 17



JavaScript

Ćwiczenia

Napisz skrypt, który doda do warstwy div 5 elementów typu <p> z dowolną treścią.



Slajd 18

B2E BUSINESS TO EDUCATION | **SZCZECIŃSKI PARK** NAUKOWO-TECHNOLOGICZNY | JavaScript

Przykładowe rozwiązanie

```
var content = document.getElementById("content");
for (i = 0; i<5; i++) {
  var p = document.createElement("p");
  p.innerHTML = "Tekst";
  content.appendChild(p);
}
```

Celownik - najlepsza inwestycja

KAPITAŁ LUDZKI NARODOWA STRATEGIA SPÓJNOŚCI | UNIA EUROPEJSKA EUROPEJSKI FUNDUSZ SPOŁECZNY

Projekt współfinansowany przez Unię Europejską w ramach Europejskiego Funduszu Społecznego

DAILY GROUP

Ćwiczenia

Napisz skrypt wyszukujący na stronie element o identyfikatorze "test". Zmień kolor jego tła na zielony. Napisz skrypt umieszczający treść pierwszego paragrafu strony w znaczniku <h1>

Opis założonych osiągnięć ucznia

Uczestnik po zakończeniu lekcji posiada podstawowe informacje na temat drzewa dokumentu (modelu DOM). Potrafi wyszukiwać w nim elementy i je modyfikować.

Lekcja 7 Zdarzenia

Cel lekcji

Interakcja na stronach www realizowana jest przez zdarzenia. „Coś” się na stronie wydarzyło. Celem lekcji jest wprowadzenie pojęcia zdarzenie i pokazanie w jaki sposób można reagować na zmiany na naszej witrynie.

Sposoby osiągnięcia celów kształcenia

Czynności nauczyciela	Czynności ucznia
wprowadza nowe pojęcia związane z tematem zajęć (materiał ze slajdów) przygotowuje ćwiczenia z zaprezentowanego materiału	Wykonuje zadane przez nauczyciela ćwiczenia na komputerze
Prowadzi z uczniami dyskusję na temat wybranych przez nich rozwiązań ćwiczeń	Prezentuje rozwiązania ćwiczeń nauczycielowi i pozostałym uczniom, tłumaczy wybrane przez siebie rozwiązanie
Podsumowuje lekcję, ocenia pracę uczniów na zajęciach.	Dokonuje samooceny swojej pracy, omawia z nauczycielem rozwiązania ćwiczeń

Treść – slajdy z opisem

Slajd 1

Zdarzeniem w JavaScript może być kliknięcie myszą, przesunięcie kursora nad element, opuszczenie strony itp. W JS możemy zdefiniować specjalne funkcje zwane funkcjami obsługi zdarzeń (z ang. event handler).
W tej lekcji zobaczymy jakie typy zdarzeń występują oraz jak włączyć ich obsługę.

Slajd 2

Reagowanie na zdarzenia zachodzące na stronie zwiększa jej dynamikę i atrakcyjność. Kiedy coś się dzieje na stronie przeglądarka generuje zdarzenie (obiekt) a następnie bada czy jest zarejestrowana funkcja obsługi tego zdarzenia. Jeśli tak to funkcja ta jest wywoływana.

Istnieje kilka metod przypisywania rejestracji zdarzeń w JavaScriptcie

-inline, zwany też podstawowym – najstarszy, obsługiwany podobnie przez większość przeglądarek.


Zdarzenie zwykle utożsamiane z nazwą funkcji obsługującej, która jest jednocześnie nazwą atrybutu znacznika html, do którego podpinamy zdarzenie. Ten sposób ma kilka wad

- po pierwsze miesza kod JavaScript z treścią strony,
- nie przekazuje automatycznej referencji do obiektu, który wywołał zdarzenie




(można to obejść przekazując referencje this jako argument wywołania funkcji)
-oddzielenie kodu JS od html – umieszczenie wielu instrukcji JavaScript bezpośrednio w znaczniku jest możliwe, ale jeśli chcemy wykorzystać nasz skrypt ponownie to musimy skopiować jego treść i podłączyć do nowego elementu. Możemy to uprościć i przenieść nasz kod do funkcji a w znacznikach umieścić tylko wywołania naszych funkcji. Jednak gdy będziemy chcieli zmienić obsługę części zdarzeń na naszej stronie będziemy musieli przeszukiwać dokument html i wyszukiwać elementy, które wywołują naszą funkcję. Dlatego możliwe jest podłączenie obsługi zdarzenia bezpośrednio w kodzie skryptu. Możemy do tego wykorzystać poznaną wcześniej metodę getElementById, aby zlokalizować dany element na stronie i umieścić w nim obsługę zdarzenia. Aby usunąć obsługę zdarzenia przypisujemy mu wartość null.

Slajd 3



B2E
BUSINESS TO EDUCATION



SZCZECIŃSKI PARK
NAUKOWO-TECHNOLOGICZNY

JavaScript

Obsługa zdarzeń (event handler)


Metody rejestracji zdarzeń:

- nowy model rejestracji zdarzeń (addEventListener)

```
var element = document.getElementById('przycisk');
element.addEventListener('click', przenies, false);
element.addEventListener('click', wypiszTekst, false);
element.addEventListener('click', function() {
    this.style.color = 'blue';
}, false);
```

- removeEventListener()

```
element.removeEventListener('click', przenies, false);
```



Wadą wcześniej przedstawionych sposobów jest możliwość podłączenia tylko jednej funkcji pod dane zdarzenie. Jeśli w trakcie działania przypiszemy do zdarzenia nową funkcję to nadpiszemy starszą.

Możemy to obejść używając „nowego” modelu rejestracji zdarzeń z wykorzystaniem metody addEventListener();

Przyjmuje ona 3 parametry: typ zdarzenia, funkcja do wywołania parametr włączający lub wyłączający bąbelkowe wywołania zdarzeń (z reguły jest on ustawiony na wartość false)

Bąbelkowe wywołanie zdarzeń:

(jeśli jest ustawione na true, zdarzenie jest wywoływane dla wszystkich elementów nadrzędnych – czyli jeśli klikniemy w obszarze div, który jest częścią innego obszaru to zdarzenie onClick zostanie przekazane do obu elementów!. Jeśli oba elementy zawierają funkcję obsługującą to zdarzenie to obie funkcje zostaną wywołane).

Zdarzenie „wędruje” od elementu który je wywołał w górę dokumentu (aż do elementu document) tak jak bąbelki powietrza wędrują w górę butelki.

W elemencie przycisk przypisujemy 3 funkcje obsługujące kliknięcie: przenies, wypiszTekst, i anonimową funkcję zmieniającą kolor na niebieski. Od tego momentu

kliknięcie na elemencie uruchomi wszystkie 3 funkcje.
Aby usunąć obsługę zdarzenia z elementu wykorzystujemy metodę `removeEventListener()`, której jako argument podajemy takie same parametry jak metodzie `addEventListener()`.

Niestety nie możemy jednak usunąć anonimowych funkcji!

Slajd 4

JavaScript

Problemy z nowym modelem

starsze wersje przeglądarki Internet Explorer stosują swój własny model

```
var element = document.getElementById('przycisk');
element.attachEvent('onClick', funkcja);
element.detachEvent('onClick', funkcja);
```

DAILY GROUP

Wcześniejsze wersje Internet Explorera (≤ 8) stosują własny model. Zamiast zdarzenia `addEventListener` i `removeEventListener` są zdarzenia `attachEvent` i `detachEvent`. Jako parametry tych funkcji podajemy nazwę zdarzenia z przedrostkiem „on” np. `onClick` a jako 2 nazwę funkcji obsługującej zdarzenie.

Slajd 5

JavaScript

Słowo kluczowe this

JavaScript udostępnia słowo kluczowe `this`, które jest referencją do obiektu, wywołującego metodę.

```
function zmienKolor(kolor) {
    this.style.color = kolor;
}
element.zmienKolor('#808080');
```

DAILY GROUP

JavaScript udostępnia słowo kluczowe `this`, które jest referencją do obiektu, wywołującego metodę.

Przykładowo jeśli chcemy stworzyć funkcję, która zmieni kolor elementu możemy napisać ją w następujący sposób i wywołać.



Slajd 6

B2E
BUSINESS TO EDUCATION

SZCZECIŃSKI PARK
NAUKOWO-TECHNOLOGICZNY

JavaScript

Dodatkowe opcje

- blokowanie domyślnej akcji (preventDefault())

```

element.addEventListener('click',function (e) {
  alert('Ten link nigdzie nie przeniesie.');
```

```

  e.preventDefault();
},false)
```

- blokowanie nasłuchu innych zdarzeń (stopPropagation())

```

a.addEventListener('click',function (e) {
  alert('Kliknięto link');
```

```

  e.preventDefault();
  e.stopPropagation()
},false);
```

Elementy na stronie mają w większości przypadków przypisane domyślne akcje. Np. link przenosi w inne miejsce, formularz jest wysyłany na serwer itp.

Dodane przez nas zdarzenia zostaną wykonane jako pierwsze, ale później zostanie wykonana domyślna czynność.

Aby temu zapobiec możemy skorzystać z metody `e.preventDefault()`, gdzie `e` to obiekt zdarzenia. Zdarzenie po wystąpieniu przekazywane jest w górę dokumentu – aż do znacznika `document` i jeśli po drodze napotka na metodę obsługującą dane zdarzenie to metoda ta zostanie wywołana.

Aby wstrzymać takie działanie możemy wykorzystać metodę `stopPropagation()`

Slajd 7

B2E
BUSINESS TO EDUCATION

SZCZECIŃSKI PARK
NAUKOWO-TECHNOLOGICZNY

JavaScript

Lista zdarzeń

- onblur
- onchange
- onclick
- ondblclick
- onfocus
- onkeypress
- onload

Lista typowych zdarzeń obsługiwanych przez przeglądarkę:

-onblur: zdarzenie uruchamiane gdy element straci focus (zmienimy aktywny element na inny)

- onchange: element traci focus i zmienia się zawartość elementu
- onclick: zdarzenie, które powstaje kiedy element został kliknięty
- ondblclick: zdarzenie po dwukrotnym kliknięciu myszą
- onfocus: element otrzymuje fokus (jest aktywny)
- onkeypress: zdarzenie powstaje kiedy klawisz zostaje wciśnięty i puszczony (są jeszcze zdarzenia onkeydown i onkeyup – wywoływane po naciśnięciu i puszczeniu przycisku)
- onload: przeglądarka zakończyła ładowanie strony lub ramki

Slajd 8

JavaScript

Lista zdarzeń cd

- onmousedown
- onmousemove
- onmouseout
- onmouseover
- onmouseup
- onresize

Lista typowych zdarzeń obsługiwanych przez przeglądarkę:

- onmousedown: zdarzenie powstaje gdy klawisz myszy zostanie naciśnięty nad elementem
- onmousemove: kursor myszy jest przesuwany nad elementem
- onmouseout: kursor myszy opuścił element
- onmouseover: kursor myszy wszedł w element
- onmouseup: przycisk myszy został zwolniony nad elementem
- onresize: zmienił się rozmiar okna

Slajd 9

JavaScript

Ładowanie strony

```
<html>
<head>
<title>Moja strona</title>
</head>
<body onload="function() {alert('załadowano stronę');}" >
</body>
</html>
```

Komunikat „Załadowano stronę” pojawi się dopiero po wczytaniu całej strony do przeglądarki. Jest to bardzo przydatne, jeśli nasza funkcja ma działać na elementach strony. W tym przypadku mamy pewność, że wszystkie elementy są już wczytane i utworzone w przeglądarce a nasza metoda będzie mogła z nich skorzystać. Gdybyśmy umieścili wywołanie naszej metody poza zdarzeniem zostałaaby wywołana w momencie wczytania przez przeglądarkę, bez oczekiwania na zakończenie wczytywania strony.



Slajd 10

Obsługa kliknięć

- onclick

```
<div id="przycisk" style="background-color: red; width:100px; height: 100px;" onclick="alert('Witaj!');" > </div>
```
- ondblclick

```
<div id="przycisk" style="background-color: red; width:100px; height: 100px;" ondblclick="alert('Witaj 2x');" > </div>
```

Najczęściej wykorzystywane zdarzenie. Metodę obsługi zdarzenia możemy przypisać do większości elementów strony.

W tym przykładzie utworzyliśmy element div o rozmiarze 100 na 100px i podłączyliśmy do niego zdarzenie click z instrukcją wywołującą okno z komunikatem.

Zdarzenie dblclick jest wywoływane po dwukrotnym kliknięciu na element. Jeśli element ma podłączoną metodę onclick to zostanie ona wywołana zamiast oczekiwanej przez nas metody ondblclick



Slajd 11

B2E
BUSINESS TO EDUCATION

SZCZECIŃSKI PARK
NAUKOWO-TECHNOLOGICZNY

JavaScript

Reakcja na ruch myszą

zdarzenia onmouseover i onmouseout

```
<div id="przycisk"
  style="background-color: red; width:100px; height: 100px;"
  onmouseover="alert('kursor nad elementem!');" >
</div>
<div id="przycisk"
  style="background-color: red; width:100px; height: 100px;"
  onmouseout="alert('kursor opuścił element!');" >
</div>
```

DAILY
GROUP

Zdarzenia onmouseover i onmouseout służą do wykrywania czy kursor myszy znalazł się lub opuścił dany element html.

Slajd 12

B2E
BUSINESS TO EDUCATION

SZCZECIŃSKI PARK
NAUKOWO-TECHNOLOGICZNY

JavaScript

Ćwiczenia

Napisz skrypt w taki sposób, aby pierwsze kliknięcie zmieniło tekst znajdujący się na warstwie na dowolny inny, drugie przywracało jego oryginalną zawartość, trzecie ponownie go zmieniło itd.

DAILY
GROUP

Napisz kod strony zawierającej jedną warstwę (element div). Kliknięcie tej warstwy powinno spowodować utworzenie nowej warstwy, która również będzie reagować na kliknięcie



Slajd 13

B2E
BUSINESS TO EDUCATION

SZCZECIŃSKI PARK
NAUKOWO-TECHNOLOGICZNY

JavaScript

Przykładowe rozwiązanie

```

<div id="content" style="background-color:silver;width:200px;height:60px;text-align:center;"
onclick="divClick();">To jest nasza strona </div>
<script type="text/javascript">
<!--
var str = "Nowy tekst";
function divClick()
{
    var element = document.getElementById("content");
    var tempStr = element.innerHTML;
    element.innerHTML = str;
    str = tempStr;
}
// -->
</script>

```

Napisz kod strony zawierającej jedną warstwę (element div). Kliknięcie tej warstwy powinno spowodować utworzenie nowej warstwy, która również będzie reagować na kliknięcie.

Slajd 14

B2E
BUSINESS TO EDUCATION

SZCZECIŃSKI PARK
NAUKOWO-TECHNOLOGICZNY

JavaScript


Ćwiczenia

Napisz kod strony zawierającej jedną warstwę (element div).
Kliknięcie tej warstwy powinno spowodować utworzenie nowej warstwy, która również będzie reagować na kliknięcie.


Napisz kod strony zawierającej jedną warstwę (element div). Kliknięcie tej warstwy powinno spowodować utworzenie nowej warstwy, która również będzie reagować na kliknięcie



Slajd 15



B2E
BUSINESS TO EDUCATION




SZCZECIŃSKI PARK
NAUKOWO-TECHNOLOGICZNY


JavaScript

Przykładowe rozwiązanie

```
<div id="content" style="background-color:silver;width:200px;height:20px;text-align:center;" >div>  
<script type="text/javascript">  
function div1Click() {  
    alert('Pierwsza warstwa została kliknięta');  
    if(div2Exists) return;  
    var divEl = document.createElement("div");  
    divEl.addEventListener('click', function(e) {  
        alert("Druga warstwa została kliknięta");}), true);  
    var str = "To jest druga warstwa."  
    var divElTextNode = document.createTextNode(str);  
    divEl.appendChild(divElTextNode);  
    var content = document.getElementById("content");  
    content.appendChild(divEl);  
    div2Exists = true;  
    }  
    var div1 = document.getElementById("content");  
    div1.addEventListener('click', div1Click, false);  
    // -->  
</script>
```



Slajd 16



B2E
BUSINESS TO EDUCATION



SZCZECIŃSKI PARK
NAUKOWO-TECHNOLOGICZNY

JavaScript

Ćwiczenia

Napisz skrypt, który zamieni wprowadzane znaki w polu formularza na duże.



Napisz skrypt, który zamieni wprowadzane znaki w polu formularza na duże.

Slajd 17

Można wykorzystać zdarzenie onkeyup – przez zmienną "this" mamy dostęp do wartości pola tekstowego

Ćwiczenia

Ćwiczenie zostało zaprezentowane na slajdzie 12. Wymaga ono od uczestnika opanowania sposobu definiowania obsługi zdarzeń i tworzenia nowych obiektów na stronie

Opis założonych osiągnięć ucznia

Uczestnik potrafi wymienić podstawowe zdarzenia na stronie, potrafi napisać funkcję obsługującą dowolne z tych zdarzeń.

Lekcja 8 Obsługa błędów

Cel lekcji

Celem lekcji jest przedstawienie mechanizmów obsługi błędów programu w języku JavaScript. Zaprezentowane zostaną podstawowe instrukcje służące do zapewnienia poprawnego wykonania programu.

Sposoby osiągnięcia celów kształcenia

Czynności nauczyciela	Czynności ucznia
wprowadza nowe pojęcia związane z tematem zajęć (materiał ze slajdów) przygotowuje ćwiczenia z zaprezentowanego materiału	Wykonuje zadane przez nauczyciela ćwiczenia na komputerze
Prowadzi z uczniami dyskusję na temat wybranych przez nich rozwiązań ćwiczeń	Prezentuje rozwiązania ćwiczeń nauczycielowi i pozostałym uczniom, tłumaczy wybrane przez siebie rozwiązanie
Podsumowuje lekcję, ocenia pracę uczniów na zajęciach.	Dokonuje samooceny swojej pracy, omawia z nauczycielem rozwiązania ćwiczeń

Treść – slajdy z opisem

Slajd 1



W każdym programie występują błędy. Nawet najlepsi programiści popełniają błędy. W języku JavaScript oprócz typowych błędów popełnianych przez programistę, możliwe są także błędy związane z różną interpretacją standardu ECMAScript w przeglądarkach internetowych. Np. funkcja, która zadziałała w przeglądarce Google Chrome, niekoniecznie musi działać tak samo w przeglądarce Firefox.

Do obsługi błędów w języku JavaScript wykorzystywane są przeniesione z języka C++ instrukcje try catch, oraz obiekty typu Error. Program może wskazać nieprawidłowe działanie przez zgłoszenie (rzucenie) wyjątku.

Slajd 2

Wyjątki (ang exceptions) to konstrukcje służące do obsługi błędów. Obsługują one sytuacje, w których program nie zadziałał tak jak założył programista. Np. użytkownik przekazał tekst w polu przeznaczonym na liczbę; nie zadziałało pobranie danych itp. Wyjątki są informacją od programu, że dana instrukcja się nie powiodła. Część wyjątków

zgłaszana jest przez funkcje systemowe (np. konwersje typów, próby odczytania nieistniejącej właściwości itp.), ale nic nie stoi na przeszkodzie aby zgłaszać własne typy wyjątków w tworzonych przez nas funkcjach i metodach. Służy do tego instrukcja throw, której postać to: throw wyrażenie. Spowoduje to wypisanie na konsoli przeglądarki tekstu zawartego w wyrażeniu.

Slajd 3

Zdarzenie onerror

Najprostszą metodą obsługi błędu to podłączenie funkcji do zdarzenia onerror

```

window.onerror = function(komunikat, url, linia) {
    instrukcje;
}
    
```

Najprostszą metodą obsługi błędu to podłączenie funkcji do zdarzenia onerror. Zdarzenie onerror przekazuje 3 parametry do funkcji obsługującej: komunikat o błędzie, adres url strony oraz numer linii w której wystąpił błąd.

Slajd 4

Obsługa błędów

Obiekt Error

```

throw new Error();
throw new Error("Wystąpił błąd!");
    
```

Screenshot of browser console showing an error message: Error: Wystąpił błąd!

Zamiast jednak stosować typy proste przy tworzeniu wyjątków należy skorzystać z obiektu opisującego błąd. W języku JavaScript jest to obiekt Error. Jako parametr wywołania konstruktora możemy podać treść komunikatu o błędzie. Utworzony obiekt ma 2 główne właściwości: name (to nazwa konstruktora wykorzystanego do utworzenia obiektu) oraz message – treść komunikatu podana w konstruktorze. Obiekt Error zawiera metodę toString(), która może się różnić w zależności od przeglądarki – Mozilla Firefox wyświetla domyślnie właściwości name i message oddzielone dwukropkiem.

Slajd 5

B2E BUSINESS TO EDUCATION

SZCZECIŃSKI PARK NAUKOWO-TECHNOLOGICZNY

JavaScript

Obsługa błędów

Instrukcje do przechwytywania błędów:

- try ... catch
- try ... catch ... finally
- zdarzenie onerror

DAILY GROUP

Wcześniejsze przykłady pokazywały sposoby zgłaszania wyjątków, które nie były w żaden sposób obsługiwane w naszym skrypcie. Wynikiem było więc wypisanie wyjątku na konsoli przeglądarki. Zamiast wypisywać komunikaty o błędach należy odpowiednio zareagować w kodzie naszego programu. W języku JavaScript do obsługi wyjątków wykorzystywane są instrukcje zapożyczone z innych języków programowania takich jak C++ i Java. Są to instrukcje try catch, try catch z dodatkową klauzulą finally, oraz zdarzenie onerror.

Slajd 6

B2E BUSINESS TO EDUCATION

SZCZECIŃSKI PARK NAUKOWO-TECHNOLOGICZNY

JavaScript

Instrukcja try ... catch

Instrukcja try ... catch ma następującą postać

```
try {
    instrukcje;
}
catch (wyjatek) {
    obsługa wyjątku
}
```

DAILY GROUP

Instrukcja try catch ma następującą postać:
w bloku try umieszczamy kod, który chcemy wykonać i w którym może wystąpić błąd, następnie umieszczamy instrukcję catch z parametrem, w którym zostanie przekazany obiekt naszego wyjątku. Dzięki temu będziemy mieć dostęp do typu wyjątku, i komunikatu, który został w nim zdefiniowany.



Slajd 7

 **B2E**
BUSINESS TO EDUCATION

 SZCZECIŃSKI PARK
NAUKOWO-TECHNOLOGICZNY

JavaScript

Instrukcja finally


Instrukcje try catch można rozszerzyć o opcjonalną instrukcję finally.


```
try {  
    instrukcje;  
}  
catch (wyjątek) {  
    kod obsługi wyjątku;  
}  
finally {  
    instrukcje;  
}
```

 DAILY
GROUP

Instrukcje try catch możemy rozszerzyć o opcjonalną instrukcję finally. Dodanie tej klauzuli spowoduje że instrukcje wpisane w tej klauzuli zostaną wykonane zawsze, nawet gdy zostanie zgłoszony wyjątek.

Slajd 8


 **B2E**
BUSINESS TO EDUCATION

 SZCZECIŃSKI PARK
NAUKOWO-TECHNOLOGICZNY

JavaScript

Zagnieżdżanie instrukcji try catch

```
try{  
    //instrukcje, które mogą spowodować wyjątek 1  
    try{  
        //instrukcje, które mogą spowodować wyjątek 2  
    }  
    catch (wyjątek2){  
        //obsługa wyjątku 2  
    }  
}  
catch (wyjątek1){  
    //obsługa 1 wyjątku  
}
```

 DAILY
GROUP

Instrukcje try catch można zagnieżdżać. Tzn, wewnątrz instrukcji try możemy umieścić kolejny blok obsługujący inny wyjątek naszego programu.

Slajd 9

Propagacja wyjątków

- W przypadku wystąpienia wyjątku jest on przekazywany do najbliższego bloku obsługi błędów. Jeśli funkcja nie zawiera obsługi błędów to wyjątek przekazywany jest do miejsca wywołania funkcji.
- Po obsłużeniu wyjątku działanie programu jest wznowiane.
- Jeśli program nie zawiera obsługi błędu, działanie całego skryptu jest zatrzymywane i informacja o błędzie wpisywana jest na konsoli przeglądarki

W momencie wystąpienia wyjątku wstrzymywane jest wykonywanie kodu skryptu, a sterowanie zostanie przekazane do najbliższego bloku obsługi wyjątku. Jeśli definicja funkcji nie zawiera obsługi błędów wyjątek jest przekazywany do miejsca wywołania funkcji. Po obsłużeniu błędu wykonywanie programu jest kontynuowane, a jeśli w całym skrypcie nie znajdzie się blok obsługi wyjątku to działanie skryptu zostanie przerwane i komunikat o błędzie będzie wypisany na konsoli.

Slajd 10

Predefiniowane obiekty wyjątków

Oprócz typu Error w trzeciej wersji specyfikacji ECMAScript wprowadzono dodatkowe typy błędów:


- EvalError
- RangeError
- ReferenceError
- SyntaxError
- TypeError
- URIError

Oprócz typu Error ECMAScript v3 definiuje jeszcze 6 innych: EvalError, RangeError, ReferenceError, SyntaxError, TypeError i URIError. Są dostępne od wersji 1.5 JavaScriptu. Sposób tworzenia obiektów tych typów jest taki sam jak obiektu Error.


- EvalError jest wykorzystywany w przypadku nieprawidłowego użycia funkcji eval.
- RangeError jest wykorzystywany , gdy wartość numeryczna przekracza dopuszczalny zakres.
- ReferenceError jest wykorzystywany przy próbie odczytu nieistniejących zmiennych.
- SyntaxError jest wykorzystywany po wykryciu błędu składniowego, np. przez metodę eval oraz konstruktory Function i RegExp.
- TypeError jest wykorzystywany , gdy wartość jest typu innego niż oczekiwany. Może tak być przy próbie dostępu do właściwości o wartości null bądź undefined, użycia operatora new i argumentu niebędącego konstruktorem, próbie wywołania nieistniejącej metody obiektu itp.
- URIError jest wykorzystywany przez metody kodujące bądź dekodujące adresy URI, gdy wykryte zostaną nieprawidłowo sformowane dane.



Slajd 11



B2E
BUSINESS TO EDUCATION



SZCZECIŃSKI PARK
NAUKOWO-TECHNOLOGICZNY


JavaScript

Ignorowanie błędów

```


try
{
  var ajax = new XMLHttpRequest()
}
catch(e1)
{
  try
  {
    ajax = new ActiveXObject("Msxml2.XMLHTTP")
  }
  catch(e2)
  {
    try
    {
      ajax = new ActiveXObject("Microsoft.XMLHTTP")
    }
    catch(e3)
    {
      ajax = false
    }
  }
}

```




W klauzuli catch nie musimy tylko informować użytkownika o napotkanym błędzie, tylko dodać np. inne instrukcje i kontynuować wykonanie programu. Jest to mechanizm przydatny przy tworzeniu wersji skryptu działającej w wielu przeglądarkach. W przykładzie pokazano utworzenie obiektu AJAX dla różnych przeglądarek, Jeśli korzystamy z nowszych przeglądarek wykonana zostanie instrukcja pierwsza, natomiast jeśli korzystamy z przeglądarki internet explorer 7 to 1 instrukcja zgłosi błąd i w obsłudze błędu spróbuje wykonać instrukcję tworzącą obiekt żądania za pomocą metody specyficznej dla internet explorera.

Slajd 12



B2E
BUSINESS TO EDUCATION




SZCZECIŃSKI PARK
NAUKOWO-TECHNOLOGICZNY

JavaScript

Ćwiczenie


Napisz skrypt pobierający od użytkownika informacje o dacie, w kolejnych pytaniach o rok, miesiąc i dzień (wykorzystaj instrukcję prompt, lub przygotowany przez siebie formularz). Zgłoś wyjątki, jeśli wprowadzone dane są niepoprawne (podane są wartości tekstowe, lub liczby wychodzą poza zakres).




Napisz skrypt pobierający od użytkownika informacje o dacie (wykorzystaj instrukcję prompt, lub przygotowany przez siebie formularz). Zgłoś wyjątki jeśli wprowadzone dane są niepoprawne. Wyjątki powinny być zgłoszone w przypadku podanie nieprawidłowej cyfry (w przypadku miesiąca spoza zakresu 1-12 itd.), w przypadku wpisania w pole tekstu. Jeśli data zostanie wprowadzona poprawnie wypisz ją na ekranie.



Slajd 13



B2E
BUSINESS TO EDUCATION




SZCZECIŃSKI PARK
NAUKOWO-TECHNOLOGICZNY


JavaScript

Przykładowe rozwiązanie


```
function pobierzDane() {  
  try {  
    var dzien = prompt("Podaj dzień");  
    if (isNaN(dzien) || Number(dzien)<0 || Number(dzien) > 31) throw new  
    Error("Nieprawidłowy dzień");  
    var miesiac = prompt("Podaj miesiac");  
    if (isNaN(miesiac) || Number(miesiac)<0 || Number(miesiac) > 12) throw new  
    Error("Nieprawidłowy miesiac");  
    var rok = prompt("Podaj rok");  
    if (isNaN(rok) || Number(rok)<2000 || Number(rok) > 2020) throw new  
    Error("Nieprawidłowy rok");  
    alert(rok+"-"+miesiac+"-"+dzien);  
  } catch (error) {  
    alert(error.toString());  
  }  
}
```



Slajd 14



B2E
BUSINESS TO EDUCATION




SZCZECIŃSKI PARK
NAUKOWO-TECHNOLOGICZNY

JavaScript

Ćwiczenie


Zmodyfikuj poprzedni przykład dodając własny wyjątek i dodaj jego wykorzystanie.




Zmodyfikuj poprzedni przykład dodając własny wyjątek i dodaj jego wykorzystanie.



Slajd 15



B2E
BUSINESS TO EDUCATION




SZCZECIŃSKI PARK
NAUKOWO-TECHNOLOGICZNY

JavaScript


Przykładowe rozwiązanie

```
function Bład(type,msg) {
    this.type = type;
    this.message = msg;
    this.toString = function() {
        return "Nasz blad to " + this.message+ "typ: "+ this.type;
    }
}

var dzien = prompt("Podaj dzień");
if (isNaN(dzien) || Number(dzien)<0 || Number(dzien) > 31) throw new
Bład("Nasz", "Nieprawidłowy dzień");
```




KAPITAŁ LUDZKI
NARODOWA STRATEGIA SPÓJNOŚCI



UNIA EUROPEJSKA
EUROPEJSKI
FUNDUSZ SPOŁECZNY

Projekt współfinansowany przez Unię Europejską w ramach Europejskiego Funduszu Społecznego



DAILY
GROUP

Ćwiczenie

Napisz skrypt pobierający od użytkownika informacje o dacie (wykorzystaj instrukcję prompt, lub przygotowany przez siebie formularz). Zgłoś wyjątki jeśli wprowadzone dane są niepoprawne.

Wyjątki powinny być zgłoszone w przypadku podanie nieprawidłowej cyfry (w przypadku miesiąca spoza zakresu 1-12 itd.), w przypadku wpisania w pole tekstu.

Jeśli data zostanie wprowadzona poprawnie wypisz ją na ekranie.

Ćwiczenie obrazuje praktyczny sposób wykorzystania instrukcji poznanych w czasie lekcji.

Opis założonych osiągnięć ucznia

Uczestnik potrafi dodać do skryptu metody obsługi błędów.

Lekcja 9 Wykorzystanie języka JavaScript

Cel lekcji

Po wprowadzeniu podstawowych elementów języka JavaScript czas pokazać do czego można ten język wykorzystać. W lekcji zostaną przedstawione podstawowe zastosowania języka JavaScript – takie jak walidacja przesyłanych formularzy, modyfikacja elementów CSS, operacje na łańcuchach znaków.

Sposoby osiągnięcia celów kształcenia

Czynności nauczyciela	Czynności ucznia
wprowadza nowe pojęcia związane z tematem zajęć (materiał ze slajdów) przygotowuje ćwiczenia z zaprezentowanego materiału	Wykonuje zadane przez nauczyciela ćwiczenia na komputerze

Prowadzi z uczniami dyskusję na temat wybranych przez nich rozwiązań ćwiczeń	Prezentuje rozwiązania ćwiczeń nauczycielowi i pozostałym uczniom, tłumaczy wybrane przez siebie rozwiązanie
Podsumowuje lekcję, ocenia pracę uczniów na zajęciach.	Dokonuje samooceny swojej pracy, omawia z nauczycielem rozwiązania ćwiczeń

Treść – slajdy z opisem

Slajd 1

W lekcji omówimy najczęstsze sposoby wykorzystania języka JavaScript.
-zmiana elementów css
-operacje na łańcuchach znaków
-wykorzystanie do walidacji formularzy
Zobaczymy, jak wykorzystać zaprezentowane na wcześniejszych lekcjach elementy języka JavaScript.

Slajd 2

Style CSS to ważny element każdej nowoczesnej witryny WWW. Za pomocą JavaScriptu możliwa jest dynamiczna zmiana właściwości arkuszy styli.
Style CSS danego elementu przechowywane są w atrybucie „style” danego elementu. Dostęp do tego atrybutu możliwy jest z wykorzystaniem metod `getAttribute`, `setAttribute` i `removeAttribute`. Metoda `getAttribute(„style”)` pobiera wartość atrybutu `style` i umieszcza w zmiennej `styl`.
Aby ustawić styl dla obiektu wywołujemy metodę `setAttribute` z dwoma parametrami. Pierwszym jest nazwa atrybutu (w naszym przypadku `style`) a drugim jego wartość.
Usunięcie stylu możliwe jest za pomocą metody `removeAttribute`.



Slajd 3



B2E
BUSINESS TO EDUCATION



SZCZECIŃSKI PARK
NAUKOWO-TECHNOLOGICZNY

JavaScript

Obiekt style

każdy obiekt reprezentujący element strony zawiera właściwość style.

Jest ona odwzorowaniem atrybutu html style

<pre style="margin: 0;"><div style=" background-color: blue; width: 100px; height: 100px;"> </div></pre>	<pre style="margin: 0;">element.style.backgroundColor = "blue"; element.style.width = "100px"; element.style.height = "100px";</pre>
--	--



Wykorzystanie atrybutu style jest wygodne jeśli chcemy zmodyfikować całą jego zawartość. Jeśli jednak chcemy zmienić tylko jedną właściwość (np. kolor tła) pobranie całego atrybutu będzie wymagało od nas obsłużenia także innych właściwości, ustawionych dla tego elementu. Będziemy musieli wyszukać w atrybucie aktualny kolor tła, a przy zapisie zmodyfikowanej wartości pamiętać także o pozostałych ustawionych właściwościach.

Na szczęście w JavaScript każdy element ma swój obiekt style (który jest odwzorowaniem atrybutu style).


Jeśli w zmiennej obiekt przechowujemy referencje do elementu witryny do dostęp do obiektu style możliwy jest przez wypisanie obiekt.style.

Atrybuty css danego elementu są odwzorowywane do właściwości obiektu style w następujący sposób: jeśli atrybut elementu nazywał się background-color to jego reprezentacja w JavaScript będzie wyglądała następująco:


backgroundColor



Slajd 4



B2E
BUSINESS TO EDUCATION




SZCZECIŃSKI PARK
NAUKOWO-TECHNOLOGICZNY

JavaScript

Właściwość className

- odzwierciedla atrybut class elementu html

```
<div id="przycisk" class="niebieski"></div>
var element = document.getElementById("przycisk");
element.className = "zielony";
```



DAILY
GROUP


Do tej pory zmienialiśmy tylko konkretne atrybuty stylu przypisanego do danego elementu. Stosując JavaScript możemy również zmienić wartość atrybutu class elementu html, a co za tym idzie zmienić cały styl przypisany do tego elementu.

Atrybut class zamieniany jest na właściwość className obiektu odpowiadającego danemu znacznikowi.


Aby zmodyfikować właściwość class elementu należy wyszukać go za pomocą metody `document.getElementById()` a następnie do jego właściwości `className` przypisać nową wartość.



Slajd 5



B2E
BUSINESS TO EDUCATION



SZCZECIŃSKI PARK
NAUKOWO-TECHNOLOGICZNY

JavaScript

Właściwość className

```


<html><head><style type="text/css">
  .zielony { background-color: green; }
  .niebieski { background-color: blue; }
</style>
<script type="text/javascript">
function zmienKlase() {
var element = document.getElementById("przycisk");
element.className="niebieski"; }
</script></head><body>
<div id="przycisk" class="zielony" style="width: 100px; height: 100px;"></div>
<input type="button" value="zmień" onclick="zmienKlase();"></input>
</body></html>

```




Aby zmodyfikować właściwość class elementu, należy wyszukać go za pomocą metody `document.getElementById()`, a następnie do jego właściwości `className` przypisać nową wartość.

Slajd 6



B2E
BUSINESS TO EDUCATION




SZCZECIŃSKI PARK
NAUKOWO-TECHNOLOGICZNY

JavaScript

Ćwiczenia

Umieść na stronie przycisk. Niech po kliknięciu zmienia się jego kolor tła i kolor tekstu.
Wykorzystaj obie metody (modyfikacje obiektu style) i zmianę klasy css





Ćwiczenie

Umieść na stronie przycisk. Niech po kliknięciu zmienia się jego kolor tła i kolor tekstu.
Wykorzystaj obie metody (modyfikacje obiektu style) i zmianę klasy css.




Slajd 7



JavaScript

Przykładowe rozwiązanie

```
<html><head><style type="text/css">
  .zielony { background-color: green; }
  .niebieski { background-color: blue; }
</style>
<script type="text/javascript">
function zmienKlase() {
var element = document.getElementById("przycisk");
element.className="niebieski";
}
</script></head>
<body>
<div id="przycisk" class="zielony" style="width: 100px; height: 100px;" ></div>
<input type="button" value="zmień" onclick="zmienKlase();" ></input>
</body></html>
```




Slajd 8

JavaScript

Ćwiczenia

Umieść na stronie przycisk. Niech najechanie na niego zmieni kolor jego tła i tekstu.




Ćwiczenie

Umieść na stronie przycisk. Niech najechanie na niego zmieni kolor jego tła i tekstu.



Slajd 9



B2E
BUSINESS TO EDUCATION




SZCZECIŃSKI PARK
NAUKOWO-TECHNOLOGICZNY


JavaScript

Przykładowe rozwiązanie


```
<html><head><style type="text/css">
  .zielony { background-color: green; }
  .niebieski { background-color: blue; }
</style><script type="text/javascript">
function zmien() {
  var element = document.getElementById("przycisk");
  element.style.backgroundColor = 'blue';
  element.style.color = 'white'; }
function wroc() {
  var element = document.getElementById("przycisk");
  element.style.backgroundColor = 'green';
  element.style.color = 'black';}
</script></head><body>
<div id="przycisk" class="zielony" style="width: 100px; height: 100px;"
onmouseover="zmien()" onmouseout="wroc()" ></div>
</body></html>
```



Slajd 10



B2E
BUSINESS TO EDUCATION



SZCZECIŃSKI PARK
NAUKOWO-TECHNOLOGICZNY


JavaScript

Ćwiczenia

Napisz skrypt, który zmieni kolor tła aktualnie zaznaczonego pola w formularzu na żółty i z powrotem na biały po jego odznaczeniu


Ćwiczenie

Napisz skrypt, który zmieni kolor tła aktualnie zaznaczonego pola w formularzu na żółty i z powrotem na biały po jego odznaczeniu.





Slajd 11



Slide 11 content: Includes logos for B2E (Business to Education) and Szczeciński Park Naukowo-Technologiczny. The title is "JavaScript" and the main heading is "Przykładowe rozwiązanie". It contains JavaScript code for highlighting and unhighlighting text, and an HTML input field with event handlers.

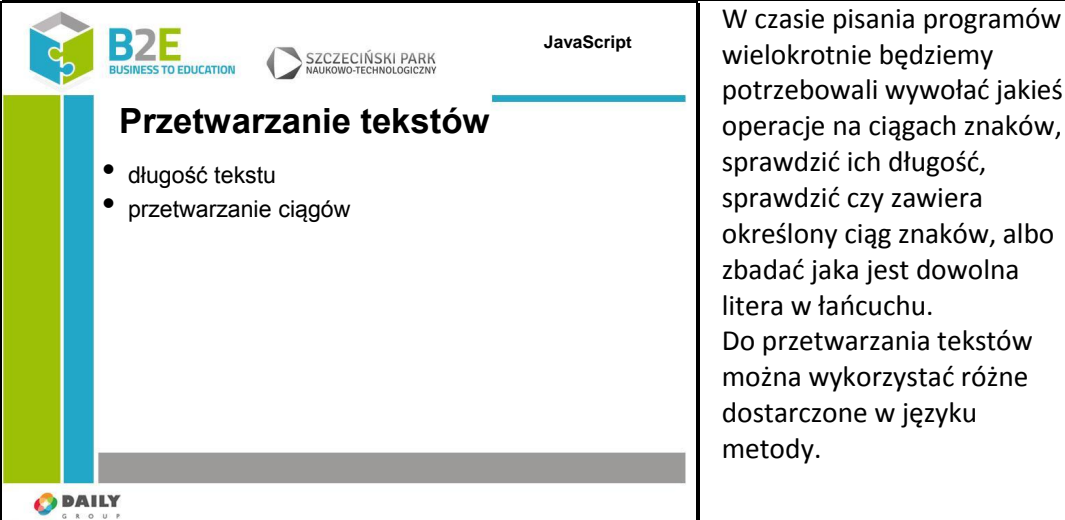
JavaScript

Przykładowe rozwiązanie

```
function zaznacz(x) {  
  x.style.background="yellow";  
}  
function odznacz(x) {  
  x.style.background="white";  
}  
(...)  
<input type="text" onfocus="zaznacz(this)"  
  onblur="odznacz(this)">
```

DAILY GROUP

Slajd 12



Slide 12 content: Includes logos for B2E and Szczeciński Park Naukowo-Technologiczny. The title is "JavaScript" and the main heading is "Przetwarzanie tekstów". It lists two bullet points: "długość tekstu" and "przetwarzanie ciągów". A text box on the right explains that during program writing, we often need to perform operations on strings, such as checking their length, checking for specific characters, or checking for a specific letter in a string. It also mentions that various methods are available in JavaScript for text processing.

JavaScript

Przetwarzanie tekstów

- długość tekstu
- przetwarzanie ciągów

W czasie pisania programów wielokrotnie będziemy potrzebowali wywołać jakieś operacje na ciągach znaków, sprawdzić ich długość, sprawdzić czy zawiera określony ciąg znaków, albo zbadać jaka jest dowolna litera w łańcuchu. Do przetwarzania tekstów można wykorzystać różne dostarczone w języku metody.

DAILY GROUP

Slajd 13

JavaScript

Jak sprawdzić długość tekstu?

Każdy łańcuch znaków w JavaScript jest obiektem. Obiekt string zawiera m.in. właściwość length – liczbę zawartych w nim znaków

```
var tekst = "Witaj świecie!";
var dlugosc = tekst.length;
alert("Długość tekstu to: "+dlugosc);
```

DAILY GROUP

Każdy łańcuch znaków w JavaScript jest obiektem. Obiekt string zawiera m.in. właściwość length – liczbę zawartych w nim znaków. Powyższy przykład po uruchomieniu pokaże komunikat: "Długość tekstu to 14".

Slajd 14

JavaScript

Przetwarzanie ciągów


- charAt tekst.charAt(indeks)
- charCodeAt tekst.charCodeAt(indeks)
- concat tekst.concat(tekst2...tekstN)
- fromCharCode String.fromCharCode(kod1...kodN)
- indexOf tekst.indexOf(wartość [,indeks])
- lastIndexOf tekst.lastIndexOf(wartość [, indeks])
- match tekst.match(wyrażenieReg)
- replace tekst.replace(wyrażenieReg, nowy)

DAILY GROUP


charAt(indeks) – zwraca znak znajdujący się pod wybranym indeksem
 charCodeAt(indeks) - zwraca kod znaku znajdującego się pod wybranym indeksem (w JS 1.2 – kod ISO Latin1 w późniejszych Unicode)
 concat – łączy łańcuchy znakowe. Wynikiem jest łańcuch powstały z połączenia tekstu oryginalnego i wszystkich tekstów podanych jako parametry
 fromCharCode – zwraca łańcuch znaków złożony z podanych jako parametry kodów.
 indexOf – zwraca indeks wystąpienia łańcucha wartość w łańcuchu tekst. Jeśli podamy parametr indeks to przeszukiwanie rozpoczyna się od tego indeksu.
 lastIndexOf – zwraca indeks ostatniego wystąpienia łańcucha wartość w łańcuchu tekst.
 match – zwraca część ciągu tekst pasującego do wyrażenia regularnego, jeśli nie znajdzie dopasowania zwraca null
 replace – zwraca ciąg znaków, w którym fragmenty opisane przez wyrażenie zostały zamienione na nowy tekst



Slajd 15



B2E
BUSINESS TO EDUCATION




SZCZECIŃSKI PARK
NAUKOWO-TECHNOLOGICZNY

JavaScript

Przetwarzanie ciągów

- search tekst.search(wyrażenieReg)
- slice tekst.slice(start [,koniec])
- split tekst.split([separator [,limit]])
- substr tekst.substr(indeks [,liczba])
- substring tekst.substring(od, do)
- toLowerCase tekst.toLowerCase()
- toUpperCase tekst.toUpperCase()



search – Sprawdza czy ciąg opisany przez wyrażenie występuje w tekście. Jeśli tak to zwracany jest indeks miejsca wystąpienia a w przeciwnym wypadku wartość -1

slice – zwraca podciąg tekstu od indeksu start do końca lub jeśli podaliśmy drugi parametr do wartości drugiego parametru.

split – dzieli ciąg znaków na podciągi względem parametru separator. Jeśli nie podamy separatora to zwracany jest cały ciąg. Parametr limit oznacza maksymalną liczbę zwróconych elementów. Wynikiem jest tablica zawierająca elementy o wartościach uzyskanych po podziale.

substr – zwraca podciąg tekstu, zaczynający się od pozycji oznaczonej przez indeks. Jeśli podamy parametr liczba – zostanie zwrócona taka liczba znaków, jeśli nie podamy tego parametru zostaną zwrócone wszystkie znaki do końca łańcucha.


substring – zostanie zwrócony podciąg rozpoczynający się na pozycji od i kończący na indeksie o pozycji do.


toLowerCase - wszystkie litery w tekście zostaną zmienione na małe.

toUpperCase – wszystkie litery w tekście zostaną zamienione na wielkie.



Slajd 16

 **B2E**
BUSINESS TO EDUCATION

 SZCZECIŃSKI PARK
NAUKOWO-TECHNOLOGICZNY

JavaScript


Przetwarzanie ciągów - przykłady

- concat


```
var t1 = "test".concat("java", "script");  
"testjavascript"
```
- indexOf


```
var indeks = "testjavascript".indexOf("va");  
6
```
- match

```
var tekst = "testjavascript".match("java");  
java
```

 DAILY
GROUP

Slajd 17

 **B2E**
BUSINESS TO EDUCATION

 SZCZECIŃSKI PARK
NAUKOWO-TECHNOLOGICZNY


JavaScript

Przetwarzanie ciągów - przykłady

- replace

```
var tekst = "Witaj %IMIE%!".replace("%IMIE%", "Tomek");  
"Witaj Tomek!"
```
- split

```
var tablica = "a,b,c,d".split(",");  
[a,b,c,d]
```


 DAILY
GROUP


replace - Metoda zamienia wystąpienie zgodne z wyrażeniem regularnym podanym jako 1 argument na tekst podany jako 2 argument.

split - dzieli tekst na fragmenty względem separatora podanego jako argument, opcjonalnie można podać maksymalną liczbę dopasowanych elementów



Slajd 18

**B2E**
BUSINESS TO EDUCATION

SZCZECIŃSKI PARK
NAUKOWO-TECHNOLOGICZNY


JavaScript

Wprowadzanie danych

- formularze

```
<form name="formularz" action="submit.php" method="POST">  
.....  
  <input type="submit" value="wyślij" />  
</form>
```
- sprawdzanie poprawności danych

```
<form name="formularz" action="submit.php"  
  onsubmit="return sprawdz()" method="POST">  
.....  
  <input type="submit" value="wyślij" />  
</form>
```



Formularz to element strony, który tworzymy za pomocą znacznika form. Dostęp do formularzy możliwy jest przez tablicę forms obiektu document. Formularz służy do przesłania zestawu danych od użytkownika do serwera. Warto wykorzystać JavaScript aby dokonać wstępnej analizy danych wprowadzonych do formularza. Sprawdzenie formularza powinno odbywać się również po stronie serwera, ale nie ma sensu niepotrzebnie obciążać serwera jeśli dane w formularzu są błędnie wprowadzone. Zobaczmy w jaki sposób możemy wykorzystać napisaną przed chwilą funkcję.



Slajd 19



B2E
BUSINESS TO EDUCATION



SZCZECIŃSKI PARK
NAUKOWO-TECHNOLOGICZNY

JavaScript

Sprawdzanie poprawności danych

```

<form name="logowanie" action="login.php" method="POST">
  <input type="text" name="login">
  <input type="password" name="password">
  <input type="button" value="Zaloguj"
onclick="sprawdz()" >
</form>
function sprawdz() {
  var loginForm = document.forms["logowanie"];
  if (loginForm.login == "" || loginForm.password == "")
  {
    alert("Nie podano nazwy użytkownika lub
hasła");
    return; }
  loginForm.submit();
}

```





Możemy sprawdzić formularz na dwa sposoby.

Po pierwsze możemy nie umieszczać w nim przycisku typu submit tylko zwykły przycisk typu button i w funkcji obsługującej zdarzenie onclick umieścić sprawdzenie danych i wysyłanie formularza jeśli dane są poprawne.

Nasz przykładowy formularz zawiera 2 pola tekstowe do wprowadzenia nazwy użytkownika i hasła oraz przycisk "Zaloguj". Po kliknięciu przycisku sprawdzane jest czy pola login i password zostały wypełnione (są różne od znaku pustego) i jeśli tak to formularz jest wysyłany przez wywołanie metody submit(). Jeśli którekolwiek z pól jest puste, wyświetlony jest komunikat i działanie funkcji zostanie przerwane.



Slajd 20




JavaScript

Sprawdzanie poprawności danych

```

<form name="logowanie" action="login.php" method="POST"
onsubmit="return sprawdz()">
  <input type="text" name="login">
  <input type="password" name="password">
  <input type="submit" value="Zaloguj" >
</form>



function sprawdz() {
  var loginForm = document.forms["logowanie"];
  if (loginForm.login == "" || loginForm.password == "")
  {
    alert("Nie podano nazwy użytkownika lub hasła");
    return false; }
  return true;
}
    
```



Możemy sprawdzić formularz na dwa sposoby.


Drugim sposobem jest podłączenie obsługi zdarzenia submit do formularza. Ogólna postać jest funkcji jest bardzo podobna. W obsłudze zdarzenia submit wpisaliśmy wywołanie return sprawdz(). Formularz zostanie przesłany do serwera tylko w przypadku gdy funkcja sprawdz zwróci jako wynik true. Stanie się tak, jeśli pola login i password zostaną wypełnione.

Slajd 21



JavaScript


Sprawdzenie poprawności danych

Sprawdzenie poprawności danych można wywołać nie tylko przy wysyłaniu formularza, ale nawet po wprowadzeniu zmian w pojedynczym polu formularza. Służy do tego zdarzenie onchange(). Wykorzystując to zdarzenie a także możliwość wywołania zdarzenia focus() możemy wymusić na użytkowniku wprowadzenie poprawnych wartości do pola tekstowego.






Slajd 22



B2E
BUSINESS TO EDUCATION



SZCZECIŃSKI PARK
NAUKOWO-TECHNOLOGICZNY


JavaScript

Przykład

```

<form name="logowanie" >
<input type="text" name="pesel"
onchange="sprawdzPesel();" >
</form>

function sprawdzPesel() {
    var form1 = document.forms["logowanie"];
    if (form1.pesel.value.length != 11) {
        alert("Nieprawidłowy numer PESEL");
        form1.pesel.focus();
    }
}
                
```




Jeśli wpisujemy liczbę znaków różną od 11, pojawi się komunikat i pole pesel zostanie ustawione jako aktywne.

Co się jednak stanie, gdy ponownie opuścimy pole formularza (bez dokonywania poprawek)?


Q: Jak możemy temu zapobiec?

A: wykorzystując zdarzenie onBlur

Slajd 23



B2E
BUSINESS TO EDUCATION




SZCZECIŃSKI PARK
NAUKOWO-TECHNOLOGICZNY

JavaScript

Ćwiczenie

Napisz funkcję javascript sprawdzającą poprawność wpisanego w formularzu adresu e-mail



Ćwiczenie:

Napisz funkcję javascript sprawdzającą poprawność wpisanego w formularzu adresu e-mail

Jako poprawny adres email uznajemy adres:



- zawiera tylko 1 znak @
- znak @ nie jest pierwszym elementem ciągu
- po znaku @ występuje cyfra lub litera.

• w ciągu po @ występuje znak . ale nie jest na ostatniej pozycji

Do napisania algorytmu można wykorzystać następujące metody (indexOf, lastIndexOf, match, search)




Slajd 24



JavaScript

Przykładowe rozwiązanie

```
function sprawdzEmail(email) {  
    var malpa = email.indexOf("@");  
    var kropka = email.lastIndexOf(".");  
    if (malpa < 1 || kropka < malpa + 2 ||  
        kropka + 1 >= email.length) {  
        alert("Nieprawidłowy adres email");  
        return false;  
    }  
}
```




Slajd 25

JavaScript

Wyrażenia regularne

- Wzorce opisujące łańcuchy tekstów
- Pozwalają na określenie czy podany ciąg znaków pasuje do wzorca.
- Mogą wyszukiwać w tekście wystąpienia wzorca.

W JavaScript reprezentowane są przez obiekt RegExp



Wyrażenia regularne to wzorce opisujące łańcuchy tekstów. Wyrażenia regularne mogą określać zbiór pasujących łańcuchów albo wskazywać istotne części łańcucha znaków. Istnieją algorytmy pozwalające sprawdzić czy podany ciąg znaków pasuje do wzorca. Za pomocą wyrażeń regularnych można wyszukiwać w tekście wystąpienia wzorca. Wyrażenia regularne są reprezentowane w JavaScript za pomocą obiektów typu RegExp.

Slajd 26

Obiekt RegExp

Tworzenie obiektów RegExp

- `var zmienna = new RegExp(wzorzec, atrybuty)`
- `var zmienna = /wzorzec/atributy`

```
var wzorzec = /JavaScript/
wzorzec.test("To jest kurs JavaScript");
```

Wyrażenia regularne są reprezentowane w JavaScript za pomocą obiektów typu RegExp.

Obiekt może zostać utworzony na dwa sposoby

-przez jawne wykorzystanie konstruktora obiektu (`var zmienna = new RegExp(wzorzec, atrybuty)`) przez przypisanie wyrażenia do zmiennej

Utwórzmy prosty wzorzec – niech to będzie ciąg znaków Javascript. Aby przetestować czy w tekście znajduje się ustalony przez nas wzorzec wywołujemy metodę test obiektu RegExp. Metoda zwraca true jeśli wzorzec występuje w tekście i false w przeciwnym przypadku. Obiekt RegExp ma jeszcze kilka innych metod:

Slajd 27

Metody obiektu RegExp


- `test(tekst) var wynik = wzorzec.test(tekst);`
- `compile(wyrażenie [, atrybuty])`
- `exec(tekst) var wynik = wzorzec.exec(tekst);`

Metoda test bada czy w zadanym jako argument tekście znajduje się wzorzec, zdefiniowany przez obiekt typu RegExp. Wynikiem jest true lub false


Metoda compile() dokonuje wewnętrznej kompilacji wyrażenia, co pozwala na szybsze jego przetwarzanie. Wyrażenie może zawierać argumenty i – pomijanie wielkości liter, g – globalne, m- przetwarzanie ma dotyczyć ciągu zawierającego wiele wierszy.

exec – metoda przeszukuje tekst w poszukiwaniu wzorca. Zwraca tablicę z wynikami dopasowań.

Slajd 28



B2E
BUSINESS TO EDUCATION



SZCZECIŃSKI PARK
NAUKOWO-TECHNOLOGICZNY


JavaScript

Tworzenie wyrażeń

Atrybuty (flagi) wyrażeń regularnych

- g
- i
- m

```
var wzorzec = new RegExp("javascipt", "i");
var wzorzec = /javascipt/i;
```



Wspomnieliśmy przed chwilą, że kompilowane wyrażenie może zawierać atrybuty, oraz że obie metody tworzenia wyrażeń także takie miejsce mają.

W wyrażeniach regularnych możemy podawać atrybuty (zwane też flagami) pozwalające na zmianę zachowania wyrażenia.


Dostępne są następujące parametry:

-g – globalny – oznaczający że przetwarzany ma być cały ciąg (bez tego znacznika) zwracany będzie tylko pierwszy dopasowany fragment. Parametr o bardzo dużym znaczeniu przy poszukiwaniu wszystkich wystąpień wzorca w tekście, można go pominąć jeśli interesuje nas tylko czy tekst zawiera fragment opisany wzorcem


-i – oznaczający że wielkość liter w tekście ma być ignorowana. Bez podania tego znacznika domyślną wartością jest false – treść w tekście musi mieć taką samą wielkość liter jak ta zdefiniowana we wzorcu

-m – oznaczający, że przetwarzanie ma dotyczyć tekstu wielowierszowego. W trybie tym znak początku i końca wzorca (^\$) jest wstawiany przed i po znaku nowej linii (\n).

Slajd 29



B2E
BUSINESS TO EDUCATION



SZCZECIŃSKI PARK
NAUKOWO-TECHNOLOGICZNY


JavaScript

Tworzenie wyrażeń

Wyrażenia budowane są ze znaków zwykłych (litery alfabetu i cyfry) i znaków specjalnych (większość znaków interpunkcyjnych) takich jak:

`^ $. * + ? = ! : | \ / () [] { }`

Aby wykorzystywać znaki specjalne w wyrażeniach należy poprzedzić je znakiem lewego ukośnika (\)



Wyrażenia budowane są ze znaków zwykłych oraz znaków specjalnych. Znaki zwykłe to wszystkie litery alfabetu i cyfry. Znaki specjalne to większość znaków interpunkcyjnych takich jak: `^ $. * + ? = ! : | \ / () [] { }`. Aby z nich korzystać należy podać przed nimi znak `\` (lewy ukośnik)

Slajd 30

Tworzenie wyrażeń

W wyrażeniach regularnych wprowadzono pojęcie klasy znakowej.

Klasa znakowa to zdefiniowany zestaw znaków, aby ją zbudować korzystamy z operatora zakresu [].

```
var wyrażenie = /[a-zA-Z]/
```

W wyrażeniach regularnych wprowadzono pojęcie klasy znakowej.

Klasa znakowa to zdefiniowany zestaw znaków, aby ją zbudować korzystamy z operatora zakresu [].

Istnieją klasy predefiniowane które zawierają definicje często wykorzystywanych klas

Slajd 31

Klasy znakowe

- [...] Dowolny znak znajdujący się w nawiasie
- [^...] Dowolny znak nieznajdujący się w nawiasie
- . Dowolny znak
- \d – dowolna cyfra
- \D – dowolny znak inny niż cyfra
- \s – dowolny biały znak (tabulator, spacja)
- \S – dowolny znak niebędący białym znakiem
- \w – dowolny znak wyrazu złożonego ze znaków ASCII
- \W – dowolny znak niebędący znakiem słowa złożonego ze znaków ASCII

[...] Dowolny znak znajdujący się w nawiasie – lista znaków podana w nawiasie jest listą znaków, które mogą wystąpić na tej pozycji w tekście

[^...] Dowolny znak nieznajdujący się w nawiasie – odwrotność – na pozycji w tekście nie mogą wystąpić znaki z nawiasu. Operator ^ musi być pierwszym po nawiasie

. Dowolny znak – na tej pozycji może być dowolny znak oprócz znaku nowej linii

\d – dowolna cyfra – uproszczony zapis [0-9]

\D – dowolny znak inny niż cyfra – odwrotność powyższego [^0-9]

\s – dowolny biały znak (tabulator, spacja)

\S – dowolny znak niebędący białym znakiem

\w – dowolny znak wyrazu złożonego ze znaków ASCII – dowolna cyfra, litera lub znak podkreślenia [0-9a-zA-Z_]

\W – dowolny znak niebędący znakiem słowa złożonego ze znaków ASCII [^0-9a-zA-Z_]

Slajd 32

B2E
BUSINESS TO EDUCATION

SZCZECIŃSKI PARK
NAUKOWO-TECHNOLOGICZNY

JavaScript

Powtórzenie wzorca

Często zdarza się sytuacja, że fragment wzorca powinien się powtórzyć. Istnieją odpowiednie elementy, które umieszczone w wyrażeniu oznaczają ile razy może pojawić się dany element znajdujący się przed nim.

- ?
- *
- +

ab?c+d*

DAILY GROUP

Często zdarza się sytuacja, że fragment wzorca powinien się powtórzyć. Istnieją odpowiednie elementy, które umieszczone w wyrażeniu oznaczają ile razy może pojawić się dany element, znajdujący się przed nim.

? – oznacza że element może pojawić się 0 lub jeden raz

* - oznacza że element może pojawić się dowolną ilość razy (zero lub więcej)

+ - oznacza, że element może pojawić się jeden lub więcej razy

W powyższym wyrażeniu dopasowane zostaną fragmenty tekstu, które: zaczynają się od znaku "a", po nich może ale nie musi wystąpić litera "b", następnie będzie co najmniej jedna litera c po których może wystąpić dowolna liczba liter d.

Slajd 33

B2E
BUSINESS TO EDUCATION

SZCZECIŃSKI PARK
NAUKOWO-TECHNOLOGICZNY

JavaScript

Powtórzenie wzorca

Dodatkowo dostępne są następujące operatory

- {n} – element ma się powtórzyć n razy
- {n,} – element ma się powtórzyć co najmniej n razy
- {n,m} – element ma się powtórzyć co najmniej n razy, ale nie więcej niż m razy

Przykład: kod pocztowy

```
var kod = ^\d{2}-\d{3}/
/[0-9][0-9]-[0-9][0-9][0-9]/
```



DAILY GROUP

Dodatkowo dostępne są następujące operatory

- {n} – element ma się powtórzyć n razy
- {n,} – element ma się powtórzyć co najmniej n razy
- {n,m} – element ma się powtórzyć co najmniej n razy, ale nie więcej niż m razy



Slajd 34




JavaScript

Kwantyfikatory zachłanne i leniwe

- zachłanne – próbują dopasować się do jak najdłuższych podciągów znaków

```
var wyrażenie = /<.*>/
var wynik = wyrażenie.exec("aaa <br> bbb <br> ccc");
wyrażenie = /<[^>]*>/
```

- leniwe – powstają przez dodanie dodatkowego znaku zapytania (??, *?, +?, {n}?, {n,m}?)



Wszystkie opisane wcześniej kwantyfikatory określane są jako zachłanne- będą próbowały dopasować się do jak najdłuższych podciągów. Rozważmy taki przykład – wyrażenie ma wyszukiwać znaczniki html w tekście. Wiemy że znaczniki to fragmenty tekstu ujęte w nawiasy trójkątne. Spodziewamy się wyników

 ale wyrażenie dopasowało jeszcze tekst
 bbb
. Fragment ten także pasuje do naszego wyrażenia, ale nie jest poprawnym znacznikiem html. Aby to poprawić należałoby użyć następującego wyrażenia: /<[^>]*>/ określi ono, że między nawiasami nie może znaleźć się nawias zamykający. Innym sposobem jest wykorzystanie kwantyfikatorów leniwych, powstałych przez dodanie do zwykłych kwantyfikatorów dodatkowego pytajnika.

Slajd 35



JavaScript

Grupy

```
(abc)
(abc (123)?)
aaa (bbb|ccc|ddd)
;([^;]+)\1;
Otestabc1testxyz
\d[a-z]{4}(?=abc)
\d[a-z]{4}(?!=abc)
```



Wcześniejsze metody pokazywały jak określić liczbę powtórzeń określonego znaku. Często sytuacją jest jednak powtarzanie się we wzorcu grup znaków. Aby oznaczyć grupę należy wpisać ją w nawiasy okrągłe. Pierwszy przykład pokazuje grupę składającą się z sekwencji abc. Drugi grupę składającą się z sekwencji abc, po której może wystąpić sekwencja cyfr

123. Cała grupa może pojawić się raz lub więcej.

Możemy także zdefiniować kilka różnych wersji ciągu. W przykładzie trzecim pokazano sekwencję aaa po której może wystąpić albo sekwencja bbb, ccc lub ddd. Takie warianty należy oddzielić od siebie znakiem pionowej linii.

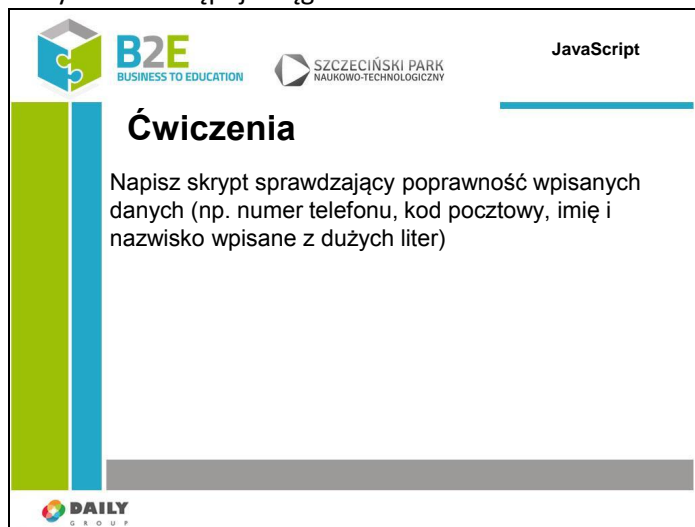
Grupowanie daje nam dodatkowo możliwość odwołania się do ciągu odnalezionego w tekście. Każda grupa wyodrębniona za pomocą nawiasu ma swój kolejny numer. Możemy się do takiej grupy odwołać za pomocą wyrażenia \numer. W przykładzie sprawdzamy czy w ciągu wartości oddzielonych średnikiem nie ma dwóch takich samych umieszczonych obok siebie

Inną dodatkową możliwością jest spojrzenie w przód – sprawdzenie czy w dalszej części wyrażenia występuje lub nie występuje określony ciąg.

Stosujemy do tego operatory `?=` i `?!=`

W powyższym przykładzie w tekście `0testabc1testxyz` w 1 przypadku zostanie zwrócony tekst `0test` (cyfra, 4 znaki a-z a po nich występuje ciąg `abc`) a w drugim przypadku `1test` – pierwszy fragment jest pominięty, ponieważ chcemy w wyniku otrzymać wyrażenie po którym nie następuje ciąg `abc`.

Slajd 36



B2E
BUSINESS TO EDUCATION

SZCZECIŃSKI PARK
NAUKOWO-TECHNOLOGICZNY

JavaScript



Ćwiczenia

Napisz skrypt sprawdzający poprawność wpisanych danych (np. numer telefonu, kod pocztowy, imię i nazwisko wpisane z dużych liter)

DAILY
GROUP




Slajd 37



JavaScript

Przykładowe rozwiązanie

```
<html><head>
<title> JavaScript</title>
<script type="text/javascript" src="l9cw5.js" >
function sprawdz() {
var regImie = new RegExp("[A-Z][a-z]+");
var regNrTel = new RegExp("\+\d{2}\(0?\d{2})\d{7}");
var imie = document.getElementById("imie");
var telefon = document.getElementById("telefon");
if (imie.value == regImie.match() && telefon.value == regNrTel.match()
)
    alert("OK");
}
</script>
</head>
```




Slajd 38

JavaScript

Przykładowe rozwiązanie

```
<body>
Proszę podać swoje dane: <br />
<form name="form_nr_1" action="form.php" method="get">
<table border="0">
<tr><td><b>Dane personalne:</b></td><td></td></tr>
<tr><td>imię:</td><td><input type="text" name="imie"></td></tr>
<tr><td>nazwisko:</td><td><input type="text" name="nazwisko"></td></tr>
<tr><td>nr telefonu:</td><td><input type="text" name="telefon"> format
+48 (022) 0000000</td></tr>
<tr><td>kod pocztowy:</td><td><input type="text" name="kod"> format 00-
000</td></tr>
<tr><td align="center" colspan="2"><input type="button" name="wyslij"
value=" Wylij! " onclick="check();"></td></tr>
</table>
</form></body></html>
```



Slajd 39



Ćwiczenia

Na slajdach zaprezentowano kilka podstawowych ćwiczeń wykorzystujących zaprezentowane możliwości języka. Pierwszym jest ćwiczenie zmieniające kolor przycisku (slajd 6) obrazujący wykorzystanie JS do zmiany wyglądu witryny www.

Drugim ćwiczeniem jest ćwiczenie wymagające napisania funkcji walidującej wartość wpisaną w pole formularza (slajd 13)

Opis założonych osiągnięć ucznia

Uczestnik po zakończeniu lekcji zna podstawowe sposoby wykorzystania języka JavaScript przy tworzeniu stron www. Potrafi za pomocą wyrażeń regularnych przeszukiwać treść strony.

Lekcja 10 Biblioteka JQuery

Cel lekcji

Celem lekcji jest wprowadzenie do zaawansowanego, ale bardzo prostego w użyciu frameworka JavaScript – jQuery. Jest to biblioteka funkcji napisana w języku JavaScript rozszerzająca i upraszczająca część rozwiązań zaprezentowanych w poprzedniej lekcji.

Sposoby osiągnięcia celów kształcenia

Czynności nauczyciela	Czynności ucznia
wprowadza nowe pojęcia związane z tematem zajęć (materiał ze slajdów) przygotowuje ćwiczenia z zaprezentowanego materiału	Wykonuje zadane przez nauczyciela ćwiczenia na komputerze
Prowadzi z uczniami dyskusję na temat wybranych przez nich rozwiązań ćwiczeń	Prezentuje rozwiązania ćwiczeń nauczycielowi i pozostałym uczniom, tłumaczy wybrane przez siebie rozwiązanie
Podsumowuje lekcję, ocenia pracę uczniów na zajęciach.	Dokonuje samooceny swojej pracy, omawia z nauczycielem rozwiązania ćwiczeń

Treść – slajdy z opisem

Slajd 1

W lekcji przedstawimy krótki opis frameworka jQuery, który jest jednym z najczęściej wykorzystywanym zbiorem funkcji w JavaScript. Zaprezentujemy podstawowe mechanizmy jQuery, jego składnię a także kilka praktycznych przykładów wykorzystania w witrynach WWW.

Slajd 2

jQuery to lekka i szybka biblioteka JavaScript. "Write less, do more"

Zawiera następujące funkcjonalności:


- manipulacje dokumentem html/ drzewem DOM
- manipulację CSS
- obsługę zdarzeń html
- efekty i animacje,
- AJAX,
- narzędzia

-system wtyczek pozwalający na rozszerzenie biblioteki


Zawiera proste do wykorzystania API, które działa w wielu przeglądarkach (obsługą różnic w interpretacji JavaScript zajęli się twórcy biblioteki)

jQuery jest wykorzystywane m.in. przez Wordpress (system do blogów, zarządzania treścią) oraz przez Wikipedię, Google, Microsoft itp.

Slajd 3



B2E
BUSINESS TO EDUCATION



SZCZECIŃSKI PARK
NAUKOWO-TECHNOLOGICZNY


JavaScript

Dodawanie jQuery

Aby wykorzystywać jQuery musimy załączyć do naszej strony plik z kodem biblioteki.

Możemy pobrać go ze strony www.jquery.com, lub wykorzystać mechanizm CDN – content delivery network.

```
<script src="jquery-1.9.1.min.js"></script>  
<script  
src="//ajax.googleapis.com/ajax/libs/jquery/1.9.1/jquery.  
min.js"></script>
```



DAILY
GROUP

Aby wykorzystywać jQuery musimy pobrać i załączyć do naszej strony plik z kodem biblioteki. Jest on dostępny na stronie www.jquery.com. Aktualna wersja to 1.9.1. Możemy pobrać plik w 2 wersjach zminimalizowanej lub normalnej. Wersja zminimalizowana ma usunięte wszystkie zbędne białe znaki. Jej celem jest maksymalne zmniejszenie rozmiaru pliku, co pozwala na szybsze załadowanie go do przeglądarki. Jeśli chcemy natomiast poznać budowę poszczególnych funkcji warto ściągnąć wersję normalną, bardziej czytelną dla człowieka.

Zamiast pobierać plik i podłączać go do naszej strony możemy wskazać odwołanie do skryptu wykorzystując mechanizm CDN – content delivery network. Są to specjalne serwisy przechowujące i udostępniające treści w internecie. Na slajdzie przedstawiłem przykład pobrania skryptu ze stron Google.

Zaletą podłączenia skryptu z CDN jest fakt, że w trakcie przeglądania internetu nasza przeglądarka już mogła pobrać jQuery (przy odwiedzeniu innej strony korzystającej z tej biblioteki) i sam plik został już umieszczony w pamięci podręcznej przeglądarki.



Slajd 4

W jQuery wyszukiwujemy element html i wykonujemy na nim akcje. Podstawowy zapis to `$(selector).action()`, gdzie znak `$` oznacza, że wykorzystujemy jQuery, `(selector)` – wyrażenie służące do wyszukiwania elementu, `action()` - akcja do wykonania

Przykłady:

`$(this).hide()` – ukrywa bieżący element

`$("p").hide()` – ukrywa wszystkie paragrafy

`$(".test").hide()` – ukrywa wszystkie elementy o klasie test

Istnieje możliwość, że zmienna `$` została już wykorzystana przez inną bibliotekę. Wtedy należy wykorzystać opcję jQuery – `noConflict()`. Pozwoli to na wykorzystywanie innych bibliotek js, a także na wykorzystanie zmiennej `$` przy dostępie do funkcji jQuery.

Standardowe wywołanie to `$.noConflict()`. Należy umieścić je przed każdym skrypcem wykorzystującym jQuery.

Slajd 5

Zdarzenie generowane po załadowaniu dokumentu. Wykorzystywane aby opóźnić wywołanie funkcji jQuery aż do momentu aż cała strona będzie dostępna.

Przykładami niepoprawnych wywołań byłoby np. ukrycie elementu, który nie został jeszcze stworzony, pobranie rozmiaru obrazka, który nie został jeszcze pobrany itp.

W skrypcie należy umieścić wywołanie:

```
$(document).ready(function() {
```

```
    //tutaj nasze instrukcje
```

```
}); lub wersję skróconą:
```

```
$(function() {
```

```
    // tutaj instrukcje });
```

// tutaj instrukcje });

Slajd 6

Selektory jQuery są jedną z najważniejszych części biblioteki. Pozwalają na manipulację elementami html.

Wykorzystując selektory można znaleźć elementy html bazując na ich identyfikatorach, klasach, typach, atrybutach a nawet wartościach atrybutów.

Selektor elementu zwraca elementy o zadanej nazwie znacznika html, np. selektor \$("p") zwróci wszystkie znaczniki p w dokumencie

Selektor identyfikatora zwraca elementy o danym identyfikatorze (atrybucie id) np.

\$("#test") zwróci element o id=test

Selektor klasy zwróci elementy o atrybucie class równym przekazanej wartości.

Selektor this zwraca bieżący element.

Selektor atrybutu [href] zwróci wszystkie elementy, które zawierają atrybut href

Slajd 7

jQuery jest dostosowany do obsługi zdarzeń. O zdarzeniach mówiliśmy na jednej z wcześniejszych lekcji. Jako przypomnienie zdarzenia pozwalają nam na reakcję na działania użytkowników na stronie. Tworząc odpowiednie metody obsługi zdarzeń możemy rozszerzać statyczne strony o interaktywne elementy.

Aby przypisać zdarzenie do elementu należy zastosować następującą składnię. Po selektorze elementu stawiamy . a po niej nazwę zdarzenia. Większość zdarzeń DOM ma swój odpowiednik w jQuery.

Najczęściej wykorzystywane zdarzenia:

\$(document).ready() – odpalane po załadowaniu dokumentu



click() – odpalane po kliknięciu na element
mouseenter() – odpalane po najechaniu kursorem na element
mouseleave() – odpalane po opuszczeniu elementu przez kursor
hover() – ma 2 argumenty, jest połączeniem mouseenter i mouseleave

Slajd 8

Slide 8 content: The slide features a header with logos for B2E (Business to Education), Szczeciński Park Naukowo-Technologiczny, and JavaScript. The main title is 'Efekty jQuery'. Below it, the text reads 'Podstawowe efekty w jQuery:' followed by a bulleted list: 'ukrywanie/pokazywanie', 'zanikanie/rozświetlanie', 'zwijanie/rozwijanie', and 'animacje'. The slide has a green and blue vertical bar on the left and a 'DAILY GROUP' logo at the bottom.

Slajd 9

Slide 9 content: The slide features the same header as slide 8. The main title is 'Efekty jQuery'. Below it, the text reads 'Wykorzystując jQuery można ukrywać i pokazywać elementy html.' followed by three code snippets: '\$(selektor).hide(predkosc, funkcja_zwrotna)', '\$(selektor).show(predkosc, funkcja_zwrotna)', and '\$(selektor).toggle(predkosc, funkcja_zwrotna)'. The slide has a green and blue vertical bar on the left and a 'DAILY GROUP' logo at the bottom.

Wykorzystując jQuery można ukrywać i pokazywać elementy html. Składnia poleceń jest następująca:

```
$(selektor).hide(predkosc, funkcja_zwrotna)
```

```
$(selektor).show(predkosc, funkcja_zwrotna)
```

prędkość to prędkość pokazywania/ukrywania elementu. Możliwe są następujące wartości: fast, slow i czas podany w milisekundach.

Opcjonalny parametr funkcja_zwrotna oznacza funkcję, która ma być uruchomiona po zakończeniu akcji. Zamiast podłączać 2 akcje – jedną do ukrywania elementu, drugą do pokazywania możemy wykorzystać akcję toggle. Jeśli element był ukryty to zostanie pokazany, jeśli jest widoczny to zostanie ukryty.

Slajd 10

JavaScript

Efekty - zanikanie

W jQuery można wykonać animację zanikania i pojawiania się elementu. Służą do tego akcje:

- fadeIn()
- fadeOut()
- fadeToggle()
- fadeTo()

W jQuery można wywołać efekt zanikania i pojawiania się elementu. Wykorzystuje się do tego następujące metody:

- fadeIn() – rozjaśnia ukryty element.
- fadeOut() – element zanika z ekranu
- fadeToggle() – przełącza między metodami fadeIn i fadeOut
- fadeTo() – element pojawia się z określoną przezroczystością (wartość między 0 a 1)

Slajd 11

JavaScript

Efekty – rozwijanie i zwijanie

Efekt rozwijania i zwijania elementów można uzyskać za pomocą następujących akcji:

- slideUp()
- slideDown()
- slideToggle()

Efekt rozwijania i zwijania elementów można uzyskać za pomocą następujących akcji:

- slideUp()
- slideDown()
- slideToggle()

Typy akcji są identyczne jak poprzednie.

Slajd 12

JavaScript

Animacje

Metoda animate() pozwala na utworzenie własnych animacji

```
$(selektor).animate(
  {parametry},
  predkosc,
  funkcja_zwrotna);
```

```
$("button").click(function() {
  $("div").animate({left:250px});
```

Metoda animate pozwala na utworzenie własnych animacji. Wymagany argument "parametry" definiuje właściwości css, które będą animowane. Powyższy przykład po naciśnięciu przycisku przesunie element w lewo aż jego właściwość left osiągnie 250 pikseli.

Domyślnie elementy HTML mają właściwość static – nie można ich przesuwac. Przed zdefiniowaniem animacji trzeba ustawić właściwość position elementu na relative, fixed lub absolute.

Można w parametrach przekazać więcej niż 1 właściwość. Możemy stworzyć animację, która przesuwa element i zwiększy jego rozmiar. Jak powinna wyglądać taka funkcja?

Za pomocą metody animate można zmieniać większość właściwości stylu css – trzeba tylko pamiętać o zmianie trybu pisania na camel case. zamiast pisać padding-left napiszemy paddingLeft itd. Podstawowa biblioteka jQuery nie obsługuje animacji atrybutu color. Aby animować zmiany kolorów trzeba pobrać odpowiednią wtyczkę z witryny jquery.

jQuery automatycznie tworzy kolejkę dla animacji. Możemy zdefiniować kilka animacji a jQuery automatycznie wyświetli je jedna po drugiej.

Animacje można zatrzymać wykorzystując metodę stop z opcjonalnymi argumentami stopAll (powoduje zatrzymanie całej kolejki animacji) i goToEnd – przechodzi na koniec animacji.

Oba argumenty mają domyślną wartość false, co oznacza że zatrzymana zostanie tylko bieżąca animacja.

Slajd 13

Funkcje zwrotne

Wyrażenia JavaScript są przetwarzane jedno po drugim.

Jednak w przypadku efektów, kolejna linia może zostać przetworzona zanim efekt się zakończy.

Funkcje zwrotne to funkcje, które zostaną wykonane, gdy działanie efektu się zakończy.

Wyrażenia JavaScript są przetwarzane jedno po drugim.

Jednak w przypadku efektów, kolejna linia może zostać przetworzona zanim efekt się zakończy.

Funkcje zwrotne to funkcje, które zostaną wykonane, gdy działanie efektu się zakończy.

Slajd 14

Łączenie akcji

Łączenie akcji pozwala na wywołanie wielu metod jQuery na danym elemencie za pomocą jednego wyrażenia. W ten sposób przeglądarka nie musi wyszukiwać danego elementu po raz kolejny. Aby połączyć akcje należy ją po prostu dodać po wywołaniu poprzedniej akcji.

```
$( "#p1" ).css( "color", "red" )
    .slideUp( 2000 ).slideDown( 2000 );
```

Łączenie akcji pozwala na wywołanie wielu metod jQuery na danym elemencie za pomocą jednego wyrażenia. W ten sposób przeglądarka nie musi wyszukiwać danego elementu po raz kolejny. Aby połączyć akcje należy ją po prostu dodać po wywołaniu poprzedniej akcji. Przykład obrazuje działanie na obiekcie o id #p1 – zmieniamy jego kolor na czerwony, po czym zwiijamy obiekt a następnie rozwijamy. Operacje zwiijania i rozwijania trwają po 2 sekundy

Slajd 15

Manipulacja DOM

jQuery zawiera metody do manipulacji elementami html i ich atrybutami. Wykorzystuje do tego metody działające na modelu dokumentu (DOM)

DOM – interfejs, który pozwala programom i skryptom na dynamiczny dostęp i modyfikację treści, struktury i stylu dokumentu

jQuery zawiera metody do manipulacji elementami html i ich atrybutami.

Wykorzystuje do tego metody działające na modelu dokumentu (DOM)

Jak wspominaliśmy wcześniej DOM to interfejs, który pozwala programom i

skryptom na dynamiczny dostęp i modyfikację treści, struktury i stylu dokumentu.

Slajd 16

Pobieranie treści text() html()

- text()

```
$("#btn1").click(function(){  
  alert("Text: " + $("#test").text());  
});
```
- html()

```
$("#btn2").click(function(){  
  alert("HTML: " + $("#test").html());  
});
```


Podstawowymi metodami do pobierania treści dokumentu są metody:

text() – pobiera lub ustawia wartość tekstową wybranego elementu.


html() – pobiera lub ustawia treść elementu (razem ze znacznikami html)



Slajd 17



B2E
BUSINESS TO EDUCATION



SZCZECIŃSKI PARK
NAUKOWO-TECHNOLOGICZNY


JavaScript

Metody val() i attr()

- val()


```
$("#btn1").click(function(){
    alert("Value: " + $("#test").val());
});
```
- attr()


```
$("#button").click(function(){
    alert($("#link").attr("href"));
});
```




val() – pobiera lub ustawia wartość pola formularza. W przypadku ustawiania pól typu select i checkbox do pobrania wartości można użyć selektorów :selected i :checked. Jeśli select nie ma ustawionej wartości wynikiem działania funkcji val będzie null – jeśli wartość jest ustawiona to w wyniku otrzymamy tablicę z zaznaczonymi elementami. Dzieje się tak dlatego, że w znaczniku select możemy wybrać więcej niż jedną wartość (ustawiając atrybut multiselect).


Metoda attr() pobiera wartość danego atrybutu

Wszystkich wymienionych teraz funkcji można do ustawienia wartości danego elementu. Wartość te należy umieścić w nawiasie jako parametr metody. Zamiast wartości możemy podać funkcję zwrrotną, która jako argumenty przyjmuje indeks bieżącego elementu w zaznaczeniu, oraz wartość oryginalną elementu. Jako wartość wynikową ustawiamy tekst, który ma być nową wartością elementu.

Slajd 18



B2E
BUSINESS TO EDUCATION



SZCZECIŃSKI PARK
NAUKOWO-TECHNOLOGICZNY


JavaScript

Dodawanie elementów

- append()


```
$("#p").append("dodano z jQuery.");
```
- prepend()


```
$(p).prepend("dodano z jQuery.");
```
- after()
- before()



Za pomocą jQuery można bardzo łatwo dodawać elementy i treść do strony. Służą do

tego następujące metody:

-append() – dodaje treść na końcu danego elementu.

Załóżmy, że mamy document zawierający jakiś paragraf o treści Witaj świecie!.

Wywołanie metody \$("p").append("dodano z jQuery."); spowoduje, że paragraf ten będzie miał teraz treść: Witaj świecie! dodano z jQuery.;

-prepend() – dodaje treść na początku danego elementu. W powyższym przykładzie wynikiem będzie tekst: "dodano z jQuery.Witaj świecie!"

Metody after i before wstawiają treść odpowiednio przed i po elemencie – nie tak jak wcześniej na początku lub końcu wartości elementu.

Slajd 19

JavaScript

Usuwanie elementów

Do usuwania elementów służą metody:

- remove()
 - \$("div").remove()
 - \$("p").remove(".italic");
- empty()
 - \$("div").empty();

Do usuwania elementów

służą metody:

-remove()

-empty()

Metoda remove usuwa wszystkie węzły potomne elementu oraz sam element, jako parametr możemy podać selektor elementów do usunięcia. W przykładzie usuwamy wszystkie elementy p, które mają klasę italic.

Metoda empty usuwa wszystkie węzły potomne danego elementu.

Slajd 20

JavaScript

Manipulacja CSS

Do zmiany stylu css elementów służą następujące metody:

- addClass()
- removeClass()
- toggleClass()
- css()

Do zmiany stylu css elementów służą następujące metody:

-addClass() – metoda pozwala na dodanie klasy do wybranego elementu na stronie. Jako parametr podajemy nazwę klasy css zdefiniowaną w pliku stylu. Możemy dodać od razu kilka klas

-removeClass() – metoda usuwa podaną klasę z elementu.

-toggleClass() – metoda dodaje lub usuwa klasę z elementu

-css() – ustawia lub zwraca jeden lub wiele właściwości stylu dla wybranego elementu; aby pobrać wartość elementu należy wywołać metodę z nazwą właściwości, aby ustawić – oprócz nazwy podajemy nową wartość. W przypadku pobierania wartości zostanie zwrócona wartość pierwszego dopasowanego obiektu do selektora. Ustawienie wartości

Slajd 21

jQuery i AJAX

- co to jest AJAX?
Asynchronous JavaScript and XML

w skrócie: służy do pobierania danych w tle i wyświetlania ich na stronie bez konieczności przeładowania strony

Co to jest AJAX?
Asynchronous JavaScript and XML.
Służy do pobierania danych w tle i wyświetlania ich na stronie bez konieczności przeładowania strony. Pisanie wywołań AJAX jest utrudnione, różne przeglądarki inaczej implementują AJAX. Oznacza to konieczność napisania dodatkowego kodu dla różnych przeglądarek. W jQuery funkcjonalność ta już została zaimplementowana i można jej używać używając pojedynczej linii kodu.

Slajd 22

metoda load()

Metoda load() pobiera dane z serwera i umieszcza je w wybranym elemencie

```
$(selektor).load(url, dane, funkcja_zwrotna)
$("#div1").load("dane.txt");
```

Metoda load() pobiera dane z serwera i umieszcza je w wybranym elemencie.

Polecenie może mieć 3 parametry:

- obowiązkowy URL, który wskazuje adres serwera (pliku lub usługi),
- opcjonalne dane: zawierający informacje które chcemy wysłać razem z żądaniem (np. identyfikatory, nagłówki autoryzacji itp.)
- opcjonalne funkcje_zwrotną, która zostanie wywołana po odebraniu danych.

Funkcja zwrotna ma 3 parametry: responseTxt – tekst odpowiedzi, statusTXT – zawiera status żądania, xhr – zawiera obiekt XMLHttpRequest

Założmy że mamy na dysku plik dane.txt zawierający dane, które chcemy wyświetlić w elemencie div1. Użyjemy do tego polecenia \$("#div1").load("dane.txt");

Slajd 23

The slide features a header with logos for B2E (Business to Education), Szczeciński Park Naukowo-Technologiczny, and JavaScript. The main title is 'jQuery get() i post()'. Below the title, it lists the methods 'Metody get() i post()' and provides two code snippets: '\$.get(URL, funkcja_zwrotna);' and '\$.post(URL, dane, funkcja_zwrotna);'. The slide also includes a 'DAILY GROUP' logo at the bottom left.

Metody get i post wysyłają żądania do serwera za pomocą żądań HTTP GET i HTTP POST.

GET – pobiera dane z określonego adresu

POST – wysyła dane na określony adres

Funkcja get() przyjmuje 2 parametry:

-obowiązkowy URL – adres zasobu, który chcemy pobrać

-i opcjonalny funkcja_zwrotna (callback) – funkcja wywołana po odebraniu danych

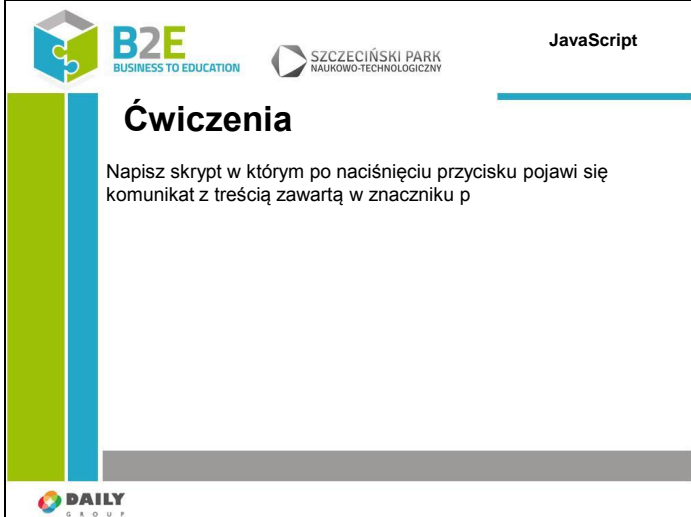
Funkcja post() przyjmuje 3 argumenty:

- obowiązkowy URL – adres na który wysyłamy żądanie.

-opcjonalny dane - dane które chcemy wysłać. Dane muszą być w formacie JSON

-opcjonalny funkcja_zwrotna – funkcja, która zostanie uruchomiona po wysłaniu danych


Slajd 24

The slide features a header with logos for B2E (Business to Education), Szczeciński Park Naukowo-Technologiczny, and JavaScript. The main title is 'Ćwiczenia'. Below the title, it contains the text: 'Napisz skrypt w którym po naciśnięciu przycisku pojawi się komunikat z treścią zawartą w znaczniku p'. The slide also includes a 'DAILY GROUP' logo at the bottom left.


Napisz skrypt w którym po naciśnięciu przycisku pojawi się komunikat z treścią zawartą w znaczniku p.



Slajd 25



B2E
BUSINESS TO EDUCATION




SZCZECIŃSKI PARK
NAUKOWO-TECHNOLOGICZNY

JavaScript


Przykładowe rozwiązanie

```
$(document).ready(function() {  
  $("#btn1").click(function() {  
    alert($("#p").text());  
  });  
});  
(...)  
<body><p>To jest nasza strona</p>  
<button id="btn1"> Wyświetl </button>  
</body>
```




DAILY
GROUP

Slajd 26



B2E
BUSINESS TO EDUCATION




SZCZECIŃSKI PARK
NAUKOWO-TECHNOLOGICZNY

JavaScript

Ćwiczenia

Napisz skrypt w którym po naciśnięciu przycisku do paragrafu został dodany tekst "dodany tekst"




DAILY
GROUP


Napisz skrypt, w którym po naciśnięciu przycisku do paragrafu został dodany tekst "dodany tekst".



Slajd 27



B2E
BUSINESS TO EDUCATION




SZCZECIŃSKI PARK
NAUKOWO-TECHNOLOGICZNY


JavaScript

Przykładowe rozwiązanie


```
$(document).ready(function() {  
  $("#btn1").click(function() {  
    $("p").append("dodany tekst");  
  });  
});  
(...)  
<body><p>To jest nasza strona</p>  
<button id="btn1"> Dodaj tekst</button>  
</body>
```



Slajd 28



B2E
BUSINESS TO EDUCATION




SZCZECIŃSKI PARK
NAUKOWO-TECHNOLOGICZNY

JavaScript

Ćwiczenia



Napisz skrypt w którym po naciśnięciu przycisku kolor tekstu w paragrafie zmieni się na niebieski.



Napisz skrypt w którym po naciśnięciu przycisku kolor tekstu w paragrafie zmieni się na niebieski.




Slajd 29





JavaScript

Przykładowe rozwiązanie

```
$(document).ready(function() {  
  $("#btn1").click(function() {  
    $("p").css(color: blue;);  
  });  
});  
(...)  
<body><p>To jest nasza strona</p>  
<button id="btn1"> Wyświetl </button>  
</body>
```




Slajd 30



JavaScript

Ćwiczenia



Napisz skrypt w którym po naciśnięciu przycisku tekst paragrafu zostanie pogrubiony



Napisz skrypt w którym po naciśnięciu przycisku tekst paragrafu zostanie pogrubiony.




Slajd 31



JavaScript



Przykładowe rozwiązanie

```

$(document).ready(function() {
  $("#btn1").click(function() {
    $("#txt").html("<b>"+$("#txt").text()+"</b>");
  });
});
(...)
<p id="txt">To jest przykładowy tekst</p>
<button id="btn1">Ustaw pogrubiony</button>
    
```




Slajd 32



JavaScript

Podsumowanie

W trakcie tej lekcji poznaliśmy podstawowe informacje o frameworku jQuery. Wiemy, że pozwala on m.in. na:

- animowanie elementów na stronie (pokazywanie, ukrywanie, przesuwanie itp.)
- wyszukiwanie i modyfikację treści elementów
- dodawanie i usuwanie elementów
- modyfikacje stylu css strony
- komunikację z innymi usługami z wykorzystaniem AJAX



W trakcie tej lekcji poznaliśmy podstawowe informacje o frameworku jQuery. Wiemy, że pozwala on m.in. na:

- animowanie elementów na stronie (pokazywanie, ukrywanie, przesuwanie itp.)
- wyszukiwanie i modyfikację treści elementów
- dodawanie i usuwanie elementów
- modyfikacje stylu css strony
- komunikację z innymi usługami z wykorzystaniem AJAX

Ćwiczenia

Napisz skrypt dodający efekt "zebry" w tabeli (wiersze parzyste i nieparzyste mają inne kolory). Dodatkowo niech wiersz nagłówkowy zawiera test pogrubiony.

Utwórz stronę zawierającą prosty formularz (zawierający jedno pole o identyfikatorze "name") i element div. Po wprowadzeniu znaku do pola dodaj wpisany znak do utworzonej warstwy. Wykorzystaj zdarzenie keyup.

Fragment przykładowego rozwiązania (część jQuery):



```
<script type="text/javascript">
$(document).ready(function() {
$("#name").bind('keyup', function() {
    $("#div").append("").append($(this).val());
});
});
</script>
```

Opis założonych osiągnięć ucznia

Uczestnik po zakończeniu lekcji potrafi dodać do strony bibliotekę jQuery, a także wykorzystać ją podczas tworzenia witryny www. Potrafi dynamicznie zmieniać wygląd strony a także dodać podstawowe efekty na stronie.

Lekcja 11 Projekt Interdyscyplinarny

Slajd 1

JavaScript

Projekt interdyscyplinarny
Baza teleadresowa koleżanek
i kolegów ze szkoły

Człowiek - najlepsza inwestycja

KAPITAŁ LUDZKI
NARODOWA STRATEGIA SPÓJNOŚCI

UNIA EUROPEJSKA
EUROPEJSKI
FUNDUSZ SPOŁECZNY

Projekt współfinansowany przez Unię Europejską w ramach Europejskiego Funduszu Społecznego

DAILY GROUP

W ramach projektu dodamy do naszej aplikacji funkcje JavaScript sprawdzające poprawność wpisanych danych.

Slajd 2

JavaScript

Założenia

- Przy usuwaniu rekordów z bazy system wyświetli okno dialogowe z potwierdzeniem usunięcia rekordu
- Do każdej klasy będzie dodany wychowawca
- Imię i nazwisko wychowawcy oraz uczniów będzie wprowadzone z dużej litery
- Pole telefon będzie nieobowiązkowe, natomiast jeśli podamy numer to wymagane będzie wprowadzenie 9 cyfr
- Adres e-mail będzie poprawnym adresem

DAILY GROUP



Slajd 3

Potwierdzenie usuwania

Funkcja confirm()
Funkcja będzie wywoływana w następujących miejscach:

- Usuwanie klas
- Usuwanie uczniów

Do uzyskania potwierdzenia wykorzystamy standardową funkcję JavaScript – confirm(). Zwraca ona wartość true jeśli użytkownik wybierze przycisk "OK" i false jeśli zamknie okno, lub wybierze przycisk "Anuluj".

Wywołanie tej funkcji dodamy w widokach listy klas i listy uczniów.

Przykładową realizacją będzie dodanie klasy do każdego odnośnika, uruchamiającego akcję usuń (w pliku GUI.php) i dodanie skryptu wyświetlającego okno z pytaniem.


```
$(document).ready(function() {  
  $(".deleteBtn").click(function(e) {  
    if (!confirm("Czy na pewno chcesz usunąć ten rekord?")) {  
      e.preventDefault();  
    }  
  });  
});
```





Klasę deleteBtn musimy dodać w metodach:

- displayClasses
- displayClass(id)

Slajd 4



B2E
BUSINESS TO EDUCATION



SZCZECIŃSKI PARK
NAUKOWO-TECHNOLOGICZNY


JavaScript

Walidacja tworzenia klasy

- Nazwa klasy i wychowawca jest wymagana
- Imię i nazwisko wychowawcy wprowadzone z dużych liter.

Przy wysyłaniu formularza tworzenia klasy uruchomimy funkcję sprawdzającą poprawność danych.

Klasa zostanie utworzona dopiero w momencie gdy formularz będzie poprawnie wypełniony.



Przy wysyłaniu formularza tworzenia klasy sprawdzimy czy oba pola są wprowadzone oraz czy w polu wychowawca wprowadzono imię i nazwisko z dużych liter (czy 1 znak to duża litera oraz czy dalej w ciągu jest spacja a po niej kolejna duża litera.

Wykorzystamy do tego wyrażenie regularne

Przykładowy skrypt:

```
var classForm = document.forms["classForm"];
var regWychowawca = new RegExp("[A-Z][a-z]*\s[A-Z][a-z]*");
if (classForm.className == "" || classForm.educator == "" || !classForm.wychowawca
== regWychowawca.match() ) {
    alert("Nieprawidłowo wprowadzone dane");
    e.preventDefault();
}
```

}

Slajd 5

Slide 5 content: B2E BUSINESS TO EDUCATION logo, SZCZECIŃSKI PARK NAUKOWO-TECHNOLOGICZNY logo, JavaScript text, title "Walidacja tworzenia uczniów", requirements list, and DAILY GROUP logo.

Walidacja tworzenia uczniów

Wymagania:

- Imię i nazwisko z dużej litery
- E-mail jeśli jest podany to występuje w nim znak @ oraz .
- Telefon jeśli jest podany składa się z 9 cyfr

Slajd 6

Slide 6 content: B2E BUSINESS TO EDUCATION logo, SZCZECIŃSKI PARK NAUKOWO-TECHNOLOGICZNY logo, JavaScript text, title "Walidacja tworzenia uczniów", requirement list, and DAILY GROUP logo.

Walidacja tworzenia uczniów

- Imię i nazwisko z dużej litery

```
var nameReg = new RegExp("[A-Z][a-z]+");
if(studentForm.imie != nameReg.match() ||
studentForm.nazwisko != nameReg.match()) {
alert("Imię i nazwisko musi być wpisane z dużej litery");
e.preventDefault();
}
```

Pobranie formularza:

```
var studentForm = document.forms["studentForm"];
```

Wyrażenie regularne sprawdzające poprawność wprowadzonego imienia i nazwiska

```
var nameReg = new RegExp("[A-Z][a-z]+");
```




Jeśli dane nie są wprowadzone poprawnie przerywamy wysyłanie skryptu. Do bazy danych nie zostanie dodany nowy student.

```


$( "#studentForm" ).submit(function(e) {
    var result = true;
    console.log(document.forms);
    var sf = document.forms["studentForm"];
    if (sf.imie == "" || sf.nazwisko == "") {
        alert("Imię i nazwisko muszą być wypełnione");
        result = false;
    }
    var nameReg = new RegExp("[A-Z][a-z]+");
    if (sf.imie != nameReg.match() || sf.nazwisko != nameReg.match())
    {
        alert("Imię i nazwisko musi być wprowadzone z dużych liter");
        result = false;
    }
    if (!result) {
        e.preventDefault();
    }
});

```

Slajd 7



B2E
BUSINESS TO EDUCATION



SZCZECIŃSKI PARK
NAUKOWO-TECHNOLOGICZNY


JavaScript

Walidacja tworzenia uczniów

Wprowadzenie adresu e-mail

Wyrażenie regularne:

```
var emailReg = new RegExp("[A-Za-z0-9]+@[A-Za-z0-9]+.[A-Za-z]{2,}");
```

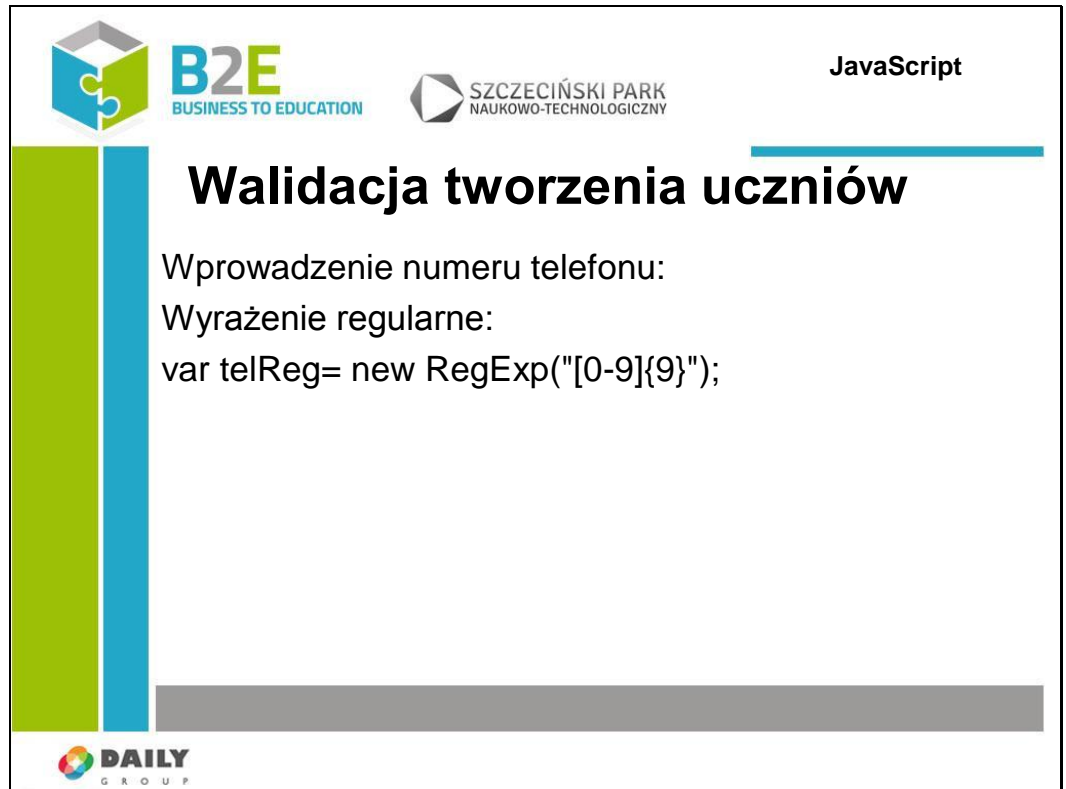


Dodajemy sprawdzenie adresu email – jeśli jest wprowadzony to musi spełniać następujące warunki:

- Zaczyna się dowolnym ciągiem znaków, po którym występuje znak @,
- Następnie występuje kolejny dowolny ciąg znaków a po nim znak .
- Po ostatniej kropce występują 2 lub więcej znaków (domeny głównego poziomu)

```
var sf = document.forms["studentForm"];  
var emailReg = new RegExp("[A-Za-z0-9]+@[A-Za-z0-9]+.[A-Za-z]{2,}");  
If (sf.email != "" && sf.email != emailReg.match()) {  
    alert("Nieprawidłowy adres email");  
    e.preventDefault();  
}
```

Slajd 8



The slide features a header with logos for B2E (Business to Education) and Szczeciński Park Naukowo-Technologiczny, and the text 'JavaScript'. The main title is 'Walidacja tworzenia uczniów'. Below the title, it says 'Wprowadzenie numeru telefonu: Wyrażenie regularne: var telReg= new RegExp("[0-9]{9}");'. The slide also includes logos for 'DAILY GROUP' and 'Szczyński Park Naukowo-Technologiczny'.

Dodajemy sprawdzenie numeru telefonu – jeśli jest wprowadzony to musi składać się z 9 cyfr.

```
var sf = document.forms["studentForm"];  
var telReg = new RegExp("[0-9]{9}");  
If (sf.telefon != "" && telefon != telReg.match()) {  
    alert("Numer telefonu powinien mieć 9 cyfr");  
    e.preventDefault();  
}
```



Slajd 9

JavaScript

Podsumowanie

- Dodawanie klas możliwe jest tylko gdy wprowadzimy poprawne dane wychowawcy
- Dodanie uczniów możliwe jest gdy wprowadzimy poprawne imię i nazwisko.
- Jeśli podamy adres email to musi on zawierać znak @ i kropkę.
- Numer telefonu składa się z 9 cyfr.

KAPITAŁ LUDZKI
NARODOWA STRATEGIA SPÓJNOŚCI

SZCZECIŃSKI PARK
NAUKOWO-TECHNOLOGICZNY

UNIA EUROPEJSKA
EUROPEJSKI
FUNDUSZ SPOŁECZNY

Projekt współfinansowany przez Unię Europejską w ramach Europejskiego Funduszu Społecznego

DAILY
GROUP

Podsumowanie:

- Dodawanie klas możliwe jest tylko gdy wprowadzimy poprawne dane wychowawcy
- Dodanie uczniów możliwe jest gdy wprowadzimy poprawne imię i nazwisko.
- Jeśli podamy adres email to musi on zawierać znak @ i kropkę.
- Numer telefonu składa się z 9 cyfr.

Cel lekcji

Celem lekcji jest dodanie do zaprojektowanej aplikacji sprawdzenia wprowadzanych danych. W tej lekcji uczniowie będą mogli wykazać się umiejętnościami nabytymi w czasie poprzednich zajęć.

Treść – slajdy z opisem

Opis założonych osiągnięć ucznia

Po tej lekcji uczniowie uzupełnią utworzoną przez siebie aplikację o dodatkową walidację wprowadzanych danych. Dane dostępne w książce teleadresowej kolegów i koleżanek z klasy i ze szkoły będą pozbawione błędów.

Opis założonych osiągnięć ucznia

Po tej lekcji uczniowie uzupełnią utworzoną przez siebie aplikację o dodatkową walidację wprowadzanych danych. Dane dostępne w książce teleadresowej kolegów i koleżanek z klasy i ze szkoły będą pozbawione błędów.

Bibliografia

Bougon M.K.E. , Weick D., Binkhorst D., Cognition in Organizations: An Analysis of the Utrecht Jazz Orchestra, Administrative Science Quarterly 22, 1997 [w:] baza cyfrowa ERIC.

Craik F.I.M., Lockhart R.S., Levels of processing: A Framework for Memory Research, Journal of Verbal Learning and Verbal Behaviour, 11, 1972, s. 671-684.

Czerepaniak – Walczak M., Perzycka E., (ed.) Media and Trust – Theoretical and Practical Contexts, Wydawnictwo Uniwersytetu Szczecińskiego, Szczecin, 2014.

Perzycka E., Struktura i dynamika kompetencji informacyjnych nauczyciela w społeczeństwie sieciowym, Wyd. Uniwersytet Szczeciński, Szczecin 2008.

Pety G., Nowoczesne nauczanie. Praktyczne wskazówki i techniki dla nauczycieli, wykładowców i szkoleniowców, Wydawnictwo GWP, Sopot 2010.

Sysło M., Lekcja informatyki, edunews.pl z dnia 06.02.2010.

Thinking in Java (4th Edition) Tłumaczenie: Przemysław Szeremiota na podstawie tłumaczenia Adriana Nowaka, Szymona Kobalczyka, Łukasza Fryza i Piotra Rajcy 2006.

<http://syslo.nq.pl/Edukacja/Dokumenty/Standardy-przygotowania-nauczycieli>