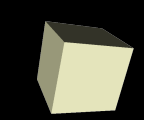
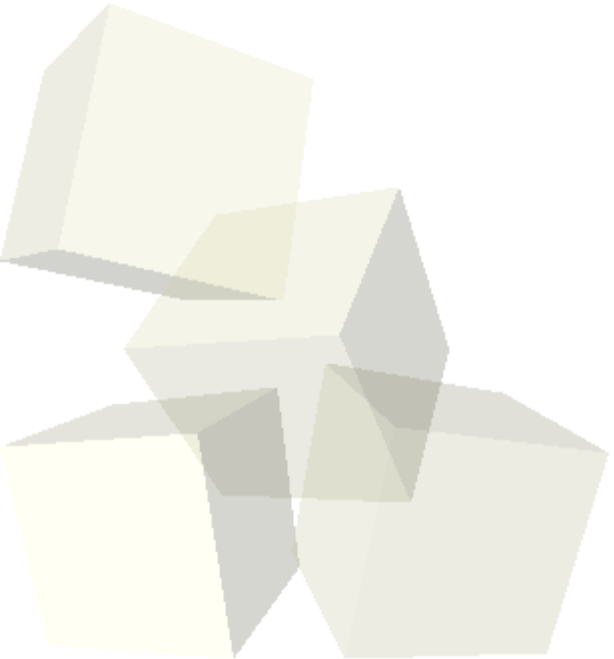


JavaScript



Wstęp

JavaScript powstał w 1995. Opracowany przez Netscape oraz Sun Microsystem. Jest obiektowym skryptowym językiem programowania.

Cel: Wprowadzanie elementów interaktywnych na stronach internetowych.

Dialekty: Adobe Flash, ActionScript, JSrcipt (Microsoft).

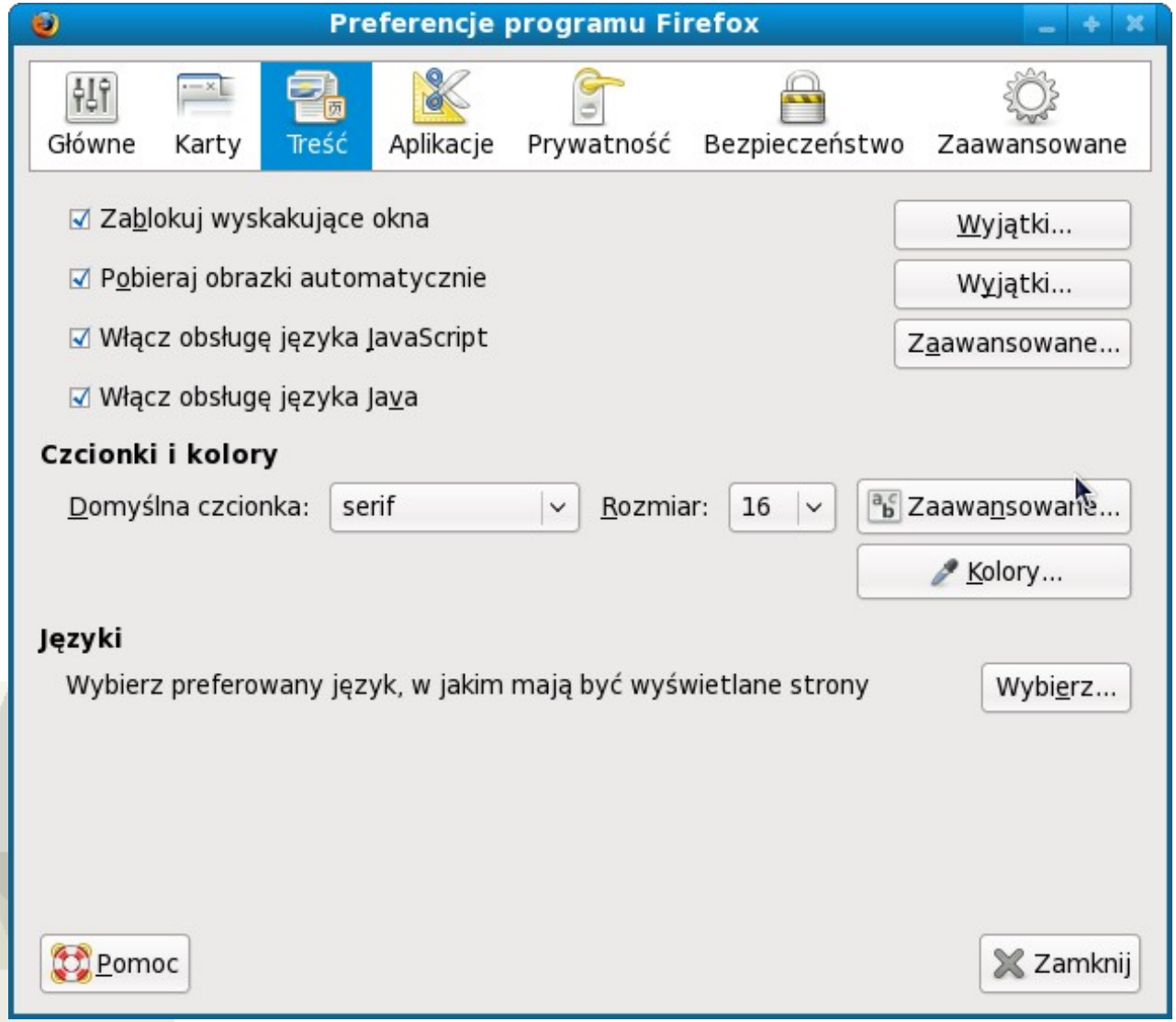
Zastosowanie

Do wykonywania operacji po stronie klienta (przeglądarki):

- Zapewnienie interakcji z użytkownikiem
- Tworzenie komunikatów, okienek dialogowych
- Tworzenie dynamicznych formularzy, w których możemy np. sprawdzić poprawność danych
- Gry interaktywne
- Wykonywanie skomplikowanych obliczeń matematycznych
- Tworzeniu efektów (ruchomy tekst, animacje, wykorzystanie dźwięku)
- Dynamiczne modyfikacje wyglądu strony (zmiana tła, tekstu, ...)
- Tworzenie stron html w zależności od potrzeb użytkownika
- Komunikacja pomiędzy elementami strony (np. Ramki)
- i wiele, wiele innych

Włączanie obsługi JavaScript

- Przeglądarka Firefox 3.0



Model obiektów dokumentu

DOM – Document Object Model

DPM to sposób reprezentacji złożonych dokumentów XML i HTML w postaci modelu obiektowego. Model ten jest niezależny od platformy i języka programowania.

Standard DOM jest definiowany przez W3C.

Poziomy DOM

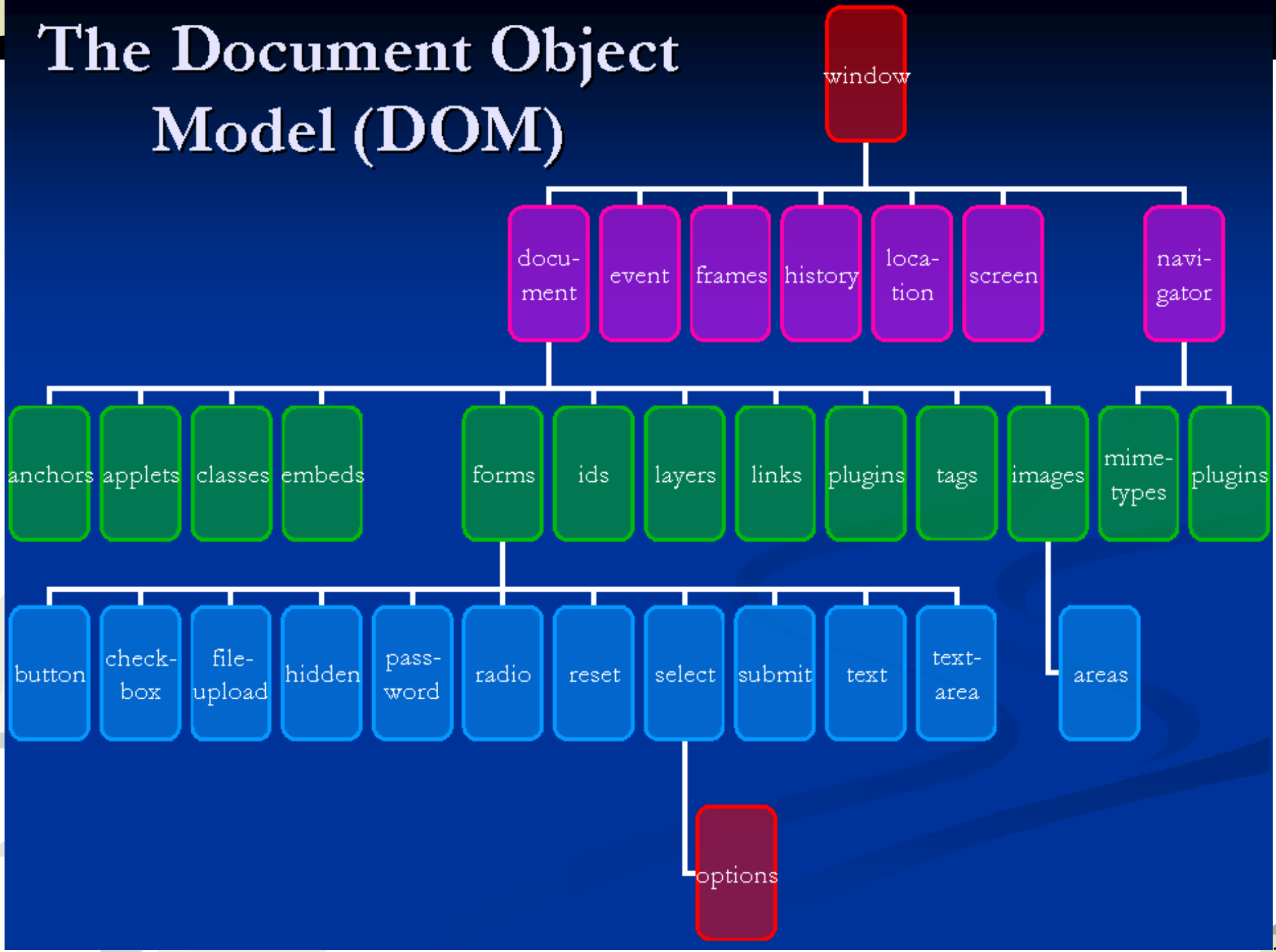
- ***DOM Level 0*** – nieoficjalny; model DOM z przeglądarki Netscape Navigator 3.0, skopiowany przez Microsoft i zaimplementowany we wszystkich przeglądarkach internetowych, mimo że nie stanowi oficjalnego standardu W3C. Zapewnia prosty dostęp głównie do elementów formularzy i obrazków.

- ***DOM Level 1*** – dostępny z poziomu JavaScriptu w przeglądarkach internetowych oraz w wielu innych językach programowania. Poziom ten odpowiada za dostęp do treści dokumentu poprzez tworzenie, modyfikowanie i dołączanie węzłów i atrybutów.

- ***DOM Level 2*** – dostępny w większości współczesnych przeglądarek internetowych oraz w wielu językach programowania. Poziom ten odpowiada m.in. za obsługę zdarzeń i przestrzeni nazw.

- ***DOM Level 3*** - składa się na niego sześć specyfikacji:
  - DOM Level 3 Core
  - DOM Level 3 Load and Save
  - DOM Level 3 XPath
  - DOM Level 3 Views and Formatting
  - DOM Level 3 Requirements
  - DOM Level 3 Validation

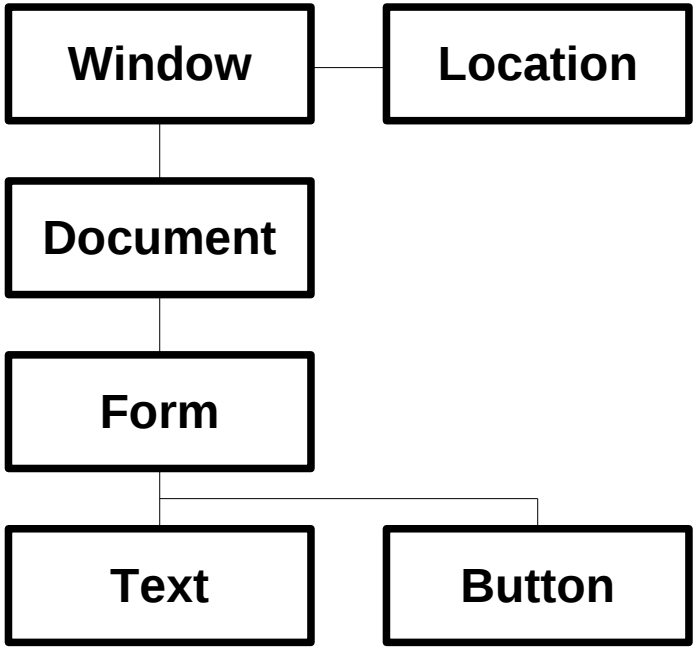
The Document Object Model (DOM)



Źróðło: http://dsmith77.files.wordpress.com/2008/07/the-document-object-model-dom.gif

Formularz a DOM

```
<HTML>
 <BODY>
  <H1> Przykład formularza </H1>
  <FORM>
   <INPUT TYPE=”text”>
   <INPUT TYPE=”button”>
  </FORM>
 </BODY>
</HTML>
```

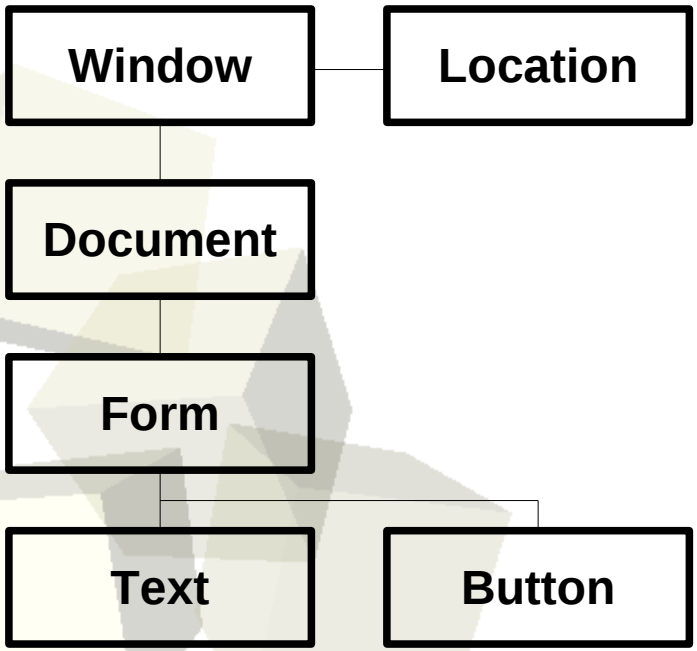


Każdy obiekt “Window” zawiera obiekt “Location” przechowujący informacje o adresie URL odczytanego dokumentu.

Odwołania do obiektów

Odwołania do obiektów realizowane za pomocą nazw, czyli zmiennej NAME.

```
<HTML>
 <BODY>
  <H1> Przykład formularza </H1>
  <FORM NAME=”pizza”>
   <INPUT TYPE=”text” NAME=”rodzaj”>
   <INPUT TYPE=”button” NAME=”sprawdz”>
  </FORM>
 </BODY>
</HTML>
```



window

window.document

window.document.pizza

window.document.pizza.rodzaj
window.document.pizza.sprawdz

Odwołania do obiektów

W DOM może występować tylko jeden obiekt *window* oraz tylko jeden *document*.

Przy odwołaniach do obiektów można pomijać *window*, ale nie można pominąć obiektu *document*.

```
window.document.pizza.rodzaj
```

OK

```
document.pizza.rodzaj
```

OK

```
pizza.rodzaj
```

BŁĄD !!!

Schemat skryptu JavaScript

```
<SCRIPT LANGUAGE="JavaScript">
 <!-- a w środku instrukcje -->
 Instrukcja 1
 Instrukcja 2
 Instrukcja 3
......
 </SCRIPT>
```

gdy kod skryptu jest umieszczony w oddzielnym pliku

```
<SCRIPT LANGUAGE="JavaScript" SRC="skrypt.txt">
```

JavaScript w HTML-u

- W sekcji ***HEAD***
  - Jeśli skrypt będzie uruchamiany w reakcji na jakieś zdarzenia, np. naciśnięcie przycisku
- W sekcji ***BODY***
  - Jeśli skrypt ma być uruchamiany w czasie otwierania strony, tak aby wygenerować jej zawartość.

Liczba skryptów w obydwu sekcjach jest nieograniczona.

OnLoad

Uruchamianie skryptów podczas otwierania dokumentów:

- Umieszczenie skryptu w sekcji **BODY**.
- Obsługa zdarzenia OnLoad – zdarzenia zachodzącego w chwili, gdy przeglądarka zakończy pobieranie wszystkich elementów strony (w tym rysunków, apletów, osadzonych plików multimedialnych)

```
<HTML>
 <HEAD>
  <SCRIPT LANGUAGE="JavaScript">
   <!-- function done() {
      alert ("Zakończyło się ładowanie strony") }
  <SCRIPT>
 </HEAD>
 <BODY onLoad="done()">
      TRESC STRONY
 </BODY>
</HTML>
```

Komentarze

<!-- jakis komentarz HTML -->

Używany do ukrywania kodu przed starszymi wersjami przeglądarek.

Dodatkowo w Java Script można używać:

// to jest komentarz jednolinijkowy

/* a to jest komentarz
wieloliniowy */

Zmienne w języku JavaScript nie mają ustalonych typów - mogą one przechowywać wartośći dowolnego typu danych.

- Typy zmiennych: liczbowe, łańcuchy znaków, logiczne, null.
- Nazwy zmiennych w przeciwieństwie do wartości znakowych nie zapisuje się w cudzysłowach.
- JS rozróżnia duże i małe litery. Tak więc zmienna Tekst nie będzie traktowana jako zmienna TeKsT.
- Nazwa zmiennych nie może zawierać spacji.
- Nazwa nie może być słowem zarezerwowanych dla JavaScript.
- Nazwa zmiennych powinna zacząć się od litery lub znaku podkreślenia "_".

Typy wartości

Typ	Przykład	Opis
String	“ciag znakow”	Ciąg znaków wewnątrz cudzysłowu
Number	7.53 074 0XAF	Dowolna liczba (bez cudzysłowu) Liczba w postaci ósemkowej Liczba w postaci szesnastkowej
Boolean	true lub false	Wartość logiczna
Null	null	Brak określonej wartości
Object		Wszystkie właściwości i metody należące do dowolnego obiektu lub tablicy
Function		Definicja funkcji

Zmienne

Deklaracja zmiennych – słowo kluczowe ***var.***
Tylko raz, dana zmienna może być zadeklarowana.

```
var zmienna
var zm1=25 var zm2=”hello”
var zm3=34.34
var zm4=zm1+12
```

Konwersje typów danych

```
12+16                       //wynik=28
12+"16"                     //wynik="1216"
12+13+"16"                  //wynik="2516"
```

```
parseInt("12")              //wynik=12
parseInt("12.73")           //wynik=12
parseFloat("12")            //wynik=12
parseFloat("12.73")         //wynik=12.73

12+13+parseInt("16")        //wynik=41
```

Przy zamianie liczb na łańcuchy, wykorzystuje się pusty łańcuch.

```
(""+25)                     //wynik="25"
(""+25).length              //wynik=2
```

Tablice

- Różne wiersze tablicy mogą przechowywać różne typy danych.

```
var kraj = new Array[10]

kraj[0]=”Albania”
kraj[9]=98.765
```

Tablice – sortowanie

```
<html>
<body>

<script type="text/javascript">

var arr = new Array(5);
arr[0] = "Zosia";
arr[1] = "Marysia";
arr[2] = "Ola";
arr[3] = "Zuzia";
arr[4] = "Jadzia";

document.write(arr + "<br />");
document.write(arr.sort());
</script>

</body>
</html>
```

Wynik:

Zosia, Marysia, Ola, Zuzia, Jadzia

Jadzia, Marysia, Ola, Zosia, Zuzia

Tablice – sort. wart. liczbowych

```
<html>
<body>

<script type="text/javascript">

function compareNumbers(a, b) {
  return a - b;
}

var arr = new Array(6);
arr[0] = "10";
arr[1] = "5";
arr[2] = "40";
arr[3] = "25";
arr[4] = "1000";
arr[5] = "1";

document.write(arr + "<br />");
document.write(arr.sort(compareNumbers));

</script>
</body>
</html>
```

Przy sortowaniu wartości liczbowych musi być zdefiniowana funkcja porównująca wartości. W przeciwnym przypadku, wartości liczbowe będą traktowane jako łańcuchy.

Operatory

- Operatory porównania

Operator	Opis
==	Jest równe
!=	Jest różne
>	Jest większe niż
>=	Jest większe bądź równe
<	Jest mniejsze niż
<=	Jest mniejsze bądź równe

- Operatory arytmetyczne
  +, -, *, /
- Operatory logiczne
  ||, &&

Łańcuchy

- Konkatenacja (łączenie łańcuchów)

```
Imie = "Jan";
Nazwisko = "Nowak";
Osoba = Imie " " + Nazwisko;  //wynik "Jan Nowak"

Imie += "Maria";     // wynik "Jan Maria"
```

- Zmiana wielokości liter
  - toUpperCase()
  - to LowerCase()

```
Imie = "Jan";
Imie.toUpperCase();  // "JAN"
Imie.toLowerCase();  // "jan"
```

Łańcuchy

- Przeszukiwanie

```
"traktor".indexOf("rak") // wynik=1
```

- Odczytywanie znaków

```
"traktor".charAt(4) // wynik="t"
```

- Odczytywanie podłańcuchów

```
"traktor".substring(1,5) // wynik="rakt"
```

Obiekt Math

- Stałe

```
Math.E              //liczba e
Math.PI             //liczba pi
Math.SQRT2          //pierwiastek kwadratowy z 2
Math.SQRT1_2        //pierwiastek kwadratowy z 1/2
Math.LN2            //logarytm naturalny z 2
Math.LN10
Math.LOG2E
Math.LOG10E
```

Obiekt Math

- Funkcje

```
Math.min(val1,val2)  //zwraca wartość mniejszą
Math.max(val1,val2)  //zwraca wartość większą
Math.pow(val1,val2)  //val1 do potęgi val2
Math.sqrt(val)       //pierwiastek z val
Math.exp(val)        //e do potęgi val
Math.round(val)      //zaokrąglenie do liczby całkowitej
Math.random()        //wartość losowa z przedziału [0,1]
Math.sin(val)        //sinus
Math.cos(val)
Math.tan(val)
Math.asin(val)
Math.acos(val)
Math.atan(val)
Math.log(val)        //logarytm naturalny z val
Math.floor(val)      //zaokrąglenie do l. całk. w dół
Math.ceil(val)       //zaokrąglenie do l. całk. w górę
```

Obiekt Date

- Konstruktory

```
var data = new Date (“Month dd, yyyy hh:mm:ss”);
var data = new Date (“Month dd, yyyy”);
var data = new Date (“yy,mm,dd,hh,mm,ss”)
var data = new Date (“yy,mm,dd”)
```

- Funkcje

```
getTime() - liczba milisekund od godz. 00:00:00 czasu
            Greenwich 01-01-1970
getYear() - rok
getMonth() - miesiąc ( styczeń=0 )
getDate() - dzień miesiąca
getDay() - dzień tygodnia (niedziela = 0 )
getHours() - godzina w zapisie 24 godzinnym
getMinutes() - minuta
getSeconds() - sekunda

oraz odpowieniki setXXXXX(val)
```

Przykład - zegar

```
<html><head>
<script type="text/javascript">

function startTime(){
 var today=new Date();
 var h=today.getHours();
 var m=today.getMinutes();
 var s=today.getSeconds();

 // add a zero in front of numbers<10
 m=checkTime(m);
 s=checkTime(s);
 document.getElementById('txt').innerHTML=h+":"+m+":"+s;
 t=setTimeout('startTime()',500);
}

function checkTime(i){
 if (i<10)  {
   i="0" + i;
 }
 return i;
}
</script>
</head>

<body onload="startTime()">
<div id="txt"></div>
</body>
</html>
```

Przykład – porównywanie dat

```
var myDate=new Date();
myDate.setFullYear(2010,0,14);

var today = new Date();

if (myDate>today)
{
  alert("Dzisiaj jest przed 14-tym styczniu 2010");
}
else
{
  alert("Dzisiaj jest po 14-tym styczniu 2010");
}
```

Typ zmiennej(obiektu)

- Funkcja `constructor`

```
<script type="text/javascript"&gt>
  var test=new Boolean();
  if (test.constructor==Array) {
    document.write("This is an Array");
  }
  if (test.constructor==Boolean) {
    document.write("This is a Boolean");
  }
  if (test.constructor==Date) {
    document.write("This is a Date");
  }
  if (test.constructor==String) {
    document.write("This is a String");
  }
</script>
```

Wynik: This is a Boolean

Struktury sterujące

- Instrukcja ***if...else***

```
if (warunek1){
  instrukcje1;
} else if (warunek2){
  instrukcje2;
} else if (warunek3){
..instrukcje3;
}
...
else if (warunekn){
..instrukcjen;
}
else{
..instrukcjem;
}
```

- Instrukcja ***switch***

```
switch(wartość){
case 1:
  instrukcje1;
  [break;]
case 2:
  instrukcje2;
  [break;]
default:
  instrukcje_def;
  [break;]
}
```

- Pętla ***for***

```
for([wyrażenie_początkowe]; [warunek]; [zmiana_wyrażenia]) {
  instrukcja1
  ........
  instrukcjan
}
```

- Pętla ***while***

```
while (wartość<=wartość_końcowa)
{
  instrukcja1
  ........
  instrukcjan
}
```

W pętlach można stosować instrukcje ***break*** oraz ***continue***.

Funkcje

```
function nazwaFunkcji ([par1]...[,par[N]) {
  instrukcja1
  ........
  instrukcjan
}
```

```
function mojaFunckcja(txt,abc)
{
  alert(txt);
  alert(abc);
}
```

OnLoad – zachodzi w chwili, gdy przeglądarka zakończy pobieranie wszystkich elementów strony (w tym rysunków, apletów, osadzonych plików multimedialnych)

OnUnload – przed opuszczeniem strony

```
<BODY onLoad=”fukcja_obsługująca_onload()”>

<BODY onUnload=”fukcja_obsługująca_onunload()”>

<BODY onLoad=”fukcja_obsługująca_onload()”
onUnload=”fukcja_obsługująca_onunload()”>

<BODY onLoad=”self.name='Main'” onUnload=”self.name=''”>
```

Zdarzenia zwiazane z walidacją wartości.

OnFocus – przejmowanie fokusu
OnBlur – tracenie fokusu
OnChange – zmiana zawartości

```
<input type="text" size="30"
id="email" onchange="checkEmail()">
```

Zdarzenia zwiazane z ruchem myszką.
Zastosowanie: np. do tworzenia przycisków animowanych.

OnMouseOver – gdy kursor kyszki znajduje na obiekcie
OnMouseOut – gdy kursor myszki

Tworzenie okna

- Tworzenie `window.open(url, nazwa, wyglad)`

```
<script language="JavaScript">
function newwin()
{
var subwin = window.open("../Formularze/Formularz1.html",
"def", "HEIGHT=200,WIDTH=200")
}
</script>
```

- Zamykanie `window.close()`

Okna dialogowe

- Metoda `window.alert(tekst)` – okno z ostrzeżeniem

```
alert('Wartość musi być większa od 0');
```

Okna dialogowe

- Metoda `window.confirm(tekst)` – okno z zapytaniem

```
if ( confirm(“Czy chcesz rozpocząć grę od nowa?”) )
{
  alert(“Nowa gra.”);
}
```

Właściwości obiektu `window`

- Status – `window.status`

```
<a href="http://wi.pb.edu.pl"
OnMouseOver="window.status='Strona Wydziału Informatyki
PB'; return true">Wydział Informatyki PB</a>
```

Uwaga: Niektóre funcje obsługi zdarzeń, np. OnMouseOver, wymagają instrukcji "return true".

- Domyślny status – `window.defaultStatus`

- `window.closed` – ***true*** gdy zamknięte

Uwaga: Odwołanie do zamkniętego okna zawsze zwraca ***null***.

- `window.document` – odwołanie do obiektu ***document***
- `window.location` – odwołanie do obiektu ***location***
- `window.history` – odwołanie do obiektu ***history***

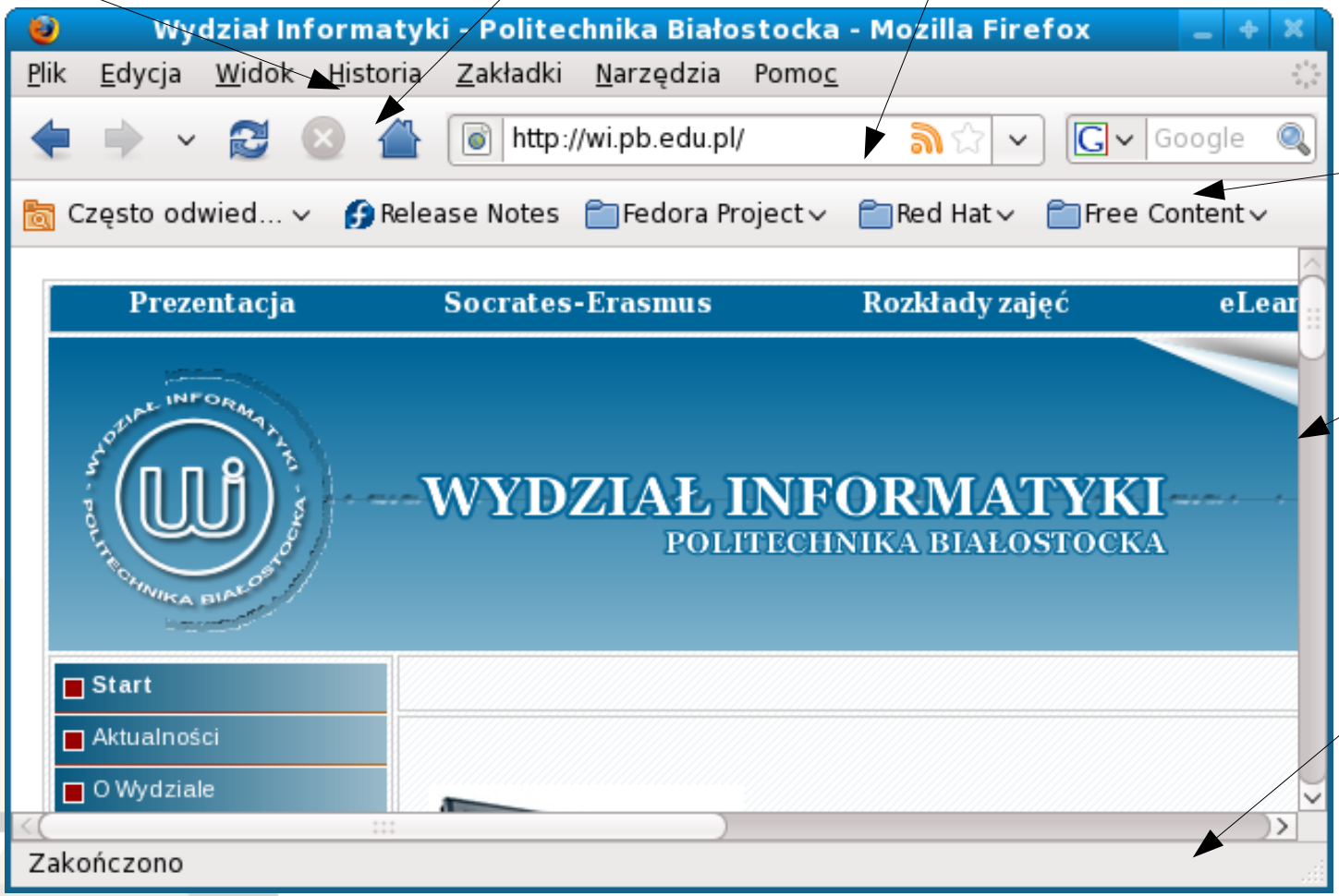
Właściwości obiektu `window`

- `window.innerHeight` – wysokość okna wewnetrznego, czyli obszaru okna przeglądarki wykorzystywanego do wyświetlania dokumentu
- `window.innerWidth` – szerokość okna wewnętrznego

- `window.outerHeight` – wysokość okna zewnetrznego, czyli rozmiaru pełnego okna przeglądarki, łącznie ze wszystkimi elementami wykończenia: paski przewijania, paski stanu, menu, itd.
- `window.outerWidth` – szerokość okna zewnętrznego

Właściwości obiektu `window`

Pasek menu (menubar) Pasek zadań (toolbar) Pasek adresu (locationbar)



Pasek osobisty (personalbar)

Pasek przewijania (scrollbar)

Pasek statusu (statusbar)

Wszystkie elementy wykończenia mają właściwość `visible` (wartość boolowska).

Metody obiektu `window`

- `window.alert(wiadomość)` – wyświetlenie alertu
- `window.close()` – zamknięcie okna

- `window.back()` – powrót do poprzedniego okna (programowy odpowienik przycisku nawigacyjnego podobnie jak `forward, home` oraz `print`).
- `window.forward()` – powrót do poprzedniego okna
- `window.home()` – powrót do okna strony domowej
- `window.print()` – drukowanie

- `window.moveBy(deltaX, deltaY)` – przesunięcie okna w stosunku do aktualnego położenia
- `window.moveTy(x, y)` – przesunięcie okna do pewnej pozycji bezwzględnej.

Zdarzenia obiektu `window`

- `window.onLoad()`
- `window.onUnload()`
- `window.onMove()`
- `window.onResize()`
- `window.onDragDrop()` - argumentem może być obiekt typu `event`, zawierający informacje m.in. w którym miejscu upuszczono element, jaki jest adres URL dokumnetu.

Właściwości obiektu `location`

Zawiera informacje na temat adresu URL okna lub ramki.

- `location.protocol` – np. “http:”
- `location.hostname` – np. “aragorn.pb.bialystok.pl”
- `location.port` – np. “80”
- `location.host` – np. “aragorn.pb.bialystok.p:80”
- `location.pathname` – np. “/poczta/odebrane.html”
- `location.hash` – np. “#nowyMail”, odnośnik lokalny w dokumencie
- `location.href` – pełny adres URL,
  np. “http:aragorn.pb.bialystok.pl:80/poczta/odebrane.html#nowyMail”

Metody obiektu `location`

- `location.assign("adresURL")` – przypisanie adresu spowoduje przejście do strony o zadanym adresie. Ten sam efekt można uzyskać przypisując adres zmienenj `location.href`.

- `location.reload(odczytBezwarunkowyBool)` – odświeżanie strony

- `location.replace("adresURL")` – przejście do strony o podanym adresie URL. Różnica w stosunku do `location.assign` polega na tym, że w tym przypadku po otwarciu nowego dokumentu na liście historii nie będzie widoczny dokument, w którym umieszczono instrukcję powodującą nawigację.

Właściwości obiektu `history`

Zawiera informacje o ostatnio odwiedzanych adresach URLi.

- `history.current` – obecny adres
- `history.previous` – poprzedni adres
- `history.next`– następny adres
- `history.length`– liczba elementów w tablicy historii

```
var hist = window.history[5];
```

Metody obiektu `history`

- `history.back()` – przejście do poprzedniego dokumentu zapisanego w historii.
- `history.forward()` – przejście do następnego dokumentu zapisanego w historii.
- `history.go([+|-]krok)` – przejście do dokumentu znajdującego się na liście historii o pewną liczbę kroków (do przodu lub do tyłu).

Obiekt document

- `document.write` – wypisywanie tekstu wewnątrz dokumentu

```
document.write("tekst który pojawi się na stronie");
document.write("<h1>Nagłówek</h1>");
```

- `document.title` – zwraca tytuł dokumentu (określony za pomocą znacznika <title>

```
document.write(document.title);
```

- `document.URL` – zwraca adres URL dokumentu

```
document.write(document.URL);
```

Obiekt document

- `document.referrer` – zwraca adres URL dokumentu, który załadował ten dokument

```
document.write(document.referrer);
```

- `document.domain` – zwraca domenę dla dokumentu

```
document.write(document.domain);
```

- `document.URL` – zwraca adres URL dokumentu

```
document.write(document.URL);
```

Zmiana linka

Strona Wydzialu Informatyki [Zmien link]

```
function zmienLink() {
  document.getElementById('linkToPage').innerHTML="Strona
Politechniki PB";
 document.getElementById('linkToPage').href="http://www.pb.edu.pl";
 document.getElementById('linkToPage').target="_blank";
}
```

```
<a id="linkToPage" href="http://wi.pb.edu.pl">Strona Wydzialu
Informatyki</a>
<input type="button" onclick="zmienLink()" value="Zmien link">
```

Ustawienie właściwości target na “_blank” spowoduje, że po kliknięciu na link, zostanie on otwarty w nowym oknie.

Strona Politechniki PB [Zmien link]