



Technologie internetowe

JavaScript

Paweł Rajba

pawel@ii.uni.wroc.pl

<http://www.itcourses.eu/>

Spis treści

- Wprowadzenie
- JavaScript i ECMAScript
- Narzędzia, osadzanie JavaScript
- Podstawy języka
 - Komentarze, typy i konwersje typów
 - Zmienne i stałe, Operatory, instrukcje sterujące, funkcje
 - Właściwości i funkcje predefiniowane
 - Przegląd obiektów:
Array, Boolean, Date, Math, String, RegExp
- DOM
 - Window, Location, History, Document, Image, Link
- Formularze
- Zdarzenia, kompatybilność, wydajność

Wprowadzenie

- Netscape 2.0 w 1995
 - Brendan Eich
- Język skryptowy wieloparadygmatowy
 - obiektowy, funkcyjny, imperatywny
- Implementacje
 - Najpopularniejsze w przeglądarkach
 - Coraz popularniejsze po stronie serwera (node.js)
 - Wiele innych mniej znanych
- Drugie życie dzięki AJAX i Web 2.0

JavaScript i ECMAScript

- Standaryzacja – organizacja ECMA:
<http://www.ecma-international.org/>
- Współpraca Netscape z ECMA w celu standaryzacji JavaScript (1996)
 - ECMAScript – język
 - JavaScript, ActionScript, JScript – dialekty
- Obecnie najnowsza jest edycja 6 specyfikacji:
 - <http://www.ecma-international.org/publications/standards/Ecma-262.htm>

JavaScript i ECMAScript

The screenshot shows the ECMA International website's 'Standards' section. The page features a navigation menu with 'Standards' selected, a sidebar with a 'SITE MAP' containing links to 'Standards Index', 'Standards List', 'Withdrawn Standards', 'Tech. Reports Index', 'Tech. Reports List', 'Withdrawn Tech. Reports', and 'Mementos'. The main content area displays the title 'Standard ECMA-262' in large orange font, followed by 'ECMAScript® 2015 Language Specification' in a smaller orange font. Below this, it states '6th edition (June 2015)'. A paragraph explains that this standard defines the ECMAScript 2015 general purpose programming language. A section titled 'The following files can be freely downloaded:' contains a table with two rows: 'ECMA-262.pdf' (12 501 480 bytes, Acrobat (r) PDF file) and 'ECMA-262 edition 6' (Browsable HTML). The page also includes a 'Printer Friendly Version' link and 'Back' navigation buttons.

ecma INTERNATIONAL **Standards** [Contact Ecma](#)
Rue du Rhône 114 CH-1204 Geneva
T: +41 22 849 6000 F: +41 22 849 6001

[SITE MAP](#) [What is Ecma](#) [Activities](#) [News](#) **[Standards](#)**

[Standards Index](#)
[Standards List](#)
[Withdrawn Standards](#)
[Tech. Reports Index](#)
[Tech. Reports List](#)
[Withdrawn Tech. Reports](#)
[Mementos](#)

[Printer Friendly Version](#)
[Back](#)

Standard ECMA-262

ECMAScript® 2015 Language Specification

6th edition (June 2015)

This Standard defines the ECMAScript 2015 general purpose programming language.

The following files can be freely downloaded:

File name	Size (Bytes)	Content
ECMA-262.pdf	12 501 480	Acrobat (r) PDF file
ECMA-262 edition 6		Browsable HTML

This 6th edition will be also approved as fourth edition of the international standard ISO/IEC 16262:2016.

This 6th edition has been prepared under the Ecma RF patent policy.

The previous replaced "historical" editions of this Ecma Standard are available [here](#).

[Back](#)

Narzędzia

- Konsola Google Chrome
- JSFiddle, <https://jsfiddle.net/>
- JSBin, <http://jsbin.com/>
- Wklejamy

- HTML

```
<div id="test"></div>  
<button id="button">BUTTON</button>
```

- CSS

```
#test { font-family: Arial,sans-serif; }
```

- JS

```
document.getElementById('button').onclick =  
  function() {  
    document.getElementById('test').innerHTML = '<b>ALA MA KOTA</b>';  
  }
```

Osadzanie JavaScript

- Wewnątrz dokumentu HTML
 - ```
<script language="javascript" type="text/javascript">
<!--
...tutaj umieszczamy skrypt...
//-->
</script>
```
- Dołączenie zewnętrznego pliku
  - ```
<script language="javascript" type="text/javascript"  
src="skrypt.js"></script>
```
- Wewnątrz znaczników:
 - ```
Tu nie klikać
```

# Osadzanie JavaScript

- Gdzie umieszczać?
  - W nagłówku
  - W treści dokumentu (początek, koniec)
  - Podpięcie pod zdarzenia
- Unobtrusive JavaScript
  - Analogia do CSS



# Podstawy języka

- Zanim zaczniemy, mała przestroga...

Wat

@garybernhardt

# Podstawy języka

- Komentarze
  - Tak jak w innych językach podobnych do języka C
    - // komentarz jednowierszowy
    - /\* komentarz blokowy \*/

# Podstawy języka

- Literały
  - Liczby całkowite, liczby zmiennoprzecinkowe
  - Logiczne (true, false)
  - Napisy
    - W apostrofach lub cudzysłowach
    - Znaki specjalne (np. \n, \t, \", \\)
  - null i undefined
  - Tablice
    - `auta = ["Audi", "Ford", "Fiat"]`
    - `kolory = [ , "niebieski", , "zielony", ]`

# Typy i konwersja typów

- JavaScript jest językiem typowanym dynamicznie
- Nie deklarujemy typów, a w razie potrzeby dokonywane są odpowiednie konwersje
- Przykład

```
- var zmienna = 69;
- zmienna = "nowa wartość" // nie będzie błędu
- x = "x = "+40; // zwraca "x = 40"
- y = "69"-9; // zwraca 60
- z = "69"+9; // zwraca 699
```

# Zmienne i stałe

- Deklaracje zmiennych
  - przez przypisanie wartości: `x=5;`
  - przez słowo `var` (lokalnie)
  - jeśli zmiennej nie zostanie przypisana wartość to przyjmuje wartość `undefined`
- Deklaracja stałych
  - stała nie może zmieniać wartości lub być przedeklarowana
  - Przykład:
    - `const wroclaw = "071";`

# Operatory

- Przypisania: =, +=, -=, /=, %=  
`x = 7; x += 4; x %= 10;`
- Porównania: ==, ===, !==, !=, <=, <, >, >=  
`4=='4'; 3=== '3'; 3!== "3"; 3<10;`
- Arytmetyczne: ++, --, %  
`x++; --x; x%4;`
- Bitowe: &, |, ^, ~, <<, >>, >>>  
`15 & 9 // 9; 15 ^ 9 // 6; 9 << 2 // 36;`
- Logiczne: &&, ||, !  
`true && false; !false`

# Operatory

- Operator łączenia napisów: +  
`"Paweł "+"Rajba"=="Paweł Rajba"`
- Operator warunkowy: ?:  
`status = (wiek>=18) ? "pełnoletni" : "dziecko";`
- Operator przecinek - stosowany głównie w for  
`for (var i=0, j=9; i<=9; i++, j--) { ... }`
- Operator in – sprawdza, czy obiekt ma szukaną własność  
`auta=new Array("Volvo", "Audi", "Mercedes");  
0 in auta; 4 in auta; PI in Math;`
- Operator void – wymusza obliczenie wyrażenia bez zwracania wartości  
`<a href="javascript:void(0)">Click</A>`

# Instrukcje sterujące

- Instrukcje if i switch

```
- if (warunek) { ... } else { ... }
- switch (expr) {
 case etykieta :
 // instrukcje...
 break
 ...
 default :
 // instrukcje...
}
```



# Instrukcje sterujące

- Pętle

- `for (var i=0; i<10; ++i) { ... }`
- `do { ... } while (warunek)`
- `while (warunek) { ... }`
- `for .. in`
  - `tablica = [ , "Ala", "Basia", "Małgosia" ] ;`  
`delete tablica[2];`  
`for ( zm in tablica ) {`  
`alert( tablica[zm] );`  
`}`

- Instrukcje `break` i `continue`

# Instrukcje sterujące

- Instrukcja with

```
- var a, x, y;
 var r=10
 with (Math) {
 a = PI * r * r;
 x = r * cos(PI);
 y = r * sin(PI/2);
 }
```

# Funkcje

- Każda funkcja jest obiektem
- Parametry
  - Dodatkowe parametry są ignorowane
    - `add(a,b) { ... }`      `add(1,2,3)`       $\rightarrow 3$  (OK!)
  - Dla brakujących przypisywane jest `undefined`
  - Obiekty przekazywane przez referencję, typy proste przez wartość
- Nie ma przeciążania funkcji

# Funkcje

- `function square(x) { return x*x; }`
  - `square(2) → 4`
- `var squareF = function (x) { return x*x; };`
  - `squareF(2) → 4`
- `(function(x) {return x*x;})(argument);`
- `function map(f,a) {  
 var result=new Array;  
 for (var i = 0; i != a.length; i++)  
 result[i] = f(a[i]);  
 return result;  
}`
- **Wywołanie**  
`map(function(x) {return x * x * x},  
 [0, 1, 2, 5, 10];  
 // zwróci [0, 1, 8, 125, 1000].`

# Funkcje

- Zmienna arguments
  - Reprezentuje argumenty, typu object
  - Jest trochę niezależna od listy argumentów

```
function add() { document.write(arguments.length); }
add(2,3);
```
- Jest wspierana rekurencja

# Funkcje

- Możliwe jest zagnieżdżanie funkcji
- Closure: wewnętrzna funkcja ma dostęp do parametrów funkcji zewnętrznej
- Do odpalenia w JSBin

```
var a = "1";
var outer = function() {
 var b = "2";
 var inner = function()
 {
 document.write(a+" "+b);
 }
 inner();
}
outer();
```

```
$.get('ajax/test.html', function (data) {
 $(resultSelector).html(data);
 alert('Load was performed.');
```

# Właściwości predefiniowane

- Infinity – stała reprezentująca nieskończoność
  - Infinity jest większa od każdej liczby
  - -Infinity jest mniejsza od każdej liczby
  - Infinity zachowuje się w operacjach matematycznych podobnie do nieskończoności
- `var wartosc = Infinity;`
  - `alert(isFinite(wartosc));`
  - `alert(isFinite(23444));`
- NaN – nie-liczba

# Funkcje predefiniowane

- `eval(wyrażenie)` – oblicza wyrażenie lub wykonuje instrukcje
  - nie stosujemy do wyrażeń arytmetycznych – są obliczane automatycznie
  - można wykorzystać do odraczania obliczeń
    - `eval("2+2")`
    - `eval(new String("3+5"))`
    - ```
var str = "if (x == 5)
  { alert('z is 42'); z = 42;} else z = 0; ";
document.write("z is ", eval(str))
```


Funkcje predefiniowane

- `isFinite(liczba)`
- `parseFloat(napis)`
- `parseInt(napis)`
- `isNaN(napis)`

Przegląd obiektów

- Array
- Boolean
- Date
- Math
- String
- RegExp

Array

- Tworzenie tablic:
 - konstruktor Array
 - `t = new Array()`
 - `t = new Array(10)`
 - `t = new Array("A", " B", "C")`
 - literał
 - `t = ["Zebra", "Ryjówka", "Tygrys"]`
- Właściwości
 - length – liczba elementów w tablicy

Array

- Metody

- concat(tab2, tab3, ... tabN)

- łączy tablice w jedną

- przykład

```
num1=[1,2,3]
```

```
num2=[4,5,6]
```

```
num3=[7,8,9]
```

```
numy=num1.concat(num2,num3)
```

```
// tworzy tablicę [1,2,3,4,5,6,7,8,9]
```

- join(separator)

- zamienia tablicę w napis
 - domyślnie separatorem jest ","

Array

- Metody
 - pop()
 - usuwa i zwraca ostatni element tablicy
 - push(elem1, ..., elemN)
 - wstawia elementy na koniec tablicy i zwraca nowy rozmiar
 - reverse()
 - odwraca kolejność elementów
 - shift()
 - usuwa i zwraca pierwszy element tablicy

Array

- Metody
 - slice(begin[,end])
 - zwraca fragment tablicy
 - splice(index, ile, [elem1][,....,elemN])
 - usuwa ile elementów zaczynając od index i w ich miejsce wstawia elem1 do elemN
 - toString()
 - zamienia tablicę na napis (robi join())
 - unshift(element1, ..., elementN)
 - wstawia elementy na początek tablicy

Array

- Metody

- `sort([funkcja_porównująca])`

- sortuje tablice

- funkcja porównująca

- jeśli `funkcja_porównująca(a, b)` zwraca <0 , $b < a$

- jeśli `funkcja_porównująca(a, b)` zwraca >0 , $a < b$

- jeśli `funkcja_porównująca(a, b)` zwraca $=0$, $a = b$

- Przykład

```
function cmp(a,b) {  
    if (a<b) return -1;  
    if (a>b) return 1;  
    return 0;  
}  
liczby = new Array(20,3,12,200)  
liczby.sort(cmp) // [3,12,20,200]
```

Boolean

- Konstruktor
 - Boolean(wartość)
 - Nie mylić literałów true i false z obiektami
 - Przykład

```
x = new Boolean(false);  
if(x) // warunek jest prawdziwy  
x = false;  
if(x) // warunek jest fałszywy
```
- Metody
 - toString()
 - zwraca wartość przechowaną w obiekcie

Date

- Konstruktor
 - `new Date()`
 - `new Date(milliseconds)`
 - `new Date(dateString)`
 - `new Date(yr_num, mo_num, day_num
[, hr_num, min_num, sec_num, ms_num])`
- Uwagi:
 - numer roku powinien być 4 cyfrowy
 - miesiące: 0=styczeń – 11=grudzień
 - dni tygodnia: 0=niedziela – 6=sobota
 - milisekundy podajemy od 1.1.1970, 00:00:00

Date

- Metody pobierające czas lokalny
 - `Date.getDate()` – zwraca dzień
 - `Date.getMonth()` – zwraca miesiąc
 - `Date.getFullYear()` – zwraca rok
 - `Date.getFullYear()` – zwraca rok 4 cyfrowy
 - `Date.getDay()` – zwraca dzień tygodnia
 - `Date.getHours()` – zwraca godzinę
 - `Date.getMinutes()` – zwraca minuty
 - `Date.getSeconds()` – zwraca sekundy
 - `Date.getMilliseconds()` – zwraca milisekund

Date

- Metody pobierające czas względem UTC
 - Analogicznie, np. `Date.getUTCFullYear`
- Inne metody
 - `Date.getTime()`
 - zwraca liczbę milisekund od 1.1.1970 00:00:00
 - `Date.parse(data)`
 - zwraca ilość milisekund od 1.1.1970 00:00:00
 - metoda statyczna
 - `Date.getTimezoneOffset()`
 - zwraca różnicę w minutach pomiędzy czasem lokalnym a UTC

Date

- Przykłady

- `d = new Date(04,02,20)`
`alert(d.getFullYear()) // 4`
`alert(d.getFullYear()) // 1904`
`alert(d.getMonth()) // 2`
- `d = new Date(2004,07,14)`
`alert(d.getFullYear()) // 2004`
`alert(d.getFullYear()) // 104`
`alert(Date.parse(2004,07,14) // 1092434400000`
`alert(d.getTime()) // 1092434400000`
- `d = new Date(104,02,20)`
`alert(d.getFullYear()) // -1796`
`alert(d.getFullYear()) // 104`

Date

- Metody ustawiające czas
 - `Date.setDate(day)`
 - `Date.setMonth(month[, day])`
 - `Date.setYear(year[, month[, day]])`
 - `Date.setFullYear(year[, month[, day]])`
 - `Date.setTime(timevalue)`
 - `Date.setSeconds(seconds[, ms])`
 - `Date.setHours(hours[, minutes[, seconds[, ms]]])`
 - `Date.setMinutes(minutes[, seconds[, ms]])`
 - `Date.setMilliseconds(milliseconds)`

Date

- Metody ustawiające czas względem UTC
 - Analogicznie, np. `setUTCDate(dayValue)`
- Metody zamieniające datę na napis
 - `Date.toGMTString()`, `Date.toUTCString()`
 - zwraca datę w odniesieniu do i formacie GMT
 - `Date.toLocaleString()`
 - zwraca datę i czas lokalną w formacie lokalnym
 - `Date.toLocaleDateString()`
 - zwraca lokalną datę w formacie lokalnym
 - `Date.toString()`
 - zamienia na napis

Math

- Stałe
 - E – stała Eulera, liczba e, podstawa log naturalnego ok. 2.718
 - LN2 – logarytm naturalny z 2, ok. 0.693
 - LN10 – logarytm naturalny z 10, ok. 2.302
 - LOG2E – logarytm dwójkowy z E ok. 1.442
 - LOG10E – logarytm dziesiętny z E ok. 0.434
 - PI – liczba pi, ok. 3.14159
 - SQRT1_2 – pierwiastek kwadratowy z $\frac{1}{2}$; to samo co 1 przez pierwiastek kwadratowy z 2, ok. 0.707
 - SQRT2 - pierwiastek kwadratowy z 2, ok. 1.414

Math

- Wybrane metody
 - $\text{abs}(x)$ – wartość bezwzględna z x
 - $\text{acos}(x)$ – arcus cosinus z x
 - $\text{asin}(x)$ – arcus sinus z x
 - $\text{atan}(x)$ – arcus tangens z x
 - $\text{cos}(x)$ – cosinus z x
 - $\text{exp}(x)$ – Math.E do potęgi x
 - $\text{ceil}(x)$ – sufit z x
 - $\text{floor}(x)$ – podłoga z x

Math

- Wybrane metody c.d.
 - $\log(x)$ – logarytm naturalny z x
 - $\max(x,y)$ – większa z x i y
 - $\min(x,y)$ – mniejsza z x i y
 - $\text{pow}(x,y)$ – x do potęgi y
 - $\text{random}()$ – losuje liczbę pomiędzy 0 a 1
 - $\text{round}(x)$ – zaokrąglenie x
 - $\sin(x)$ – sinus z x
 - $\text{sqrt}(x)$ – pierwiastek kwadratowy z x
 - $\tan(x)$ – tangens z x

Math

- Przykład

- `Math.ceil(4.2)`
- `Math.round(4.2)`
- `Math.floor(4.2)`
- `Math.pow(2,10)`
- `function losuj(x)`
 - {
 - `alert(Math.round(Math.random() * (x-1)))` ;
 - }

String

- Właściwości
 - length
 - długość napisu
- Metody
 - `String.charAt(index)`
 - zwraca znak na pozycji `index`, numeracja od 0
 - `String.charCodeAt(index)`
 - zwraca kod znaku na pozycji `index`, numeracja od 0
 - `String.concat(s1, s2, ..., sN)`
 - dokleja do stringa `s1, s2, ..., sN`

String

- Metody

- `String.fromCharCode(k1, k2, ...,kN)`

- zwraca napis złożony ze znaków o kodach k1, k2, ..., kN
 - przykład:

```
String.fromCharCode(65,66,67) // "ABC"
```

- `indexOf(szukany[, od])`

- szuka w Stringu pozycję pierwszego wystąpienia; jeśli podane od, to szuka od pozycji od

```
count = 0; pos = str.indexOf("x");  
while ( pos != -1 ) {  
    count++;  
    pos = str.indexOf("x",pos+1);  
}
```

String

- Metody
 - `String.lastIndexOf(searchValue[, fromIndex])`
 - szuka w Stringu pozycję ostatniego wystąpienia
 - jeśli podane `fromIndex`, to szuka od pozycji `fromIndex`
 - `String.slice(beginSlice[, endSlice])`
 - pobiera i zwraca kawałek napisu
 - jeśli `endSlice` jest >0 to pobierane są znaki o indeksach od `beginSlice` do `endSlice-1`
 - jeśli `endSlice` jest <0 to oznacza ile znaków od końca nie będzie pobieranych
 - » pobieranie zaczynamy do pozycji `beginSlice`

String

- Metody
 - `String.split([separator][, limit])`
 - dzieli napis względem separator i zwraca tablicę o rozmiarze co najwyżej limit
 - jeśli nie zdefiniujemy separatora, to zwracana jest tablica o jednym elemencie zawierającym cały napis
 - `String.substr(start[, długość])`
 - zwraca napis długość znaków pobrany od pozycji start

String

- Metody

- String.substring(k, l)

- zwraca napis zawarty pomiędzy indeksami k i l

- Przykłady

```
napis = "Netscape"; napis.substring(0,3) // "Net"  
napis.substring(3,0) // "Net"  
napis.substring(7,4) // "cap"  
napis.substring(-1,10) // "Netscape"
```

- toLowerCase(), toUpperCase()

RegExp

- Literał
 - re = /wzorzec/flagi
 - re = /ab+c/i
- Konstruktor
 - re = new RegExp("wzorzec"[, "flagi"])
 - re = new RegExp("\\w+")
 - re = \wedge w+/ \wedge

RegExp

- Właściwości
 - global – czy wyszukiwanie jest do pierwszego wystąpienia, czy wyszukiwane są wszystkie dopasowania; flaga g
 - ignoreCase – nie ma różnicy między wielkimi i małymi literami; flaga i
 - multiline – rozpatruje każdy wiersz osobno; flaga m
 - lastIndex – miejsce od którego będzie kolejne wyszukiwanie; ma sens z opcją global
 - source – napis reprezentujący wzorzec

RegExp

- Metody
 - `exec(napis)`
 - sprawdza, czy napis dopasowuje się do wzorca i zwraca tablicę
 - `test(napis)`
 - to samo co `exec`, tylko zwraca `true` lub `false`

RegExp

- Budowa wyrażeń regularnych
 - \ – nadaje stojącemu za nim znakowi specjalne znaczenie
 - ^ – początek wejścia
 - \$ – koniec wejścia
 - * – element poprzedzający musi powtórzyć się 0 lub więcej razy
 - + – element poprzedzający musi powtórzyć się 1 lub więcej razy

RegExp

- Budowa wyrażeń regularnych
 - ? – element poprzedzający musi powtórzyć się 0 lub 1 raz
 - (x) – dodatkowy wzorzec do zapamiętania
 - x(?=y) – dopasuje się do x, pod warunkiem, że po x jest y
 - x(?!y) – dopasuje się do x, pod warunkiem, że po x nie ma y
 - x|y – x lub y

RegExp

- Budowa wyrażeń regularnych
 - **{n}** – dopasuje się do dokładnie n wystąpień poprzedzającego elementu
 - **{n,}** – poprzedzający element musi wystąpić co najmniej n razy
 - **{n,m}** – poprzedzający element musi wystąpić co najmniej n i co najwyżej m razy
 - **[xyz]** – określa zbiór, dopasuje się do jednej z liter w nawiasach
 - **[^xyz]** – określa dopełnienie zbioru, dopasuje się do czegokolwiek co nie jest w nawiasach

RegExp

- Budowa wyrażeń regularnych
 - **\d** – dopasuje się do cyfry ([0-9])
 - **\D** – dopasuje się do czegoś co nie jest cyfrą
 - **\s** – dopasuje się do pojedynczego odstępu, równoważne [\f\n\r\t\u00A0\u2028\u2029]
 - **\S** – dopasuje się do czegoś co nie jest pojedynczym odstępem
 - **\t** – tabulator
 - **\n** – nowy wiersz

RegExp

- Budowa wyrażeń regularnych
 - **\w** – znak alfanumeryczny ([A-Za-z0-9_])
 - **\W** – dopełnienie \w
 - **\xhh**, **\uhhhh** – kody znaków

RegExp

- Przykłady

- `re = new RegExp("ala+", "ig");`
- `re.exec("ala ma kota ala")`
- `re.lastIndex`
- `re.exec("ala ma kota ala")`
- `re.lastIndex`
- `/a{5}/ig.exec("aaaaaaaaa")`
- `/a{5,10}/ig.exec("aaaaaaaaa")`

RegExp

- Przykłady

- `re = /ab*/ig;`
- `t = re.exec("abbadona");`
- `alert(t);`
- `/^Kasia/.exec("Ala\nKasia")`
- `/^Kasia/m.exec("Ala\nKasia")`
- `myRe=/d(b+)(d)/ig;`
`myArray = myRe.exec("cdbBdbbsbz");`
`// ["dbBd", "bB", "d"]`

Document Object Model

- DOM
 - jest to platforma, niezależna od języka pozwalająca programom i skryptom na dynamiczną modyfikację zawartości okna.
- Obiekty przeglądarki
- Zdarzenia

Przegląd obiektów DOM

- Window
- Location
- History
- Document
- Image
- Link
- Formularze

Window

- Podstawowe właściwości
 - name
 - nazwa okna
 - status
 - tekst pasku stanu przeglądarki
 - closed
 - wartość logiczna, czy okno zostało zamknięte
 - parent
 - okno będące bezpośrednim przodkiem (np. ramka zawierająca inną ramkę)

Window

- Podstawowe właściwości
 - self, window
 - odwołanie do siebie samego
 - top
 - okno najwyższego poziomu w strukturze (np. ramka zawierająca wszystkie inne ramki)
 - opener
 - referencja do obiektu Window, który utworzył okno lub null, jeśli okno utworzył użytkownik
 - navigator
 - obiekt przeglądarki

Window

- Podstawowe właściwości
 - document
 - referencja do obiektu Document
 - frames[]
 - tablica obiektów Window, które reprezentują ramki
 - history
 - referencja do obiektu History
 - location
 - referencja do obiektu Location reprezentującego adres URL dokumentu, jego zmiana powoduje załadowanie nowego dokumentu

Window

- Metody
 - alert(komunikat)
 - wyświetla proste okno z komunikatem
 - confirm(komunikat)
 - wyświetla okno z dwoma przyciskami: Ok i Anuluj
 - prompt(komunikat, domyślna_wartość)
 - wyświetla okno w celu pobrania napisu
 - focus(), blur()
 - aktywuje, dezaktywuje klawiaturę w oknie

Window

- Metody
 - `open("okno.html","nazwa","opcje")`
 - otwiera nowe okno
 - opcje: `width=300, height=300, status=no, location=no, menubar=no, resizable=no, scrollbars=no, titlebar=no, toolbar=yes`
 - `close()`
 - zamyka okno
 - `print()`
 - drukuje, to samo co po wciśnięciu przycisku drukuj w przeglądarce

Window

- Metody
 - `moveBy(x,y)`
 - przesuwa okno o x, y
 - `moveTo(x,y)`
 - przesuwa okno do x, y
 - `resizeBy(x,y)`
 - zmienia rozmiar o x,y
 - `resizeTo(x,y)`
 - zmienia rozmiar do x, y
 - `scrollBy(x,y)`
 - przesuwa dokument w oknie o x,y
 - `scrollTo(x,y)`
 - przesuwa dokument w oknie do x,y

Window

- Metody
 - `setTimeout(wyrażenie/funkcja, milisekundy)`
 - odrocza wykonanie funkcji
 - `clearTimeout(TimeoutID)`
 - anuluje odroczenie i funkcja nie będzie wykonana
 - `setInterval(wyrażenie/funkcja, milisekundy)`
 - wykonuje wyrażenie co określoną liczbę milisekund
 - `clearInterval(TimeoutID)`
 - przerywa wykonywanie funkcji

Przykład

- okienko.html



Location

- Właściwości
 - href, protocol, host (z portem), hostname, port, search (zapytanie), pathname (adres zasobu), hash (identyfikator zakładki, razem z #)
- Metody
 - reload([true])
 - odświeża zawartość okienka
 - dodatkowo można wymusić GET bezwarunkowy
 - replace(URL)
 - wymienia adres okna na ten podany jako argument
 - zamieniany adres nie jest dodawany do historii (nie działa „wstecz” przeglądarki)

Location

- Przykłady:

```
- function adres()  
  {  
    var ad = prompt("Podaj nowy adres:");  
    window.location.replace(ad);  
  }  
  
- function zmien()  
  {  
    window.location.href="http://www.onet.pl/";  
  }
```

History

- Właściwości
 - length – liczba elementów w historii
 - current – bieżący adres URL
 - previous – poprzedni adres w historii
 - next – następny adres w historii
- Metody
 - back() – o jeden w tył (wstecz)
 - forward() – o jeden do przodu (naprzód)
 - go(liczba) – względne wybranie pozycji w historii

Zależności między oknami

- Odwołanie `frames[i]`
- Przykłady
 - `parent.frames[0].i`
 - odwołanie do zmiennej `i` w w oknie pierwszej ramki
 - `parent.menu.podswietl(5)`
 - wywołanie funkcji `podswietl` w sąsiedniej ramce `menu`
 - `var x = parent.banner.x`
 - przypisanie do zmiennej lokalnej

Przykład

- ramki.html



Document

- Właściwości
 - `alinkColor`, `linkColor`, `vlinkColor`
 - kolory hiperłączy, odpowiedniki `alink`, `link`, `vlink` w `body`
 - `bgColor`, `fgColor`
 - kolory tła i tekstu, odpowiedniki `bgcolor` i `text` w `body`
 - `anchors[]`
 - tablica zakładek w dokumencie
 - `cookie`
 - do obsługi cookies

Document

- Właściwości
 - forms[]
 - tablica obiektów Form reprezentujących elementy <form> dokumentu
 - images[]
 - tablica obiektów Image reprezentujących elementy dokumentu
 - lastModified
 - data modyfikacji dokumentu
 - links[]
 - tablica obiektów Link reprezentujących łącza hipertekstowe
 - title
 - tekst zawarty w znacznikach <title></title>

Document

- Metody
 - open()
 - otwiera nowy dokument usuwając zawartość
 - close()
 - zamyka, kończy dokument
 - write(s1, ..., sn)
 - dodaje tekst do dokumentu
 - writeln(s1, ..., sn)
 - dodaje tekst do dokumentu i znak nowego wiersza

Przykład

- dokument.html



Image

- Lista obrazków w dokumencie
 - tablica `document.images[]`
- Tworzenie
 - `new Image([szer, [wysokość]])`
 - `im = new Image()`
 - `im.src = 'konewka.gif'`

Image

- Właściwości
 - border – integer, wielkość obramowania
 - complete – bool, sprawdza, czy obrazek jest już załadowany (tyko do odczytu)
 - height, width – wysokość, szerokość obrazka
 - hspace, vspace – margines poziomy, pionowy
 - name – nazwa obrazka
 - src – źródło obrazka
 - lowsrc – źródło obrazka przy urządzeniu o niskiej rozdzielczości

Przykład

- [obrazki.html](#)



Link, anchor

- Lista linków w dokumencie
 - `document.links[]`
 - `documents.anchors[]`
- Właściwości obiektu link
 - `href`, `protocol`, `host`, `port`, `hostname`, `pathname`, `search`, `target`, `text`

Formularze

- Form
 - Właściwości: action=URL, elements[], encoding, length, method, name
 - Metody: reset(), submit()
 - Zdarzenia: onReset, onSubmit
- Button
 - Właściwości: form, name, type="button", value
 - Metody: focus(), blur(), click()
 - Zdarzenia: onFocus, onBlur, onClick, onMouseDown, onMouseUp

Formularze

- **Checkbox**
 - Właściwości: checked, defaultChecked, form, name, type, value
 - Metody: focus(), blur(), click()
 - Zdarzenia: onFocus, onBlur, onClick
- **Radio**
 - Właściwości: checked, defaultChecked, form, name, type, value
 - Metody: focus(), blur(), click()
 - Zdarzenia: onFocus, onBlur, onClick

Formularze

- Text
 - Właściwości: defaultValue, form, name, type, value
 - Metody: blur(), focus(), select()
 - Zdarzenia: onBlur, onChange, onFocus, onSelect
- TextArea
 - Właściwości: defaultValue, form, name, type, value
 - Metody: blur(), focus() select()
 - Zdarzenia: onBlur, onFocus, onChange, onKeyDown, onKeyUp, onKeyPress, onSelect
- Hidden
 - Właściwości: form, name, type, value

Formularze

- **Select**
 - Właściwości: form, length, name, options[], selectedIndex, type select-one | select-multiple
 - Metody: blur(), focus()
 - Zdarzenia: onBlur, onFocus, onChange
- **Option**
 - Tworzenie
 - `new Option([text[, value[, defaultSelected[, selected]]])`
 - Właściwości: defaultSelected, selected, text, value

Przykład

- formularze.html



Event

- Związany z pojawieniem się zdarzenia
- Jego własności zależą od przeglądarki
- Właściwości
 - screenX, screenY
 - pozycja kursora względem ekranu
 - x,y (IE), layerX, layerY (Mozilla)
 - współrzędne względem dokumentu (IE)
 - type
 - np. click, keydown, itp.
 - altKey, shiftKey, ctrlKey

Event

- Właściwości:
 - which
 - przycisk (1 - lewy, 2 - środek, 3 - prawy)
 - keyCode
 - kod znaku z klawiatury (IE)
 - srcElement (IE), target (Mozilla)
 - element dokumentu w którym nastąpiło zdarzenie

Zdarzenia

- onblur – utrata fokusu
- onchange – zmiana zawartości elementu
- onclick – kliknięcie
- ondblclick – podwójne kliknięcie
- onfocus – przy ustawieniu fokusu
- onkeydown – przy wciśnięciu klawisza
- onkeyup – przy zwolnieniu klawisza
- onkeypress – przy naciśnięciu klawisza

Zdarzenia

- onmousedown – przy wciśnięciu przycisku
- onmouseup – przy zwolnieniu przycisku
- onmousemove – przy przesuwaniu wskaźnika
- onmouseover – przy najechaniu wskaźnikiem
- onmouseout – przy zjechaniu wskaźnikiem

Zdarzenia

- onload – przy załadowaniu dokumentu
- onunload – przy zamykaniu dokumentu
- onreset – przy zerowaniu formularza
- onsubmit – przy wysyłaniu danych formularza
- onresize – przy zmianie rozmiaru
- onselect – przy zaznaczeniu tekstu

Zdarzenia

- Funkcje do obsługi zdarzeń tworzymy następująco:

- IE:

```
document.onclick = obsluz_klik;  
function obsluz_klik() { ... }
```

- Mozilla

```
document.onclick = obsluz_klik;  
function obsluz_klik(event) { ... }
```

Przykłady

- `coords.html`
- `cyfry.html`
- `dynamic.html`

Kompatybilność

- Sprawdzamy, czy jest tablica images[]

```
if (document.images) {  
    // kod obsługi wymiany obrazków  
}
```

- Sprawdzamy dostęp do warstw

```
if (document.getElementById) { ...DOM W3C... }  
if (document.all) { ...IE... }  
if (document.layers) { ...Mozilla... }
```

Wydajność przeglądarek

- Kraken Benchmark
 - <http://krakenbenchmark.mozilla.org/>
- JetStream
 - <http://browserbench.org/JetStream/>
- FishIE Tank
 - <http://www.fishgl.com/>
- Test Drive
 - <https://dev.modern.ie/testdrive/?o=1>
- GUIMark 2
 - <http://www.craftymind.com/guimark2/>
- WebGL Aquarium:
 - <http://webglsamples.googlecode.com/hg/aquarium/aquarium.html>