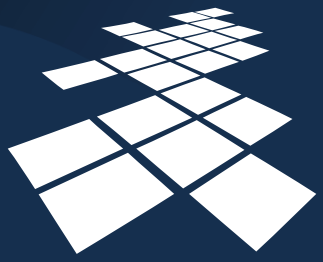
Aplikacje WWW

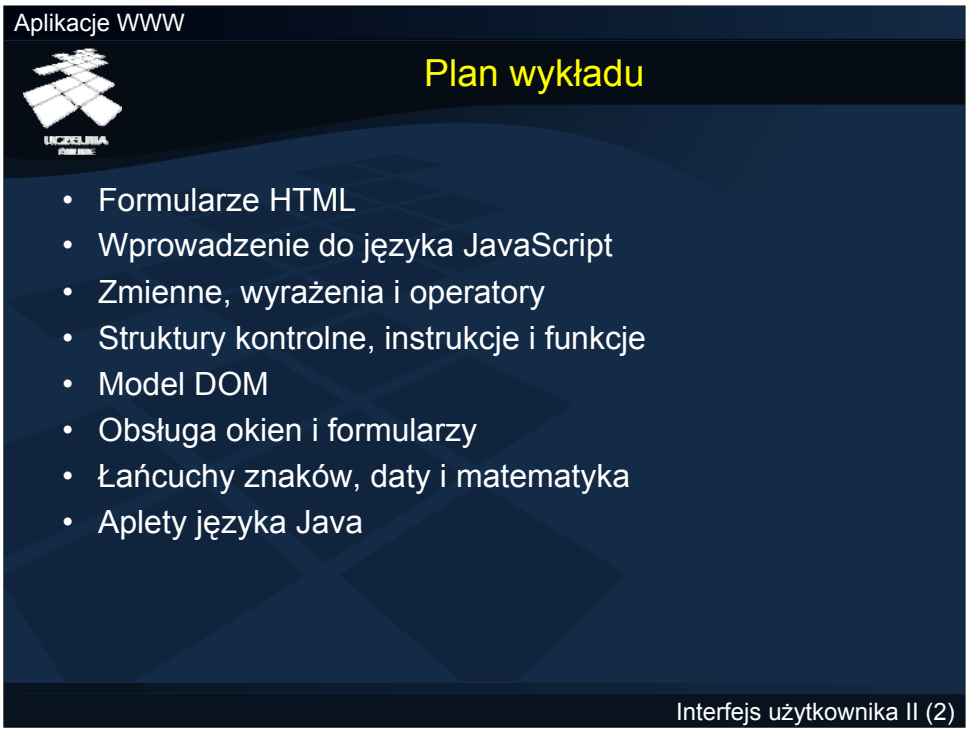
Interfejs użytkownika II

wykład prowadzi
Mikołaj Morzy

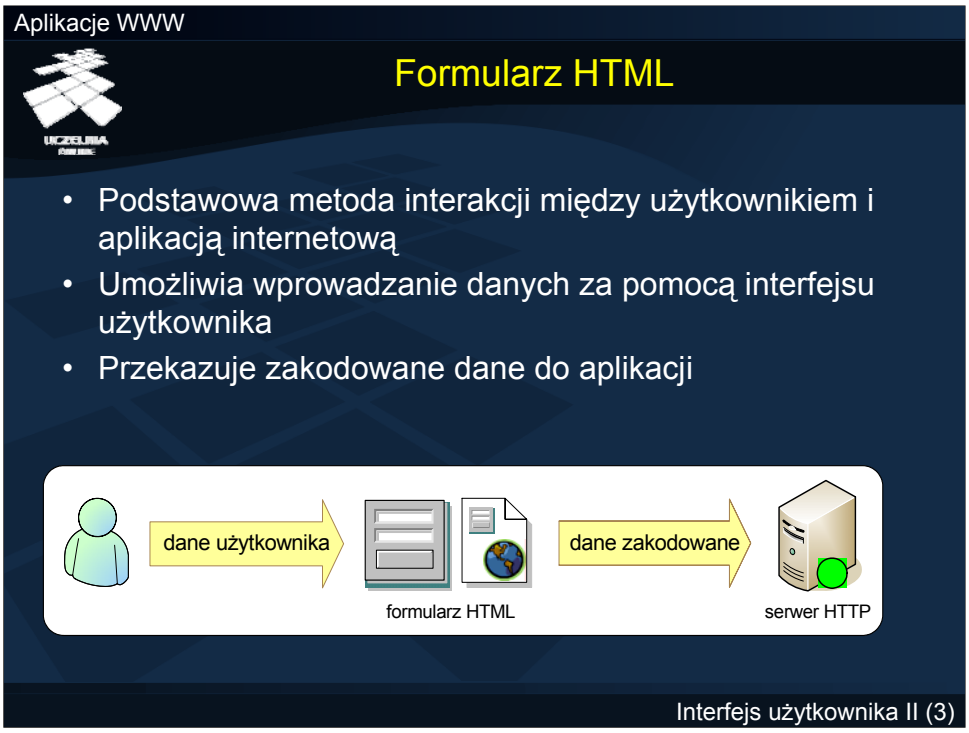


UCZELNIA
ONLINE

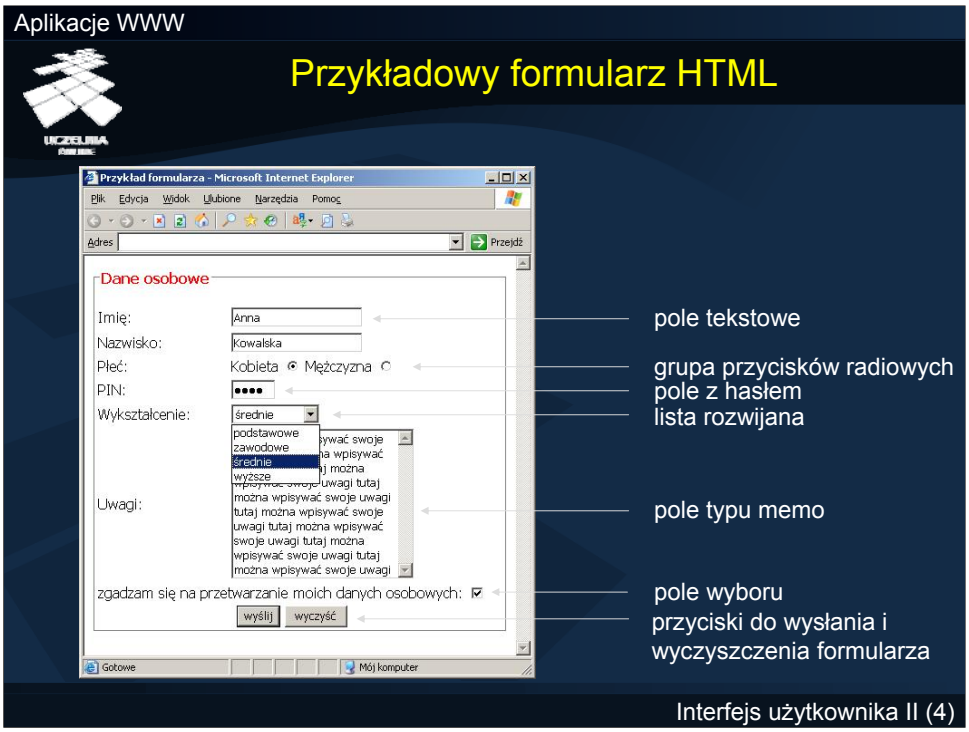
Interfejs użytkownika



Celem wykładu jest przedstawienie zaawansowanych technologii tworzenia interfejsu użytkownika na przykładzie formularzy HTML, języka skryptowego JavaScript oraz apletów języka Java. Wykład rozpoczyna prezentacja znaczników języka HTML służących do konstruowania formularzy. Następnie zaprezentowany zostanie popularny język skryptowy JavaScript, a w szczególności zaprezentowane zostaną podstawy języka (deklaracje zmiennych, wyrażenia, operatory języka, struktury kontrolne i instrukcje, w tym pętle i instrukcje warunkowe) oraz model DOM (ang. Document Object Model) definiujący interakcję między skryptem JavaScript i obiektami zawartymi w dokumencie HTML. Przedstawione zostaną również metody walidacji danych wprowadzanych przez użytkownika do formularzy HTML oraz metody manipulowania zawartością okna i dokumentu. Prezentację języka JavaScript zamknie omówienie metod pracy z łańcuchami znaków i datami oraz biblioteką oferującą bogaty zestaw funkcji matematycznych. Wykład zakończy prezentacja możliwości tworzenia interfejsu użytkownika za pomocą apletów języka Java.



Formularz HTML jest fragmentem dokumentu HTML zawierającym kontrolki umożliwiające wprowadzanie danych. Formularz HTML jest de facto jedynym mechanizmem umożliwiającym użytkownikowi przekazanie danych do aplikacji internetowej. Dane przekazywane z formularza mogą mieć różne zastosowanie, mogą to być np. kryteria wyszukiwania przekazane do wyszukiwarki, dane osobowe rejestrowane w formularzu zgłoszeniowym, czy numer konta i hasło podawane podczas logowania do banku internetowego. Kontrolki umieszczane w formularzach HTML obejmują: pola tekstowe, pola haseł, pola wyboru, pola radiowe, pola typu memo, listy rozwijane i przyciski. Dodatkowo, formularz może zawierać pola ukryte. Dzięki bogatemu zestawowi kontrolek użytkownik wprowadza dane w sposób przypominający pracę z tradycyjnymi aplikacjami. Formularz przesyła dane wprowadzone przez użytkownika do aplikacji działającej po stronie serwera HTTP. W zależności od wybranej metody przesłania formularza dane mogą być jeszcze dodatkowo zakodowane za pomocą kodowania URL (ang. URL-encoding).



Cała zawartość formularza jest ujęta w znaczniki <form>...</form>. Najczęściej do konstruowania elementów formularza wykorzystujemy znacznik <input>. W zależności od wartości atrybutu type możemy utworzyć następujące elementy formularza:

• type="text": pole tekstowe

• type="radio": element grupy przycisków radiowych, do jednej grupy trafiają wszystkie przyciski o tej samej wartości atrybutu name

• type="password": pole z hasłem, znaki wpisywane do pola nie są widoczne

• type="checkbox": pole wyboru

• type="file": pole służące do załadowania zawartości pliku

• type="submit": przycisk służący do wysyłania formularza

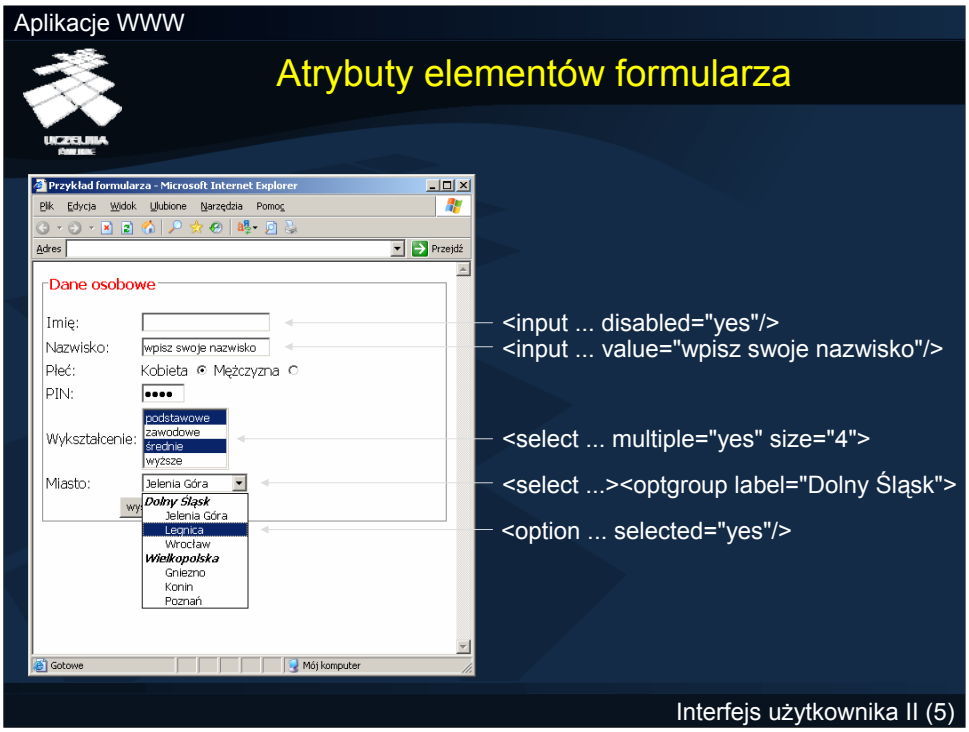
• type="reset": przycisk służący do czyszczenia zawartości formularza

Oprócz znacznika <input> elementy formularza można również utworzyć za pomocą następujących znaczników:

• <textarea rows="n" cols="m">: tworzy pole typu memo o podanej liczbie wierszy i kolumn

• <select>: tworzy listę rozwijaną, kolejne pozycje na liście są tworzone za pomocą znaczników <option>

• <hidden>: tworzy element ukryty, który nie jest wyświetlany w formularzu



Wygląd i zachowanie formularza można dowolnie modyfikować za pomocą atrybutów elementów formularza. Do najczęściej używanych atrybutów należą:

• name: nazwa elementu formularza, wymagany dla wszystkich elementów poza przyciskami typu submit i reset

• value: dla pól tekstowych i pól z hasłami zawiera domyślną wartość pola, dla przycisków zawiera etykietę przycisku, dla pól wyboru i przycisków radiowych zawiera wartość elementu przesyłaną do aplikacji

• size: rozmiar elementu

• readonly (yes|no): flaga oznaczająca, czy dany element może być modyfikowany

• checked (yes|no): flaga oznaczająca, czy dany element jest początkowo wyświetlony jako zaznaczony (dotyczy przycisków radiowych i pól wyboru)

Wysłanie formularza

- Atrybut **`action`** – identyfikator URI agenta przetwarzającego formularz
- Atrybut **`method`** - metoda wysyłania danych z formularza
  - **`GET`**: dane kodowane w adresie URL
  - **`POST`**: dane przesyłane w ciele komunikatu

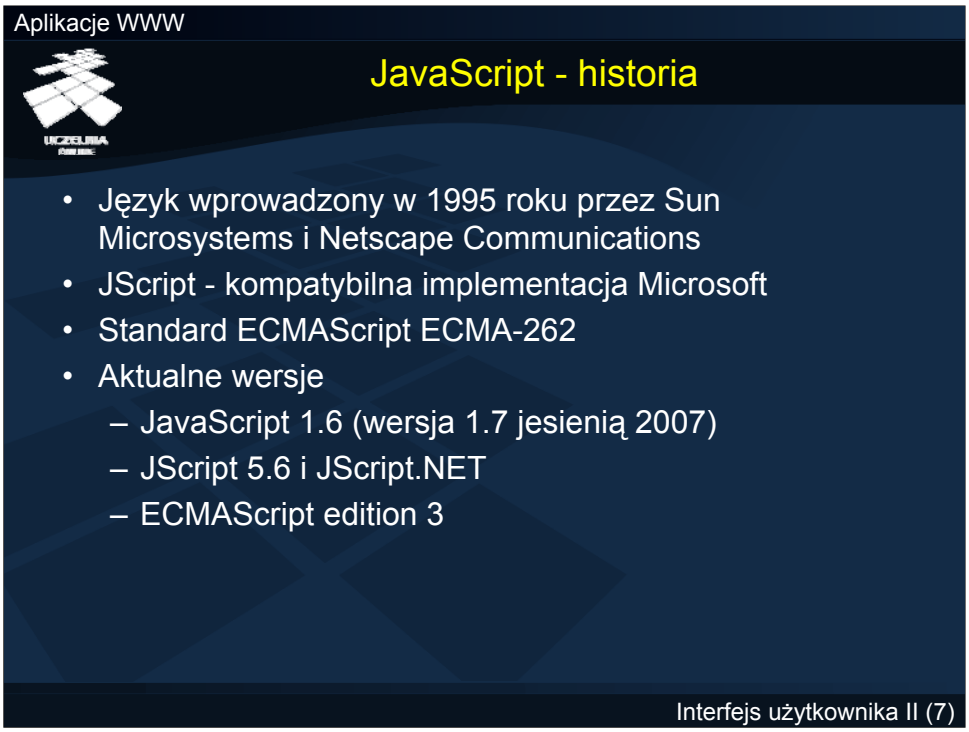
Atrybut action znacznika <form> określa adres URI (ang. Uniform Resource Identifier) agenta, który zajmie się przetwarzaniem danych wysłanych z formularza. Zawartość formularza może być wysyłana w formie listu elektronicznego pod wskazany adres:
<form action="mailto:John.Smith@acme.com" method="post">

Naciśnięcie przycisku <input type="submit"> spowoduje otwarcie klienta poczty elektronicznej i przesłanie zawartości formularza w postaci listu elektronicznego. Najczęściej jednak zawartość formularza jest przesyłana za pomocą protokołu HTTP do serwera HTTP i tam przetwarzana:
<form action="http://acme.com/form.cgi" method="get">

Dane z formularza mogą być przesłane wg. jednej z dwóch metod:

• method="GET": dane są kodowane w postaci łańcucha znaków i doklejane do adresu URL podanego w atrybucie action w postaci par nazwa atrybutu=wartość atrybutu, poszczególne atrybuty są odseparowane od siebie znakiem & a spacje są reprezentowane przez znak +, np. http://acme.com/form.cgi?osoba=Mikolaj+Morzy&miasto=Poznan&uczelnia=Politechnika+Poznanska, długość łańcucha znaków nie może przekroczyć maksymalnej długości zmiennej systemowej, poprawnie kodowane są tylko znaki ASCII

• method="POST": dane są przesyłane wewnątrz komunikatu protokołu HTTP



Język JavaScript został opracowany (początkowo pod nazwami Mocha i LiveScript) przez firmę Netscape Communications i wprowadzony do przeglądarki Netscape w 1995. W roku 1996 firma Microsoft wprowadziła własną implementację języka, nazwaną JScript, do przeglądarki Internet Explorer 3.0. Próba standaryzacji doprowadziła w roku 1997 do opracowania standardu ECMA-262, w którym język został nazwany ECMAScript. Język ECMAScript, który może być traktowany jako "standardowa" wersja języka JavaScript, jest wspierany pod postacią licznych dialektów w wielu różnych aplikacjach:

• JavaScript 1.6: Mozilla, Netscape, Firefox, Opera

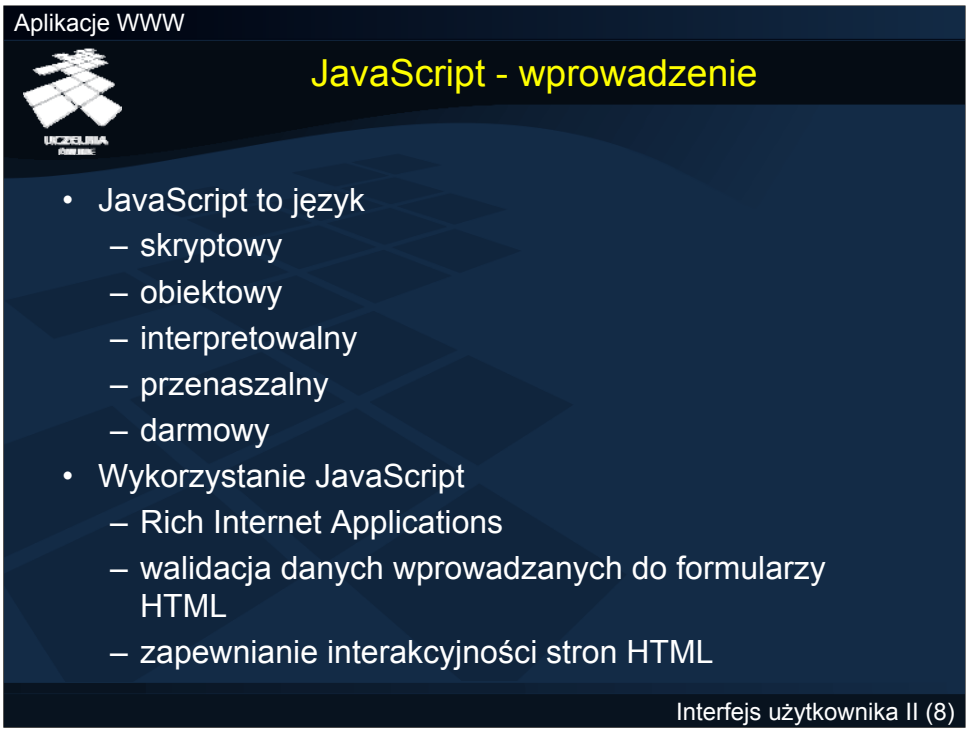
• JavaScript 1.5: KHTML, Adobe Acrobat

• JavaScript 1.4: platforma OpenLaszlo

• JScript 5.6: Internet Explorer, Opera

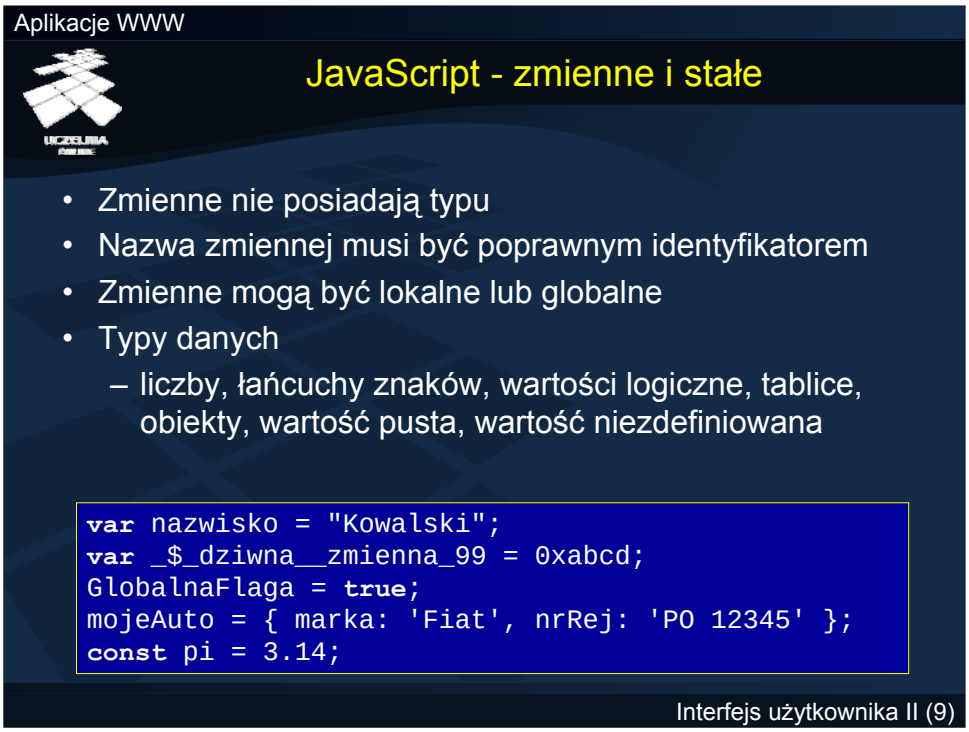
• JScript.NET 8.0: platforma .NET

• ActionScript 2: Macromedia Flash



Język JavaScript jest przykładem języka skryptowego. JavaScript jest językiem zorientowanym obiektowo i umożliwia rozszerzanie funkcjonalności języka przez wykorzystanie zewnętrznych bibliotek. JavaScript nie jest samodzielnym językiem programowania, co oznacza, że nie można za pomocą JavaScript pisać aplikacji użytkownika, programy napisane w JavaScript wykonują się w ramach (są zagnieżdżone) innych aplikacji. Programy JavaScript nie wymagają wcześniejszej kompilacji, są interpretowane i wykonywane w trybie runtime. Poszczególne dialekty JavaScript są do siebie bardzo podobne, co umożliwia stosunkowo łatwe przenoszenie aplikacji napisanych w JavaScript między różnymi środowiskami. Istotną zaletą języka JavaScript jest fakt, że jest to język całkowicie darmowy i jego wykorzystanie nie niesie ze sobą żadnych obciążeń finansowych.

Najczęściej język JavaScript jest wykorzystywany do pisania aplikacji wykonujących się w ramach przeglądarki internetowej. JavaScript może być wykorzystany zarówno do tworzenia tzw. bogatych aplikacji internetowych (ang. Rich Internet Applications), jak również jako narzędzie pomocnicze w tradycyjnych aplikacjach internetowych. W drugim przypadku JavaScript najczęściej służy do zapewniania interakcyjności stronom HTML (dzięki zdolności do reagowania na zdarzenia generowane przez użytkownika) oraz do walidacji poprawności danych wprowadzanych przez formularz HTML.



Język JavaScript jest przykładem języka słabo typowanego. Zmienne mogą przechowywać wartości dowolnego typu a konwersja między typami odbywa się automatycznie. Nazwa zmiennej musi być poprawnym identyfikatorem, tj. musi rozpoczynać się od znaku podkreślenia, znaku dolara bądź litery, po których mogą też nastąpić cyfry. JavaScript zwraca uwagę na wielkość liter, zatem zmienne MojeNazwisko i mojenazwisko to dwie różne zmienne.

Zasięg dostępności zmiennych może być lokalny lub globalny. Zmienne zadeklarowane wewnątrz definicji funkcji z wykorzystaniem słowa kluczowego var mają charakter lokalny i nie są dostępne poza funkcją, w której zostały zadeklarowane. Zmienne zadeklarowane poza jakąkolwiek funkcją oraz zmienne zadeklarowane bez użycia słowa kluczowego var mają charakter globalny i mogą być dostępne w dowolnej części programu (np. przez dowolny skrypt JavaScript zagnieżdżony w dokumencie HTML).

Zmienne JavaScript mogą przyjmować wartości następujących typów:

• liczby: liczby całkowite (100), liczby zmiennoprzecinkowe (100.01), liczby oktalne (0144), liczby heksadecymalne (0x64)

• łańcuchy znaków: ujęte w apostrofy lub cudzysłowy, możliwe wykorzystanie znaków specjalnych \n \t \r itp.

• wartości logiczne: reprezentowane przez literały true i false

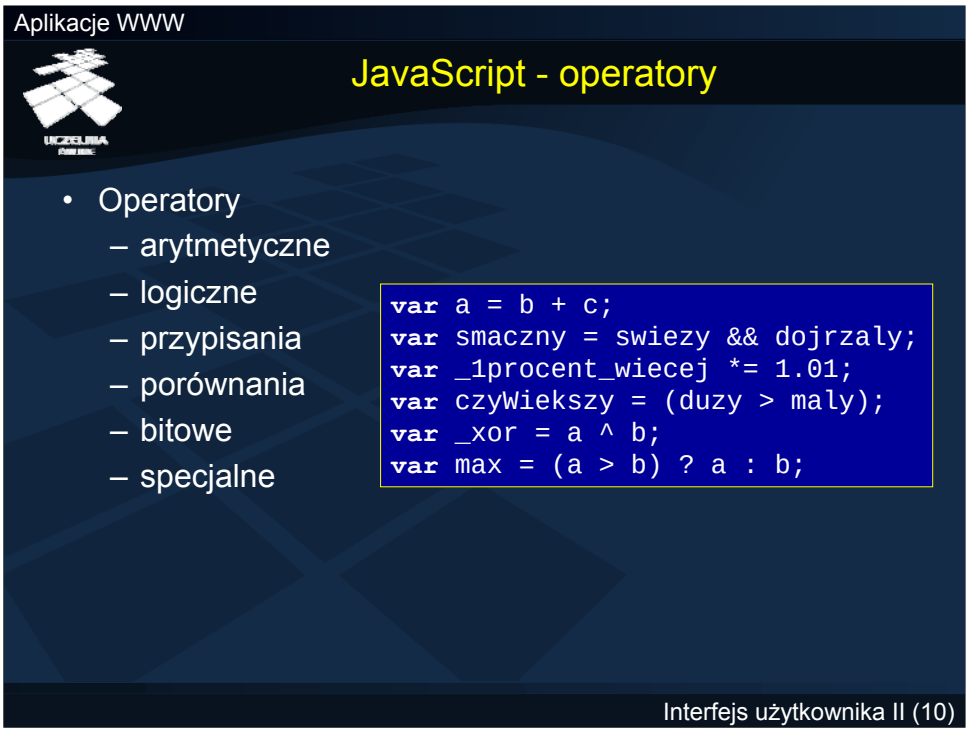
• tablice

• obiekty

• wartość pusta: reprezentowana przez literał null

• wartość niezdefiniowana: wykorzystywana gdy zmiennej lub elementowi tablicy nie przypisano wartości

Przypisanie wartości odbywa się przez wykorzystanie operatora =. Stałe deklaruje się przez wykorzystanie słowa kluczowego const.



Język JavaScript oferuje następujący zbiór operatorów:

• operatory arytmetyczne: dodawanie +, odejmowanie -, dzielenie /, mnożenie *, reszta z dzielenia %, inkrementacja ++, dekrementacja --, negacja -

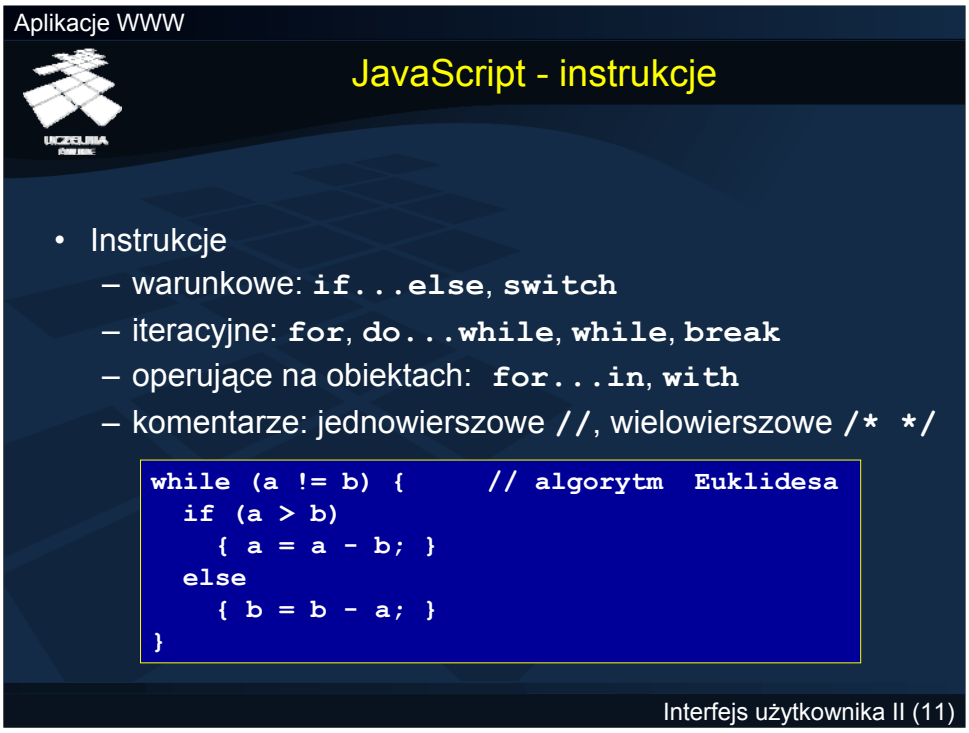
• operatory logiczne: koniunkcja &&, dysjunkcja ||, negacja ! (uwaga! JavaScript stosuje krótkie wartościowanie operatorów logicznych)

• operatory przypisania: podstawowy operator przypisania =, zmodyfikowane operatory przypisania += -= *= /= itp.

• operatory porównania: identyczny ===, równy ==, nieidentyczny !==, różny !=. większy >, mniejszy <, niemniejszy >=, niewiększy <=

•operatory bitowe: iloczyn &, suma |, różnica symetryczna ^, negacja ~, przesunięcie w lewo <<, przesunięcie w prawo <<

• operatory specjalne: konkatenacja łańcuchów znaków +, operator warunkowy (warunek) ? prawda : fałsz, operator kontroli typu typeof,



Język JavaScript zawiera instrukcje warunkowe, iteracyjne oraz operujące na obiektach. Podstawową instrukcją warunkową jest instrukcja `if...else` przedstawiona na slajdzie. Popularną instrukcją warunkową jest instrukcja wyboru warunkowego switch umożliwiająca wybór wykonywanego bloku w zależności od wartości zmiennej sterującej.

Podstawowymi instrukcjami iteracyjnymi są: pętla for wykonująca się określoną liczbę razy oraz pętla while wykonująca się tak długo, jak długo spełniony jest warunek sterujący pętlą (patrz przykład na slajdzie). Poniżej zamieszczono przykład pętli for:

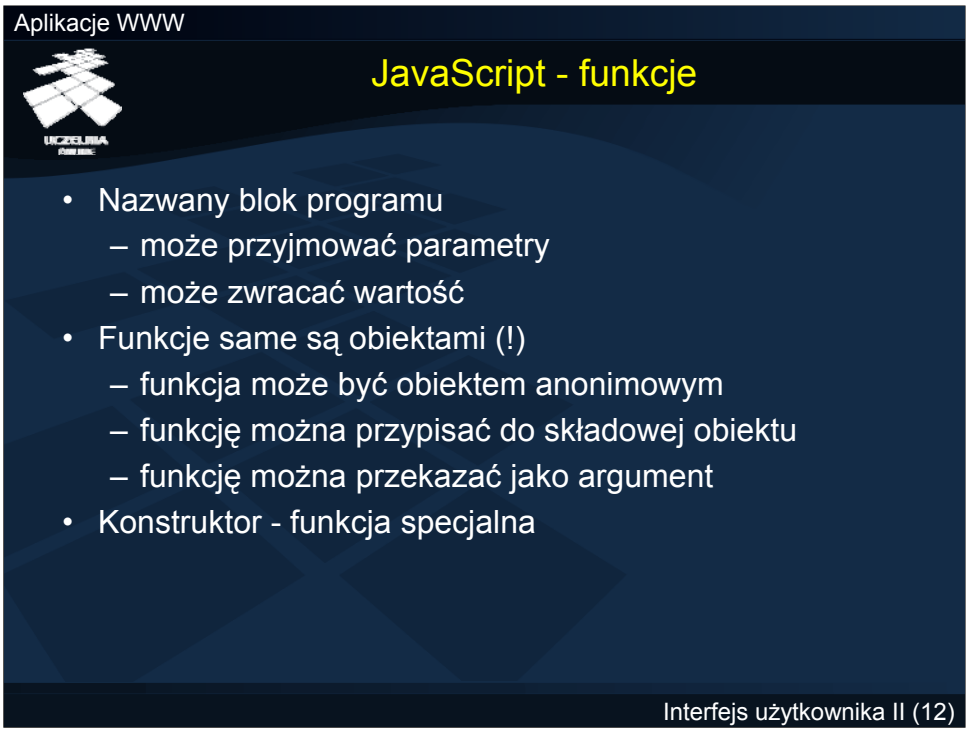
```
var n = 100;
var silnia = 1;
for (i = 1; i < n; i++)
  { silnia *= i; }
```

Do przerwania wykonywania pętli można wykorzystać instrukcje break (przerywa wykonanie pętli i przechodzi do wykonania pierwszego polecenia po pętli) oraz continue (przerywa bieżącą iterację pętli i przechodzi do kolejnej iteracji pętli).

Do iteracji po wszystkich własnościach obiektu lub wszystkich elementach tablicy można wykorzystać instrukcję for...in. Przykład wykorzystania tej instrukcji poniżej:

```
var liczbyPierwsze = [1,2,3,5,7,9,11,13,17,19];
var suma = 0;
for (var i in liczbyPierwsze)
  { suma += i; }
```

Komentarze w kodzie JavaScript można umieszczać na dwa sposoby. Do wprowadzenia komentarza jednowierszowego służą znaki // (wszystkie znaki od pierwszego wystąpienia // aż do końca bieżącej linii będą potraktowane jako komentarz). Komentarze wielowierszowe są wprowadzane przy pomocy znaków /* (otwarcie komentarza) i */ (zamknięcie komentarza)



Funkcją nazywamy nazwany blok programu, który może być wykonywany wielokrotnie i do którego dostęp odbywa się poprzez przyznaną nazwę. Definicja funkcji musi być poprzedzona słowem kluczowym function. Funkcja może przyjmować parametry. Lista parametrów podana podczas wywołania funkcji (lista parametrów aktualnych) nie musi być zgodna z listą parametrów podanych w deklaracji funkcji (listą parametrów formalnych). Parametry formalne, które nie zostaną dopasowane do parametrów aktualnych przyjmują wartość undefined. Wewnątrz funkcji można wykorzystywać parametry albo za pomocą ich nazw, albo za pomocą listy arguments, np. odwołanie do pierwszego parametru funkcji może mieć postać arguments[0]. JavaScript przekazuje parametry typów prostych (liczby, łańcuchy znaków, dane logiczne) przez wartość, natomiast obiekty są przekazywane przez referencję. Funkcja może (nie musi) zwracać wartość przy pomocy słowa kluczowego return. Poniżej przykład funkcji:

```
function Potega(podstawa, wykladnik) {
  var wynik = 1;
  for (i = 0; i < wykladnik; i++)
    { wynik *= podstawa; }
  return wynik;
}
```

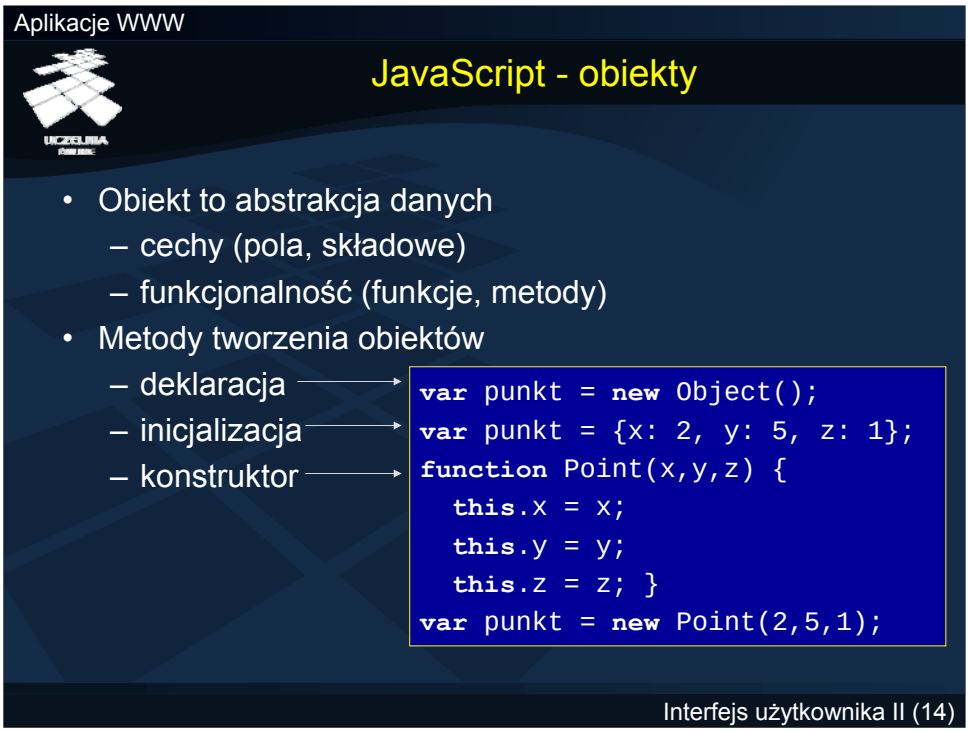
W języku JavaScript funkcje są obiektami tworzonymi na podstawie prototypu obiektu Function. Funkcja anonimowa może zostać przypisana do składowej obiektu, przykład takiego wykorzystania funkcji znajduje się poniżej:

```
Object.prototype.addTwoNumbers = function(a,b) { return a + b; }
myObject = new Object();
myObject.addTwoNumbers(2,2);
```

W powyższym przykładzie prototyp obiektu o nazwie Object został wzbogacony o nową metodę o nazwie addTwoNumbers, której implementacją jest funkcja dodająca do siebie dwa parametry wywołania. Każdy obiekt utworzony na podstawie prototypu Object posiada tę metodę i może ją wywoływać. Co więcej, funkcja może zostać przekazana jako argument wywołania do innej funkcji, jak zaprezentowano poniżej:

```
function sayHello() { return "Hello!"; }
function execute(funct) { funct(); }
execute(sayHello);
```

Specjalnym rodzajem funkcji jest konstruktor, czyli funkcja powołująca do życia nowe obiekty. Zastosowanie konstruktora zostanie omówione w dziale poświęconym prototypom.



Obiekt stanowi abstrakcję powiązanych ze sobą danych reprezentujących pewien byt świata rzeczywistego. Na obiekt składają się jego cechy (reprezentowane przez pola, zwane składowymi obiektu) oraz funkcjonalność (reprezentowana przez funkcje, zwane metodami).

Do tworzenia obiektów można wykorzystać prostą deklarację, tworzącą nowy nazwany obiekt na podstawie prototypu obiektu Object. Po utworzeniu obiektu można dowolnie dodawać składowe i metody. Inną metodą tworzenia obiektu jest inicjalizacja, podczas której tworzony jest nowy obiekt o podanych cechach, najczęściej cechom przypisywane są również początkowe wartości. Na etapie inicjalizacji można również określić metody tworzonego obiektu, albo na podstawie istniejących funkcji, albo na podstawie funkcji anonimowych. Wreszcie, do utworzenia obiektu można wykorzystać prototyp. Prototyp to struktura szkieletowa obiektu tworzona przez specjalną funkcję, zwaną konstruktorem. Nazwa konstruktora jest zawsze taka sama, jak nazwa prototypu. W powyższym przykładzie prototypem jest obiekt Point.

JavaScript - obiekty predefiniowane

- JavaScript zawiera zestaw przydatnych obiektów predefiniowanych:
  - Array
  - Boolean
  - Date
  - String
  - Function
  - Math
  - Number
  - RegExp

```
var zwierzaki  = new Array("pies","kot");
var czyKawaler = new Boolean(true);
var dzisiaj    = new Date();
var maleLitery = new String("ABC").small();
var czysc = new Function("document.clear");
var los    = new Math.random();
var wiek   = new Number.parseInt("18.234");
var kod    = new RegExp("\\d{2}-\\d{3}");
```


JavaScript oferuje kilka predefiniowanych obiektów znakomicie ułatwiających oprogramowywanie typowych zadań:

• Array: reprezentuje tablice, zawiera m.in. metody concat(), join(), pop(), push(), reverse(), shift(), slice(), sort(), splice() oraz składową *length*

• Boolean: stanowi obiektową reprezentację bazowego typu logicznego, zawiera metody toString(), valueOf()

• Date: ułatwia pracę z danymi reprezentującymi datę i czas, zawiera metody do pobierania informacji o składowych daty i czasu (getDay(), getMonth(), getFullYear(), getHours(), getMinutes(), getSeconds(), ...) oraz metody do ustawiania składowych daty i czasu (setDay(), setMonth(), setFullYear(), setHours(), setMinutes(), setSeconds(), ...). Dodatkowo, obiekt umożliwia konwersję daty i czasu na UTC.

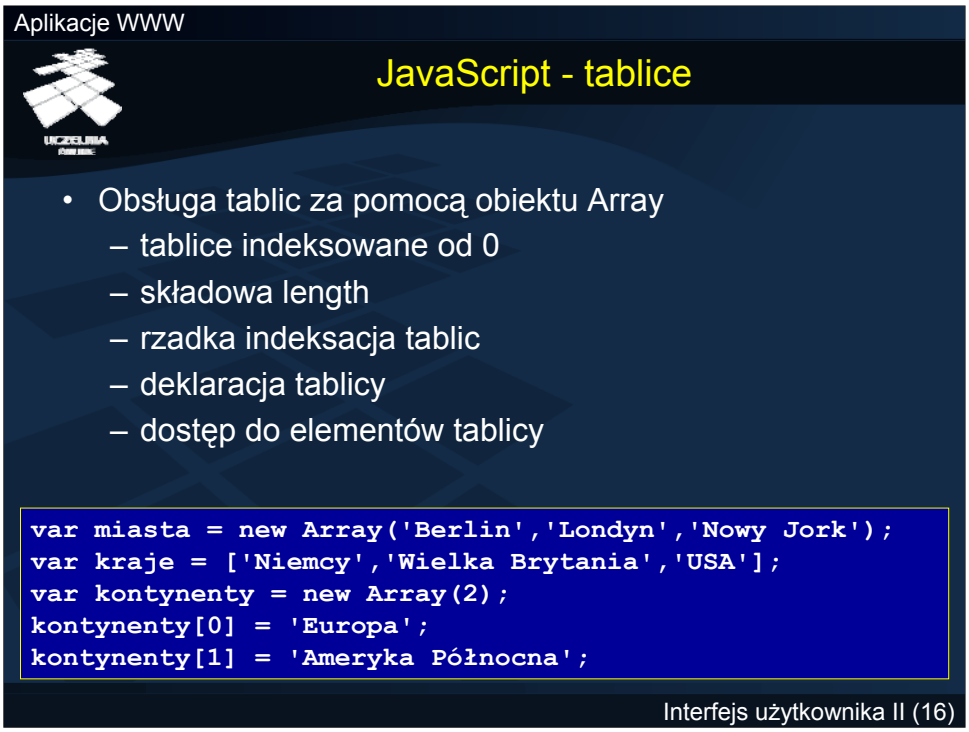
• String: obiekt umożliwiający manipulacje na łańcuchach znaków, zawiera wiele użytecznych metod, np. big(), bold(), concat(), indexOf(), link(), match(), replace(), search(), slice(), substr(), i wiele innych. Dodatkowo, obiekt posiada składową length zawierającą długość łańcucha znaków

• Function: obiekt reprezentujący prototyp wszystkich funkcji

• Math: obiekt ułatwiający wykonywanie operacji matematycznych, zawiera implementację wielu operacji matematycznych w postaci składowych, np. abs(), cos(), exp(), floor(), log(), max(), min(), pow(), random(), sin(), sqrt(). Obiekt Math zawiera też składowe reprezentujące często występujące stałe, np. E, LN2, LN10, PI, SQRT2

• Number: obiekt ułatwiający konwersję między typami, zawiera metody eval(), isNaN(), parseFloat(), parseInt() oraz składowe MAX_VALUE, MIN_VALUE, NaN, NEGATIVE_INFINITY, POSITIVE_INFINITY

• RegExp: obiekt ułatwiający korzystanie z mechanizmu wyrażeń regularnych (zostanie omówiony osobno)



Do obsługi tablic w JavaScript służy predefiniowany obiekt Array. Wszystkie tablice są indeksowane począwszy od 0. Składowa length przechowuje liczbę o 1 większą niż największy wykorzystany indeks tablicy (a nie liczbę elementów w tablicy!). Tablice w JavaScript są rzadkie, co oznacza, że tablica zajmuje w pamięci operacyjnej miejsce odpowiadające jedynie zainicjalizowanym elementom tablicy.

Tablica może zostać zadeklarowana na trzy sposoby. Po pierwsze, użytkownik może wykorzystać obiekt prototypowy Array i przekazać do konstruktora listę zawierającą początkowe elementy tablicy. Po drugie, użytkownik może wykorzystać notację z nawiasami kwadratowymi. Wreszcie, użytkownik może wykorzystać konstruktor obiektu Array do utworzenia pustej tablicy o zadanym rozmiarze, a następnie wypełnić poszczególne pozycje tablicy. Dodanie nowego elementu o indeksie przekraczającym aktualny rozmiar tablicy automatycznie powoduje zwiększenie rozmiaru tablicy.

Do pracy z tablicami można wykorzystać poniższe metody obiektu Array:

• concat(): łączy dwie lub więcej tablic

• join(): przekształca elementy tablicy na łańcuch znaków z podanym separatorem

• pop(): zwraca ostatni element tablicy, jednocześnie go usuwając

• push(): dodaje nowy element na końcu tablicy i zwraca nową długość tablicy

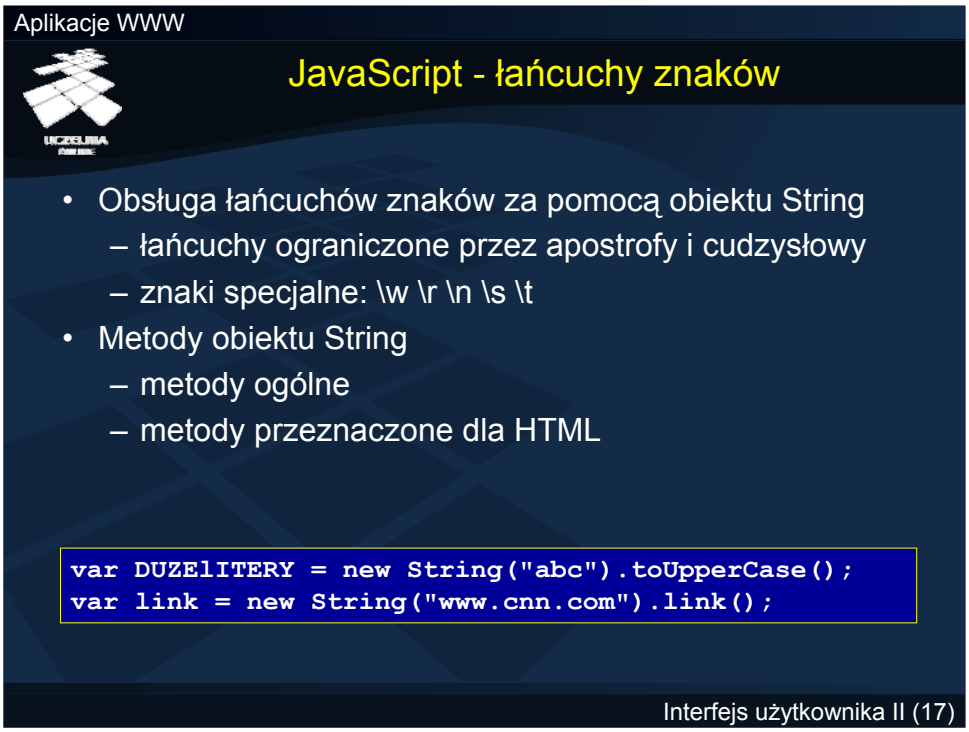
• reverse(): odwraca porządek elementów w tablicy

• shift(): zwraca pierwszy element tablicy, jednocześnie go usuwając

• slice(): zwraca wybrane elementy z tablicy

• sort(): porządkuje elementy tablicy

• splice(): dodaje i usuwa elementy tablicy



Do obsługi łańcuchów znaków JavaScript wykorzystuje predefiniowany obiekt String. Każdy łańcuch znaków jest ograniczony przez apostrofy lub cudzysłowy (znaki otwierający i zamykający muszą być takie same). W ramach łańcuchów znaków można wykorzystywać znaki specjalne:

• \w - dowolny znak alfanumeryczny

• \r - powrót karetki

• \n - nowa linia

• \s - dowolny biały znak (spacja, tabulacja, nowa linia)

• \t - znak tabulacji

Obiekt String zawiera wiele użytecznych metod, które można ogólnie podzielić na metody przeznaczenia ogólnego oraz metody zorientowane na przetwarzanie łańcuchów znaków w ramach dokumentów HTML. Pierwsza grupa zawiera, między innymi, metody:

• charAt() - zwraca znak z podanej pozycji w łańcuchu

• indexOf() - zwraca indeks pierwszego wystąpienia podanego podłańcucha znaków

• concat() - łączy dwa łańcuchy znaków

• match() - poszukuje podanej wartości w łańcuchu

• replace() - zamienia wybrane znaki w łańcuchu na inne znaki

• split() - rozbija łańcuch na tablicę elementów zgodnie z podanym separatorem

• substring() - zwraca podłańcuch danego łańcucha zgodnie z podanymi indeksami

• toLowerCase() - zamienia wszystkie znaki w łańcuchu na małe litery

• toUpperCase() - zamienia wszystkie znaki w łańcuchu na wielkie litery

Druga grupa to metody zorientowane na przetwarzanie tekstu w środowisku dokumentu HTML, metody te zwracają najczęściej łańcuch znaków wzbogacony o właściwe znaczniki HTML. Przykłady takich metod obejmują:

• anchor() - tworzy kotwicę na podstawie podanego łańcucha znaków

• big() - wyświetla łańcuch znaków korzystając z dużej czcionki ("tekst".big() odpowiada umieszczeniu w dokumencie HTML wpisu <big>tekst</big>)

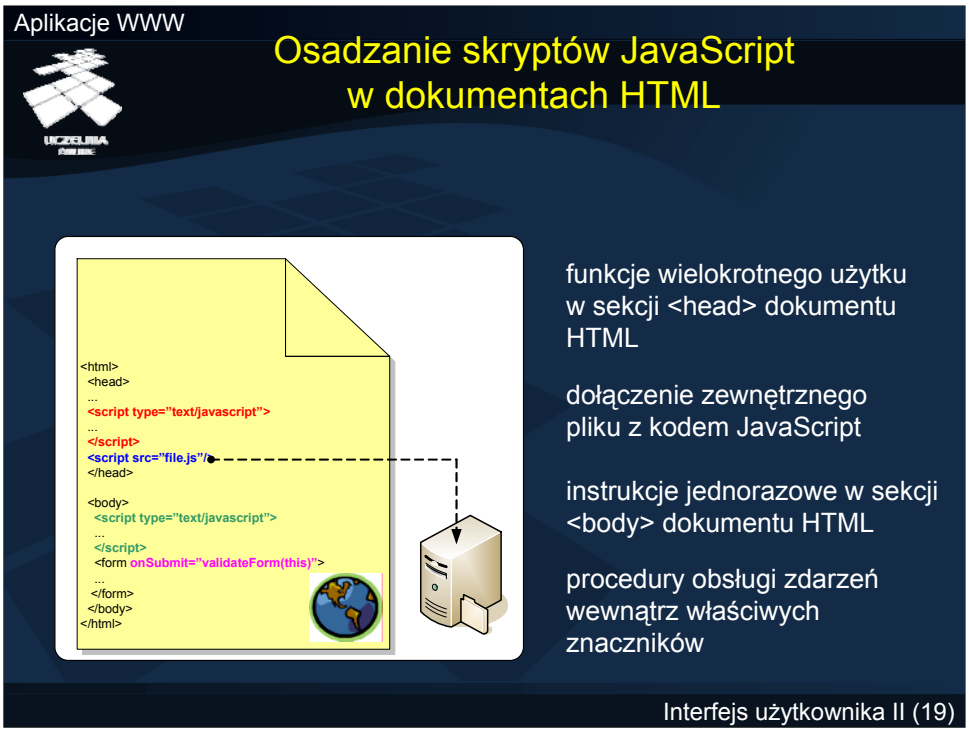
• bold() - wyświetla łańcuch znaków wytłuszczoną czcionką

• italics() - wyświetla łańcuch znaków pochyłą czcionką

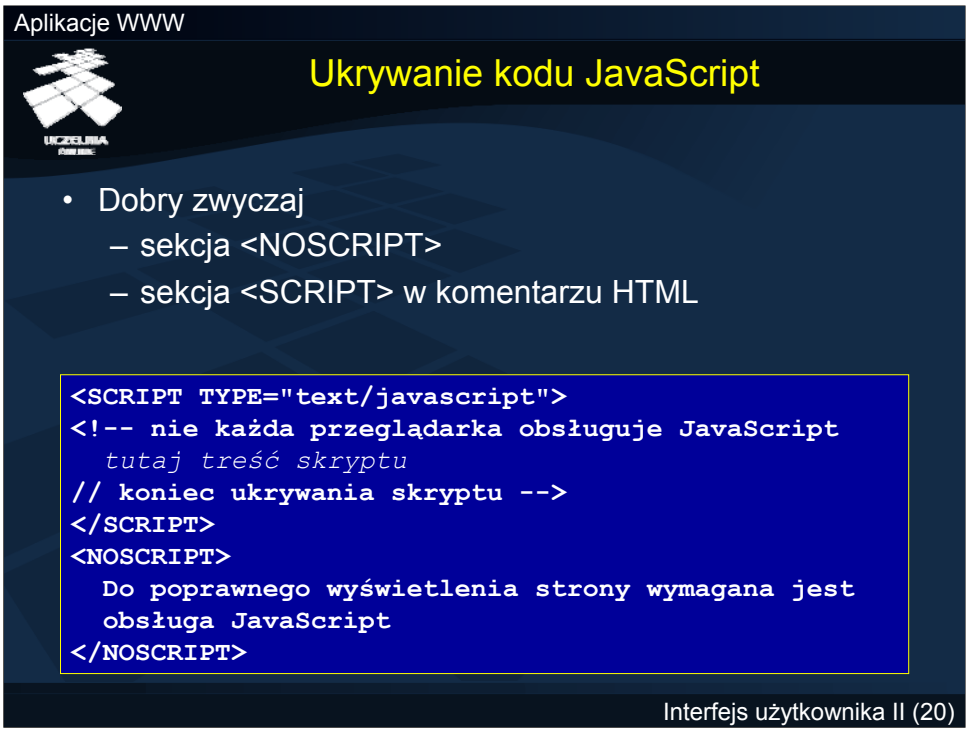
• link() - tworzy na podstawie łańcucha znaków odnośnik

• sub() - wyświetla łańcuch znaków jako indeks dolny

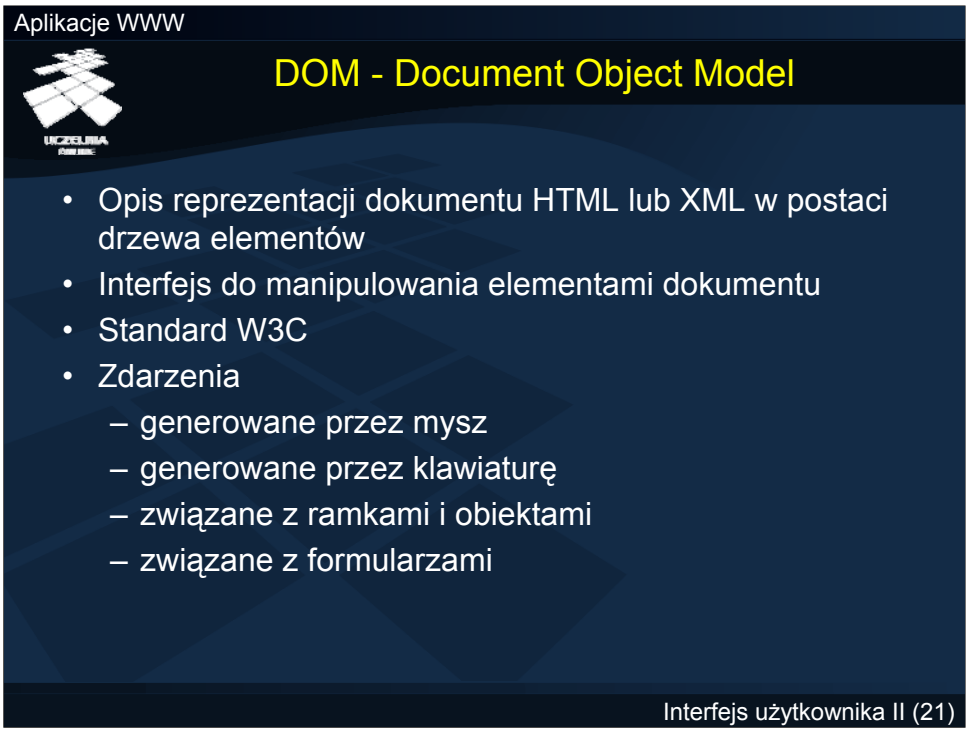
• sup() - wyświetla łańcuch znaków jako indeks górny



Kod JavaScript może być dołączony do dokumentu HTML na wiele sposobów. Najczęściej do włączenia kodu skryptu do dokumentu HTML wykorzystuje się znacznik <script>. Kod umieszczony w sekcji <head> dokumentu HTML jest dostępny z wszystkich części dokumentu. Opcjonalnie, kod JavaScript można umieścić w zewnętrznym pliku i dołączyć go za pomocą znacznika <script src="adres url pliku"/>. Pliki z kodem JavaScript powinny mieć rozszerzenie *.js i serwer HTTP powinien oznaczać ich typ MIME jako application/x-javascript. Fragmenty kodu JavaScript, które mają się wykonać tylko jeden raz można umieścić wewnątrz sekcji <body> dokumentu HTML. JavaScript może być wykorzystany do obsługi zdarzeń. Powiązanie funkcji obsługi zdarzeń ze zdarzeniami definiuje się we właściwych znacznikach.



Wielu klientów HTTP nie potrafi obsługiwać języka JavaScript. Dotyczy to zarówno starych wersji przeglądarek, jak i narzędzi współczesnych (np. przeglądarki w telefonach komórkowych, palmtopach, przeglądarki tekstowe). Dobrym zwyczajem jest ukrywanie kodu JavaScript przed klientami HTTP, którzy nie potrafią poprawnie zinterpretować skryptu JavaScript. Do ukrywania skryptu służy znacznik <noscript>, który zawiera treść wyświetlaną w przypadku gdy klient HTTP nie obsługuje języka JavaScript. Innym rozwiązaniem jest ujęcie kodu skryptu w komentarz HTML. Klienci HTTP, którzy nie rozpoznają znacznika <script> ignorują ten znacznik, a całą jego zawartość traktują jak komentarz i nie wyświetlają na ekranie.



Popularność języka JavaScript spowodowała opracowanie przez twórców różnych przeglądarek niezależnych od siebie sposobów reprezentacji dokumentów HTML i interfejsów umożliwiających manipulowanie indywidualnymi elementami dokumentów. W 1998 roku World Wide Web Consortium (W3C) opracowało standard reprezentacji dokumentów HTML i XML w postaci drzew oraz zaproponowało obiektowo zorientowany interfejs programistyczny do manipulacji na poszczególnych elementach. Do dnia dzisiejszego standard DOM doczekał się specyfikacji czterech poziomów zgodności i stał się de facto powszechnie uznanym standardem.

Standard DOM zawiera również specyfikację zdarzeń i odwzorowanie zdarzeń na elementy dokumentu, których zdarzenia dotyczą. Zdarzenia można ogólnie podzielić na cztery zasadnicze grupy:

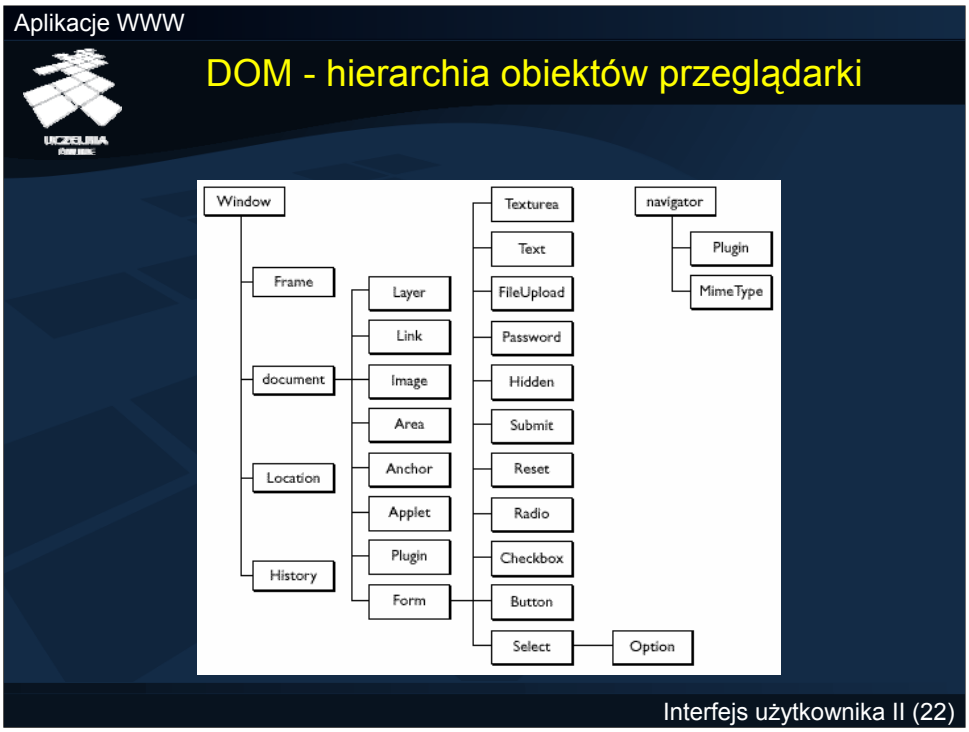
•zdarzenia generowane przez mysz: kliknięcie (onClick), podwójne kliknięcie (onDblClick), naciśnięcie elementu (onMouseDown), zwolnienie przycisku myszy (onMouseUp), najechanie myszą (onMouseOver), przesuwanie myszą nad elementem (onMouseMove), opuszczenie myszą obszaru nad elementem (onMouseOut)

•zdarzenia generowane przez klawiaturę: wciśnięcie i zwolnienie klawisza (onKeyPress), wciśnięcie klawisza (onKeyDown), zwolnienie klawisza (onKeyUp)

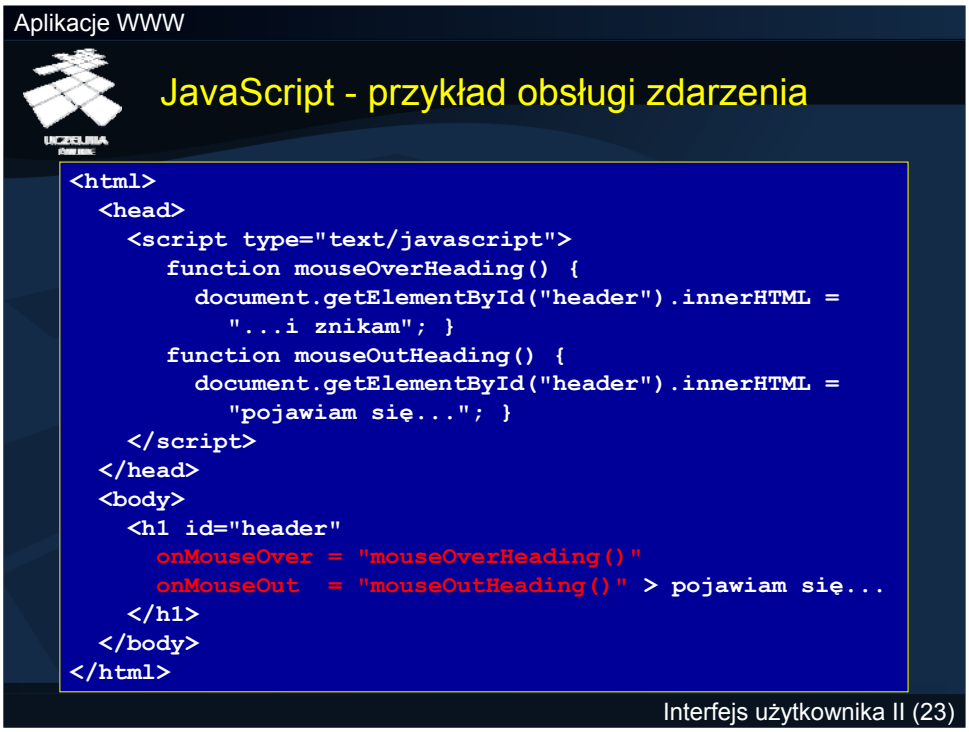
•zdarzenia związane z obiektami i ramkami: załadowanie (onLoad), przerwanie ładowania (onAbort), wystąpienie błędu ładowania (onError), zmiana rozmiaru okna/ramki (onResize), przesunięcie widocznego okna (onScroll)

•zdarzenia związane z formularzami: zaznaczenie tekstu (onSelect), zmiana wartości pola (onChange), przesłanie formularza (onSubmit), wyczyszczenie formularza (onReset), wejście do elementu (onFocus), wyjście z elementu (onBlur)

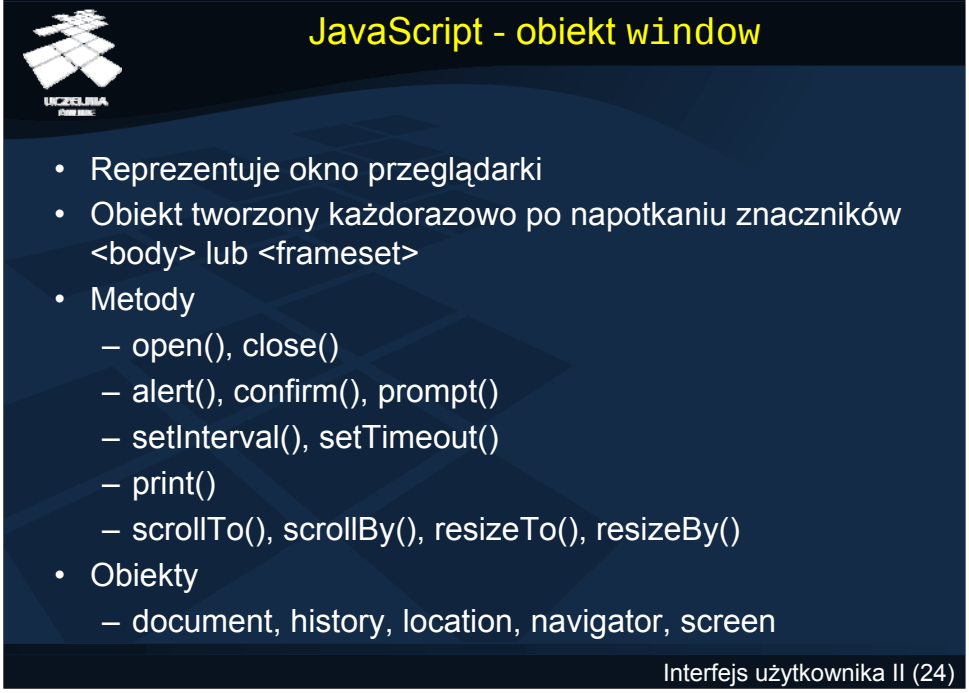
Oprócz wyżej wymienionych istnieje także wiele zdarzeń obsługiwanych tylko przez wybrane przeglądarki. Przykładowo, Internet Explorer obsługuje także zdarzenia związane z obsługą schowka (zdarzenia onCopy, onPaste, onCut), a Mozilla obsługuje zdarzenia generowane przez okienka typu pop-up (zdarzenia onPopupHiding, onPopupShowing).



W wyniku załadowania dokumentu HTML do przeglądarki powstaje drzewo DOM (ang. Document Object Model) zawierające obiektowo-zorientowany obraz dokumentu. Na obraz składa się hierarchia obiektów, zawierająca zarówno obiekty reprezentujące elementy składowe dokumentu HTML (np. obiekty Image, Applet, Form, Text, Radio, Checkbox, itp.), jak i obiekty reprezentujące stan przeglądarki (np. obiekty Location, History, Plugin). Manipulacja wartościami składowych tych obiektów pozwala na programową kontrolę nad wyglądem i zachowaniem dokumentu HTML, w szczególności pozwala na reagowanie na działania użytkownika (poprzez obsługę zdarzeń) i dokonywanie adekwatnych zmian w dokumencie HTML.



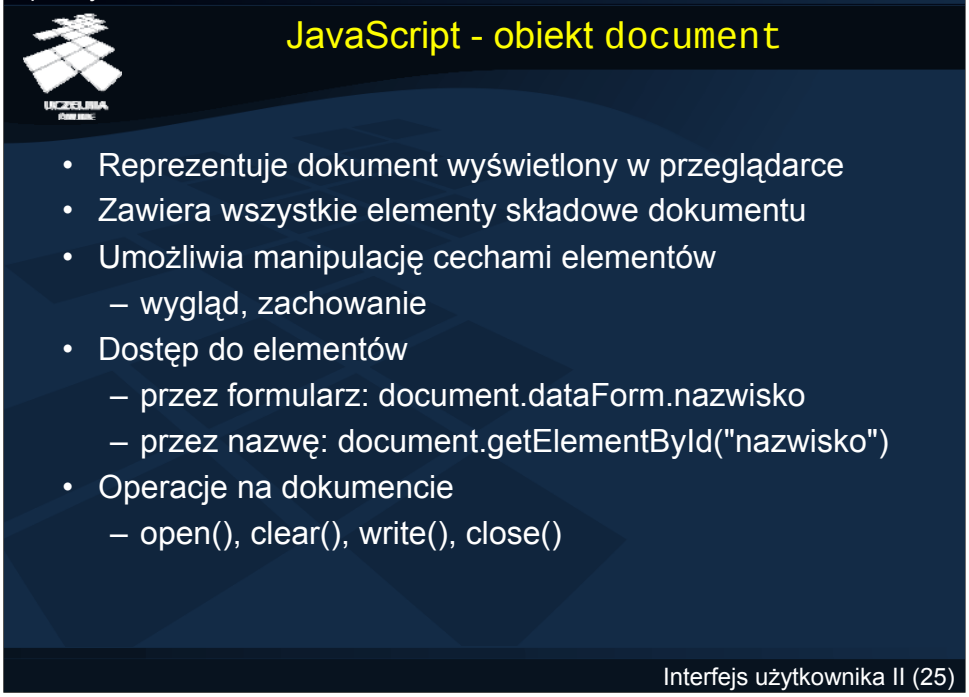
Na slajdzie zaprezentowano przykład obsługi zdarzeń związanych z ruchem myszy. W ciele dokumentu HTML znajduje się nagłówek pierwszego poziomu o identyfikatorze "header". W znaczniku <h1> zdefiniowano procedurę obsługi zdarzenia polegającego na najechaniu myszy na nagłówek (funkcja mouseOverHeading()) oraz procedurę obsługi zdarzenia opuszczenia obszaru nad nagłówkiem przez mysz (funkcja mouseOutHeading()). Obie funkcje zostały zdefiniowane w sekcji <head> dokumentu. Działanie obu procedur obsługi zdarzenia polega na zamianie tekstu wyświetlanego wewnątrz nagłówka. W drzewie DOM nagłówek zostaje zlokalizowany poprzez metodę getElementById obiektu document. Tekst wyświetlany wewnątrz elementu jest reprezentowany przez składową innerHTML, zmiana tej wartości znajduje natychmiastowe odzwierciedlenie w obrazie dokumentu wyświetlanym przez przeglądarkę.



Obiekt window reprezentuje okno przeglądarki. Obiekt jest tworzony przy każdym napotkaniu znacznika <body> lub <frameset>. Podstawowa funkcjonalność obiektu umożliwia takie czynności jak: otwieranie (open()) i zamykanie (close()) okna, wyświetlanie okien dialogowych z komunikatem (alert()), okien dialogowych z pytaniem (confirm()), okien dialogowych z polem do wprowadzenia wartości (prompt()), uruchamianie funkcji w określonych interwałach czasowych (setInterval()), uruchamianie funkcji po upłynięciu określonego czasu (setTimeout()), drukowanie zawartości okna (print()), przesunięcie zawartości o pewną liczbę pikseli (scrollBy()) lub przesunięcie zawartości do podanych współrzędnych (scrollTo()), zmianę rozmiaru okna o pewną liczbę pikseli (resizeBy()) lub zmianę rozmiaru okna do podanych współrzędnych (resizeTo()). Oprócz wymienionych metod obiekt window stanowi korzeń hierarchii obiektów DOM i zawiera w sobie pozostałe obiekty, np. document, history, location, navigator, screen, itp.

Poniżej przykład wykorzystania obiektu window:

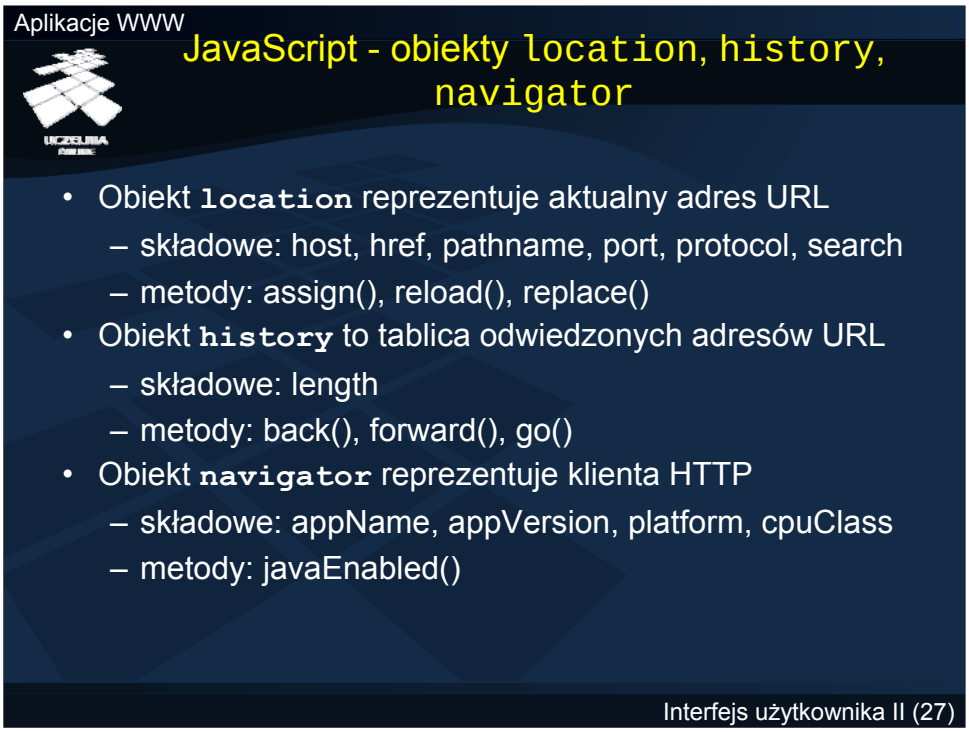
```
<html>
  <head>
    <script type="text/javascript">
      function info() { window.alert("Zmieniono rozmiar okna!"); }
      window.onresize = info;
    </script>
  </head>
  <body onLoad="window.open('http://www.javascript.com','_blank')">
  </body>
</html>
```



Obiekt document reprezentuje dokument aktualnie wyświetlony w oknie przeglądarki. Obiekt document zawiera w sobie wszystkie obiekty reprezentujące elementy umieszczone w dokumencie, takie jak: formularze, tabele, nagłówki, sekcje, itp. Za pomocą obiektu document można dynamicznie manipulować zawartością wyświetlanego dokumentu. Modyfikacjom podlegają zarówno cechy wizualne elementów (np. bgColor, fgColor) jak i cechy behawioralne (np. lastModified, checked). Najczęściej zmianom programistycznym podlegają elementy formularzy. Dokument HTML może zawierać wiele formularzy. Każdy formularz jest reprezentowany przez obiekt form, a wszystkie formularze są dostępne przez tablicę document.forms[]. Można również posłużyć się nazwą formularza, np. document.dataForm. Elementy formularza są dostępne przez tablicę document.dataForm.elements[] lub bezpośrednio przez swoją nazwę, np. document.dataForm.nazwisko. Istnieje również możliwość zlokalizowania elementu o podanym identyfikatorze przez wywołanie metody document.getElementById(). Obiekt document dostarcza także użytecznych metod służących do operowania na dokumencie i umożliwia, między innymi, otwarcie dokumentu (open()), wyczyszczenie dokumentu (clear()), wpisanie łańcucha znaków do dokumentu (write()), oraz zamknięcie dokumentu (close()).

Poniżej przykład zastosowania obiektu document:

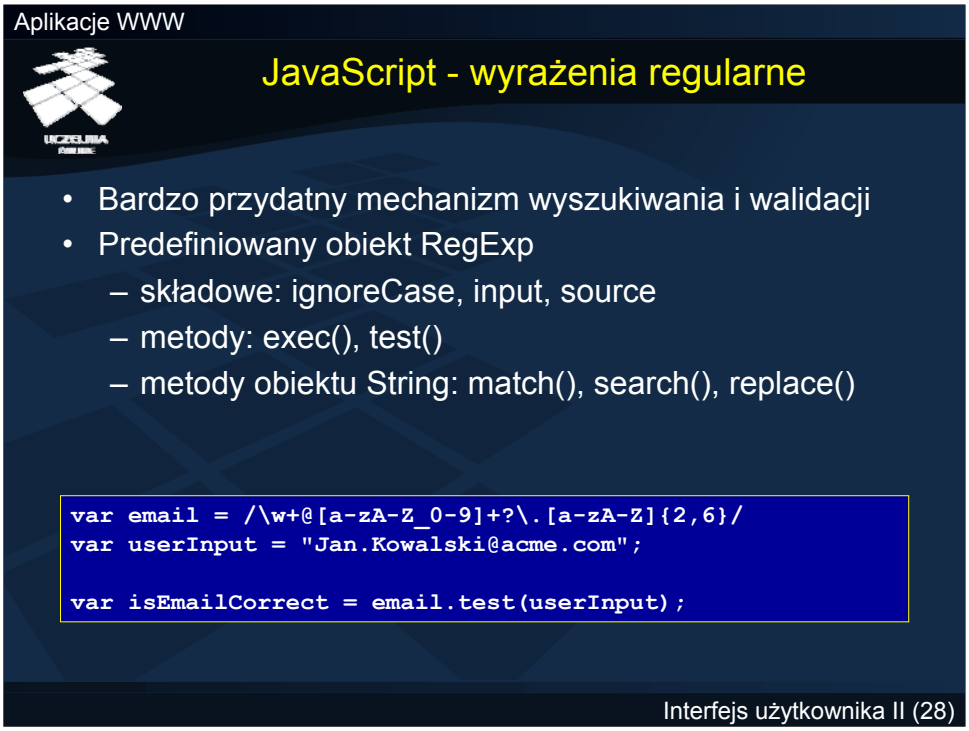
```
<html>
<head>
  <script type="text/javascript">
    function verify() {
      document.clear();
      if (document.getElementById("nazwisko").value == "")
        { document.write("Podaj nazwisko!"); return false; }
      else
        { document.write("Dziękuję za podanie nazwiska"); return true; }
      document.close();
    }
  </script>
  </head>
  <body>
    <form name="dataForm" onsubmit="return verify()">
      Nazwisko: <input name="nazwisko" type="text" id="nazwisko"/>
                    <input type="submit" value="wyślij"/>
    </form>
  </body>
</html>
```



Obiekt location reprezentuje adres URL aktualnie wyświetlany w kliencie HTTP. Obiekt ten umożliwia odczytanie poszczególnych składowych adresu URL (np. nazwa lub adres IP hosta, ścieżka, numer portu, protokół, część adresu URL na prawo od znaku '?') oraz pozwala na manipulowanie adresem URL: załadowanie nowego adresu (assign()), przeładowanie aktualnego adresu (reload()), oraz zamianę na inny adres (replace()).

Obiekt history to tablica zawierająca uporządkowane chronologicznie odwiedzone adresy URL. Obiekt history umożliwia nawigację po elementach tablicy: cofnięcie się do poprzedniego elementu (back()), przejście do następnego elementu (forward()), oraz przejście do dowolnego elementu (go()).

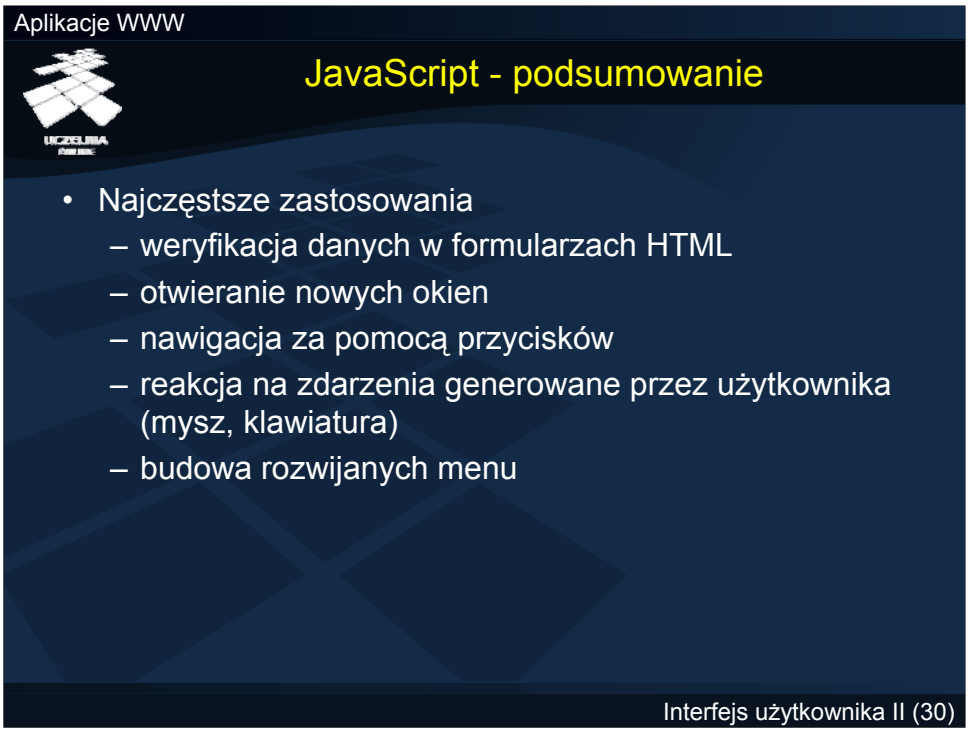
Obiekt navigator reprezentuje klienta HTTP. Cechy obiektu reprezentują wszystkie informacje udostępniane przez klienta HTTP, między innymi nazwę klienta (appName), wersję klienta (appVersion), nazwę systemu operacyjnego (platform), klasę procesora (cpuClass), języki używane w systemie operacyjnym (systemLanguage) i kliencie HTTP (browserLanguage), itp. Dodatkowo, obiekt informuje za pomocą metody javaEnabled() o obecności wtyczki obsługującej język Java.



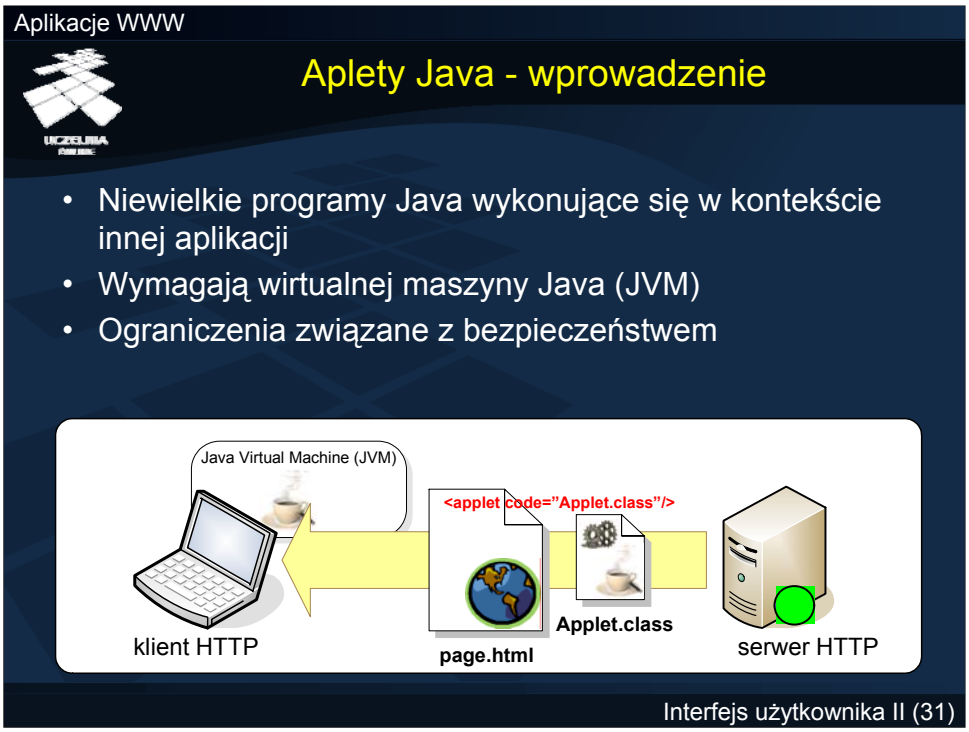
Wyrażenia regularne to bardzo potężny i przydatny mechanizm wyszukiwania i walidacji danych. Wyrażenia regularne pozwalają na proste konstruowanie złożonych warunków wyszukiwania tekstu oraz umożliwiają walidację i substytucję na podstawie konstruowanych masek i wzorców. JavaScript zawiera interpreter wyrażeń regularnych specyfikowanych w postaci rozpowszechnionej przez system Unix. Predefiniowany obiekt RegExp zawiera składowe i metody ułatwiające pracę z wyrażeniami regularnymi. Przykładowymi składowymi są składowe ignoreCase (informacja czy wyrażenie regularne jest czułe na wielkość liter), input (zawiera łańcuch znaków testowany na zgodność z wyrażeniem regularnym) czy source (zawiera źródło wyrażenia regularnego). Metody obiektu RegExp to wyszukiwanie dopasowania z wyrażeniem regularnym (exec()), sprawdzenie czy łańcuch znaków jest zgodny z wyrażeniem regularnym (test()). Obiekt String zawiera też trzy metody współpracujące z wyrażeniami regularnymi: wyszukanie dopasowania do wyrażenia regularnego (match() i search()) oraz zastąpienie wystąpienia wyrażenia regularnego (replace()).

Poniżej przykładowy kod testujący poprawność adresu email:

```
<html>
<head>
 <script type="text/javascript">
 function verify() {
   var email = /\w+@[a-zA-Z_0-9]+?\.[a-zA-Z]{2,6}/;
   var input = document.dataForm.email.value;
   var isCorrect = email.test(input);
   if (isCorrect)
    { window.alert("Adres email jest poprawny"); return true; }
   else
    { window.alert("Adres jest niepoprawny!"); return false; }
 }
 </script>
 </head>
 <body>
 <form name="dataForm" onsubmit="return verify()">
  Email: <input type="text" name="email"/>
          <input type="submit" value="sprawdź"/>
 </form>
 </body>
</html>
```

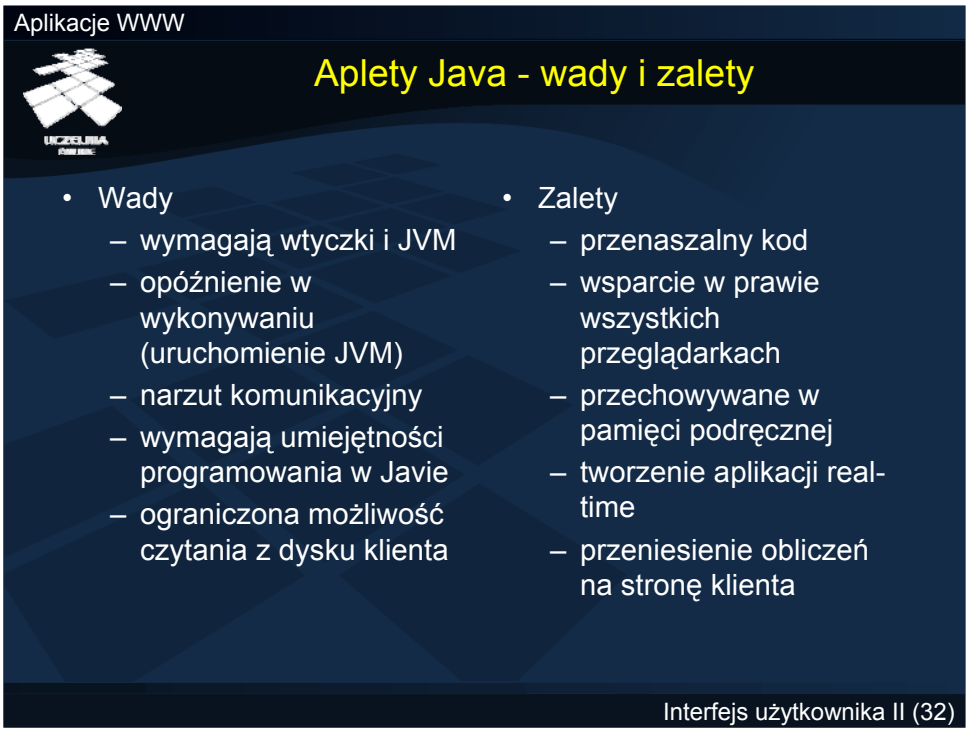


JavaScript jest bardzo popularnym i intensywnie rozwijanym językiem. Historycznie JavaScript był wykorzystywany przede wszystkim do zapewnienia podstawowej interakcyjności w statycznych dokumentach HTML. Typowe zastosowania obejmowały: tworzenie rozwijanych menu, dynamiczne konstruowanie rozwijalnych list wartości w formularzach, obsługę nawigacji za pomocą przycisków, czy reagowanie na czynności wykonywane przez użytkownika, np. poprzez poruszanie myszką. Wraz z upowszechnieniem się aplikacji internetowych szczególnego znaczenia nabrała kwestia weryfikacji poprawności danych przesyłanych za pomocą formularzy HTML. JavaScript posiada wydajne mechanizmy pozwalające na zaawansowaną kontrolę poprawności i jakości wprowadzanych danych, w szczególności możliwość obsługi zdarzeń oraz mechanizm wyrażeń regularnych.



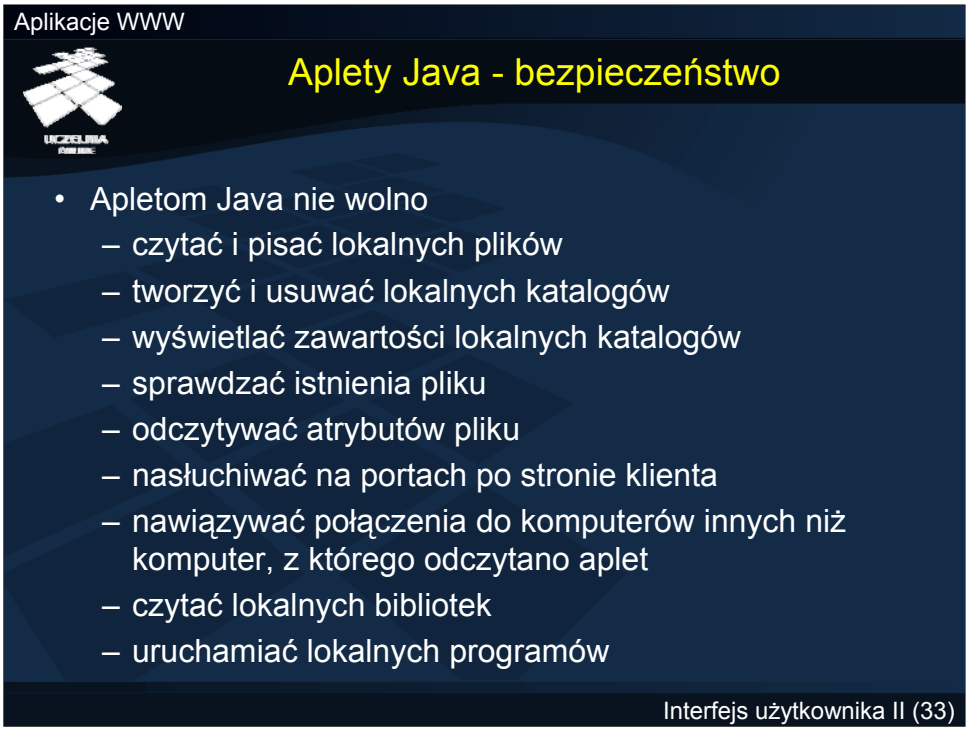
Aplet jest to niewielki program napisany w języku Java. Aplety nie mogą uruchamiać się samodzielnie, lecz wymagają środowiska uruchomieniowego. Środowiskiem uruchomieniowym może być dowolna aplikacja zaopatrzona w wirtualną maszynę Javy (ang. JVM, Java Virtual Machine), najczęściej aplikacją tą jest przeglądarka internetowa. We współczesnych przeglądarkach maszyna wirtualna jest instalowana zazwyczaj w postaci wtyczki (ang. plug-in). W celu uruchomienia apletu musi on zostać osadzony w dokumencie HTML. Wewnątrz dokumentu znajduje się znacznik <applet> lub <object> wraz ze wskazaniem na właściwy plik *.class (skompilowany pseudo-kod maszynowy Javy), który zostaje odczytany z serwera HTTP i przesłany do wirtualnej maszyny Java uruchomionej na komputerze będącym klientem HTTP.

Wykonanie apletu jest obwarowane wieloma ograniczeniami. Wiąże się to z faktem niewiadomego (najczęściej) pochodzenia apletu. W szczególności, aplet nie może odczytywać prawie żadnych danych lokalnych a interakcja z użytkownikiem jest ograniczona do pewnego obszaru ekranu.

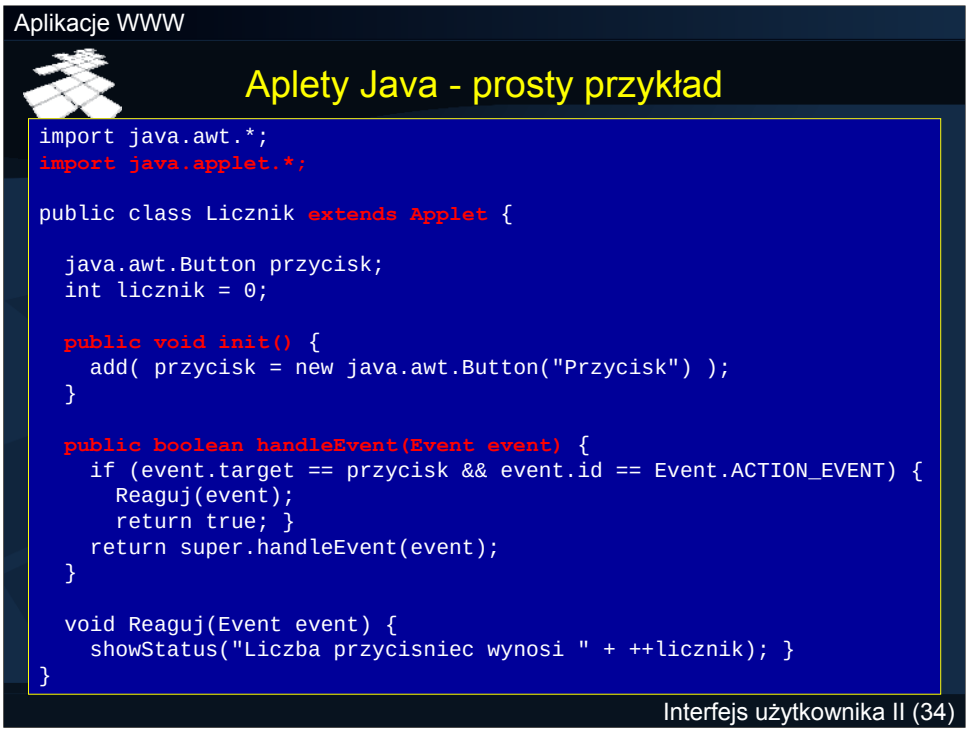


Podstawową wadą technologii apletów jest fakt, że do poprawnego funkcjonowania aplet wymaga obecności wirtualnej maszyny Java. Konieczność samodzielnego zainstalowania wtyczki oferującej JVM może przerastać umiejętności bardzo wielu użytkowników. Ponieważ aplet wykonuje się wewnątrz JVM, pierwsze uruchomienie się apletu jest opóźnione ze względu na oczekiwanie na załadowanie i gotowość JVM. W przypadku wielu aplikacji internetowych takie opóźnienie może być nieakceptowane. Jeśli aplet nie zostanie umieszczony w pamięci podręcznej klienta, to kolejne załadowanie apletu wymaga ponownego przesłania całego kodu, co również opóźnia działanie apletu. W porównaniu z wieloma innymi technologiami internetowymi, aplety są technologią dość złożoną i wymagającą biegłości w posługiwaniu się językiem Java. Wreszcie, z uwagi na względy bezpieczeństwa funkcjonalność apletów jest ograniczona przez liczne zakazy, w szczególności zakazy dotyczące pracy z plikami po stronie klienta.

Technologia apletów posiada również wiele zalet. Ponieważ pseudokod maszynowy Javy (tzw. bajtkod) jest w pełni przenaszalny między różnymi implementacjami wirtualnej maszyny Java, również aplety mogą być uruchamiane na wielu platformach sprzętowych i programowych bez konieczności wprowadzania zmian. Znaczniki do obsługi apletów, jak i same aplety, są poprawnie obsługiwane przez praktycznie wszystkie przeglądarki, więc technologia ma charakter uniwersalny. Kod apletu może być przechowywany po stronie klienta w pamięci podręcznej, co wydatnie przyspiesza czas kolejnych odwołań do apletu. Aplety umożliwiają także tworzenie aplikacji czasu rzeczywistego. Wreszcie, wykorzystanie apletów powoduje przeniesienie znacznej części pracy z serwera HTTP na stronę klienta, tym samym odciążając serwer HTTP i zwiększając skalowalność aplikacji.



Funkcjonalność apletów jest mocno ograniczona przez względy bezpieczeństwa. Aby uniknąć zagrożeń związanych z uruchamianiem anonimowego kodu po stronie klienta HTTP apletom nie wolno wykonywać praktycznie żadnych operacji na lokalnym systemie plików (czytać i pisać plików, tworzyć, usuwać i listować katalogów, poszukiwać plików i odczytywać ich atrybuty) oraz korzystać z interfejsu sieciowego (nasłuchiwać na portach lub nawiązywać połączenia do komputerów innych niż macierzysty serwer apletu). Apletom nie wolno także korzystać z lokalnych bibliotek i uruchamiać żadnych poleceń systemu operacyjnego ani odczytywać żadnych zmiennych i ustawień systemowych.



Aplet Java jest klasą dziedziczącą po klasie bazowej java.applet.Applet, która z kolei jest klasą potomną klas java.awt.Panel i java.awt.Container. Metoda init() jest wywoływana przez aplikację-hosta (np. przez przeglądarkę), aby poinformować aplet, że został w całości załadowany do maszyny wirtualnej Java. Każda klasa dziedzicząca z klasy java.applet.Applet powinna dostarczać własną implementację tej metody w celu inicjalizacji zmiennych i obiektów. Przykłady czynności wykonywanych w metodzie init() to: nawiązanie połączenia z bazą danych, utworzenie obiektów interfejsu użytkownika, inicjalizacja wątków w aplikacji wielowątkowej. Metoda handleEvent jest wywoływana przy każdym zajściu zdarzenia (w tym przypadku po kliknięciu na przycisk). Wreszcie metoda showStatus() wywoływana w metodzie Reaguj() powoduje wyświetlenie podanego łańcucha znaków w pasku statusu przeglądarki. Inne przydatne metody klasy java.applet.Applet to:

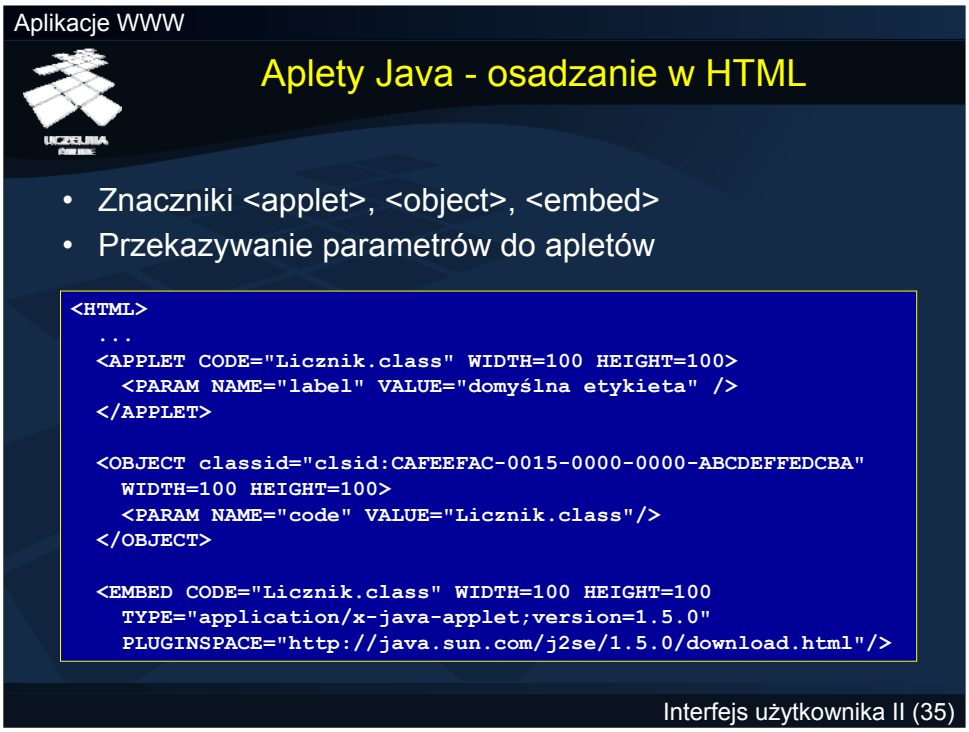
•destroy(): wołana przez aplikację-hosta przed zniszczeniem apletu, umożliwia zwolnienie zasobów

•getParameter(): zwraca parametr wywołania przekazany przez dokument HTML

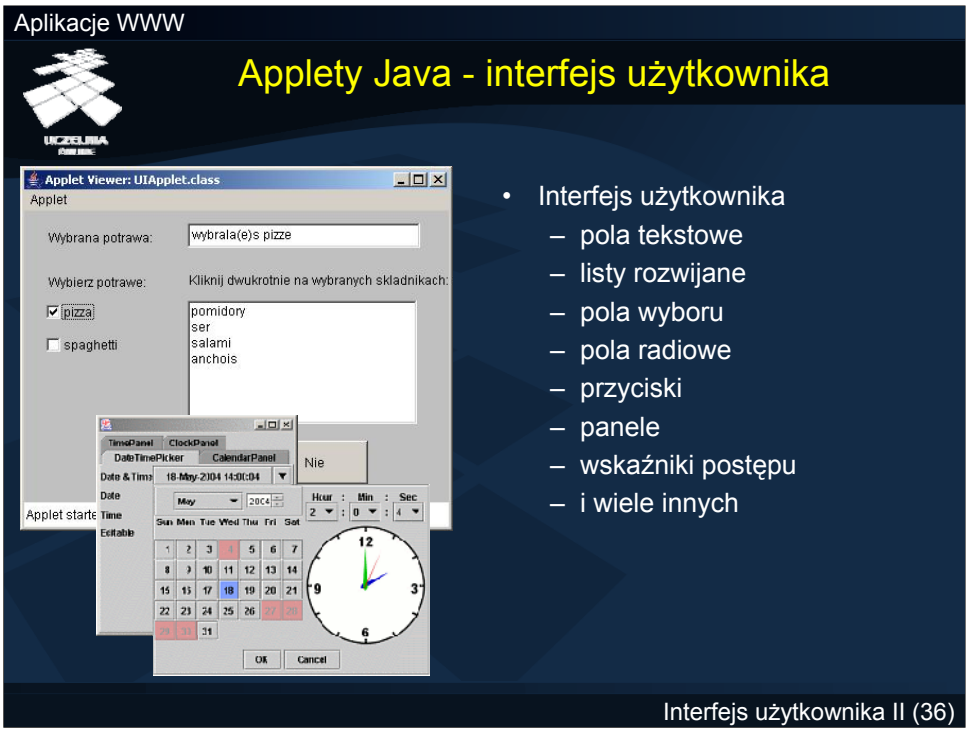
•start(): wołana przez aplikację-hosta tuż przed uruchomieniem apletu, np. przy każdym powrocie do dokumentu zawierającego aplet

•stop(): wołana przez aplikację-hosta tuż przed zatrzymaniem apletu, np. gdy użytkownik opuszcza dokument zawierający aplet

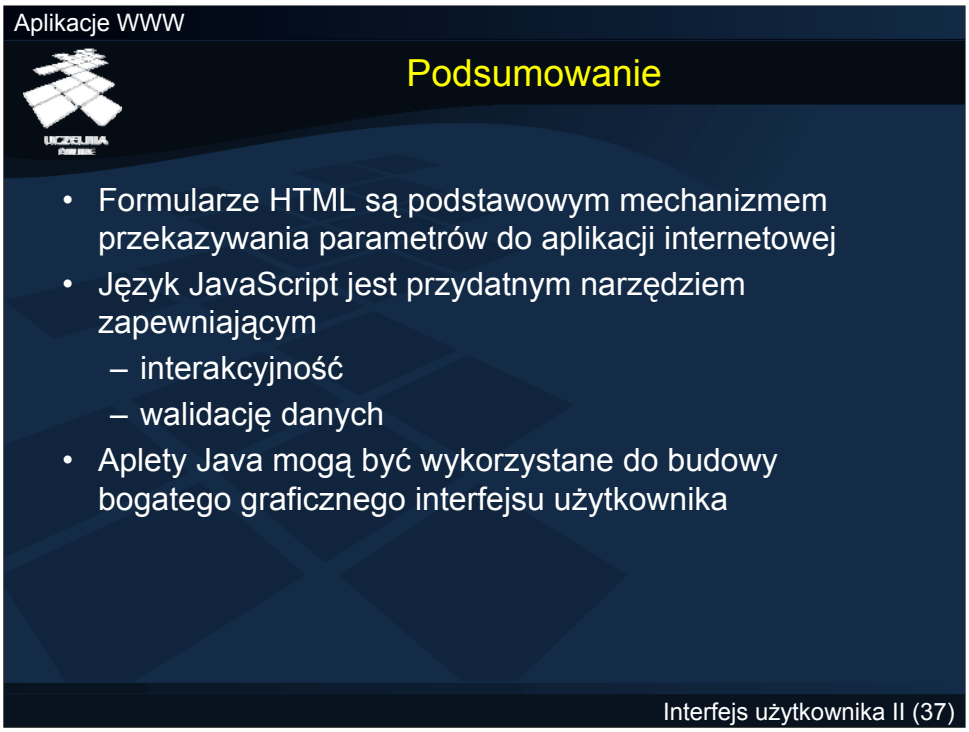
•paint(): uruchamiana wielokrotnie w trakcie życia apletu, powoduje odświeżenie apletu



Aplet można osadzić w dokumencie HTML na trzy sposoby. Najprostszym sposobem jest wykorzystanie znacznika <applet>, w którym należy podać nazwę pliku z pseudokodem (atrybut code) oraz rozmiar okna, w którym będzie wyświetlany aplet (atrybuty width i height). Do apletu można przekazać parametry wywołania za pomocą znacznika <param>, każdy parametr posiada nazwę i wartość, parametry są dostępne wewnątrz apletu przez wywołanie metody getParameter(). Mimo, że znacznik <applet> jest powszechnie rozpoznawany i poprawnie interpretowany przez większość przeglądarek, nie jest on częścią standardu języka HTML. Zgodnie z rekomendacją W3C do osadzania apletu w dokumencie HTML powinien posłużyć znacznik <object> lub <embed>. Znacznik <object> jest uniwersalnym znacznikiem umożliwiającym osadzenie dowolnej zawartości w dokumencie i może posłużyć do włączenia do dokumentu formantów ActiveX, plików multimedialnych, apletów, itd. Atrybut classid zawiera identyfikator wtyczki (ang. plug-in) obsługującej zawartość przekazaną przez parametr. Znacznik <object> jest aktualnie obsługiwany tylko przez część przeglądarek (przede wszystkim przez Internet Explorer), stąd konieczność stosowania alternatywnego znacznika <embed> (interpretowany przez przeglądarki z rodziny Mozilla).



Aplety Java mogą być wykorzystane do budowania interfejsu użytkownika oferującego dużo więcej opcji niż tradycyjne formularze HTML. Do budowania złożonego interfejsu użytkownika można wykorzystać bibliotekę AWT (Abstract Windowing Toolkit) zawierająca podstawowe kontrolki, takie jak: pola tekstowe, listy rozwijane, pola wyboru, przyciski radiowe, przyciski, panele czy pola typu memo. Dużo więcej możliwości oferują biblioteki Swing i SWT. Są to rozbudowane zbiory kontrolek (ang. widgets - window gadgets) umożliwiające tworzenie dowolnie złożonych graficznych interfejsów użytkownika. Zawierają, między innymi, wskaźniki postępu, kontrolki kalendarza i zegara, macierze elementów, panele, suwaki, liczniki, i wiele innych.



Podstawową metodą przekazywania parametrów do aplikacji internetowej są formularze HTML. Oferują podstawowy zbiór kontrolek, takich jak pola tekstowe, listy rozwijane, pola wyboru czy przyciski radiowe. Dane wprowadzane przez użytkowników muszą być starannie walidowane. Może do tego posłużyć język JavaScript. Jest to lekki język skryptowy powszechnie implementowany w wielu klientach protokołu HTTP, w tym w szczególności w przeglądarkach. JavaScript posiada zaawansowane mechanizmy obiektowe, pozwalające na budowanie złożonych aplikacji internetowych. Jedną z metod budowania bogatego graficznego interfejsu użytkownika jest wykorzystanie apletów języka Java.