



# PODSTAWY JavaScript

---

Paweł Kaczmarek

# Zastosowanie JavaScript

- Modyfikacja wyglądu strony
- Modyfikacja zawartości strony
- Dodanie dynamiki strony
- Asynchroniczne wołania do serwera (AJAX) – real-time:
  - Auto-complete
  - Odświeżanie zawartości strony
  - Walidacja formularzy
  - Komunikacja z API

# Zasady JS

- Case sensitive
- Nazwy muszą zaczynać się od litery, \_ lub \$
- Bloki kodu określamy nawiasami klamrowymi { ... }
- Linie kończymy średnikami ;

# Best Practices

- CamelCase
- Piszemy po angielsku (nie używamy polskich znaków)
- Sensowe nazewnictwo

# Słowa kluczowe

abstract	arguments	boolean	break	byte
case	catch	char	class*	const
continue	debugger	default	delete	do
double	else	enum*	eval	export*
extends*	false	final	finally	float
for	function	goto	if	implements
import*	in	instanceof	int	interface
let	long	native	new	null
package	private	protected	public	return
short	static	super*	switch	synchronized
this	throw	throws	transient	true
try	typeof	var	void	while

[http://www.w3schools.com/js/js\\_reserved.asp](http://www.w3schools.com/js/js_reserved.asp) - Pełna lista

# Osadzanie JavaScript - w pliku HTML

**Index.html**

```
<script>
```

```
    console.log('test');
```

```
</script>
```

---

# Osadzanie JavaScript - w pliku JS

**Index.html**

```
<script src = " libFile.js " ></script>
```

**libFile.js**

```
console.log('test');
```

---

# Osadzanie JavaScript - czego nie robić

```
<script >
```

```
    var text = 'tekst w konsoli'  
    console.log(text);
```

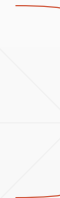


Ten kod się wykona

```
</script>
```

```
<script src = " libFile.js " >
```

```
    var text = 'tekst w konsoli'  
    console.log(text);
```



Ten kod się nie wykona

```
</script>
```

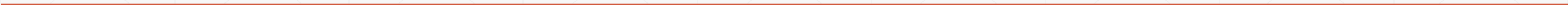
---



# Osadzanie JavaScript

Osadzamy wewnątrz:

- body
- head też działa



## 0.usingJS



# 1. TABLICE I OBIEKTY

# document

- document.title
- document.body
- document.domain
- document.cookie



1a.document

---

# document

- document.**getElementById** ( 'mojeld' )
- document.**getElementsByClassName** ( 'mojaKlasa' )
- document.**getElementsByTagName** ( 'p' )



1a.document

---

# innerHTML

`document.body.innerHTML`

`var kontener = document.getElementById ( 'mojeld' );`

`kontener.innerHTML = 'tekst to wypisania';`



1a.document

---

# 1a.document



## Array - tablice

```
var myTable = [ ];
```

```
var myTable = [ obj1 , obj2 , obj3 ];
```

```
var cars = [ 'Saab', 'Volvo', 'BMW' ];
```



# Array

Do elementów tablicy dostajemy się po indeksie

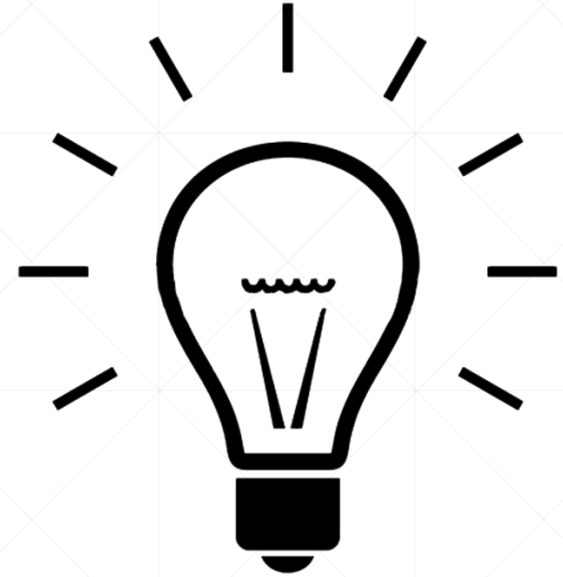
## Array – odczyt

```
var cars = [ 'Saab', 'Volvo', 'BMW' ];
```

```
cars [ 0 ] === 'Saab'
```

```
cars [ 1 ] === 'Volvo'
```

```
cars [ 2 ] === 'BMW'
```



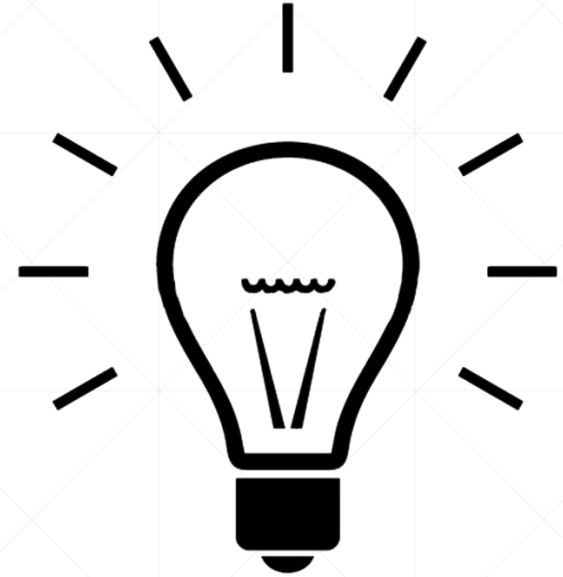
# Array – zapis

```
var cars = [ 'Saab', 'Volvo', 'BMW' ];
```

```
cars [ 2 ] = 'Niemiecki wóz'
```

```
cars [ 3 ] = 'Trabant'
```

```
cars [ 4 ] = 'Czarna Wołga'
```



# Array - length

```
var a = [ ];
```

```
var b = [ "Freeze" ];
```

```
var c = [ "Batman", "Robin", "Freeze", "Riddler" ]
```



```
a.length == 0
```

```
b.length == 1
```

```
c.length == 4
```

# Funkcje raz jeszcze

- Działające w miejscu
- Zwracające wartość

# Array - concat

```
var a = [ "Batman", "Robin" ];  
var b = [ "Freeze", "Riddler" ];  
var c = a.concat(b)
```



```
a == [ "Batman", "Robin" ];  
b == [ "Freeze", "Riddler" ];  
c == [ "Batman", "Robin", "Freeze", "Riddler" ];
```

# Array - pop, push, shift, unshift

unshift()



push()



[ "Batman", "Robin", "Freeze", "Riddler" ];

shift()



pop()



# Array - metody

```
var heroes = [ "Batman", "Robin", "Freeze", "Riddler" ];
```

```
heroes.length;
```

```
heroes.sort();
```

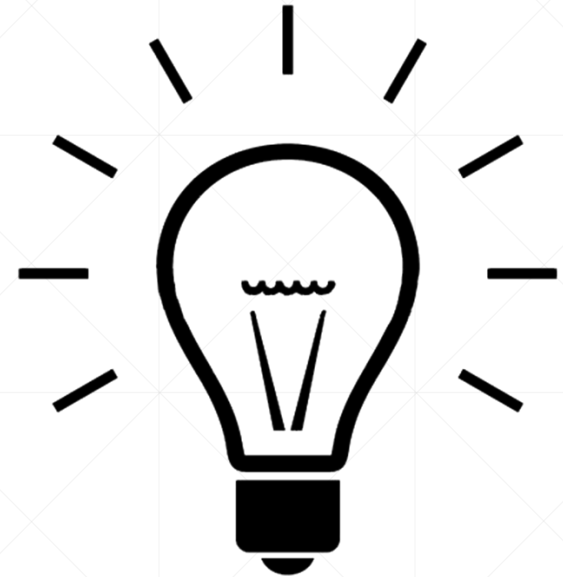
```
heroes.push( "CatWoman" );
```

```
heroes.pop();
```

```
heroes.shift();
```

```
heroes.unshift( "Batman" );
```

```
heroes.concat( [ "Gordon", "Oracle" ] );
```



Link w ćwiczeniu 1b



# 1b.arrays



# Obiekty

Object

Properties

Methods



`car.name = Fiat`

`car.start()`

`car.model = 500`

`car.drive()`

`car.weight = 850kg`

`car.brake()`

`car.color = white`

`car.stop()`

credits: [http://www.w3schools.com/js/js\\_objects.asp](http://www.w3schools.com/js/js_objects.asp)

# Obiekty

```
var myObject = {  
    prop1 : ... ,  
    prop2 : ... ,  
    method1 : ... ,  
    method2 : ...  
};
```



# Obiekty

Do elementów obiektów dostajemy się po nazwie

# Obiekty

```
var car = {  
    type : 'Fiat' ,  
    model : 500 ,  
    color : 'white'  
};
```

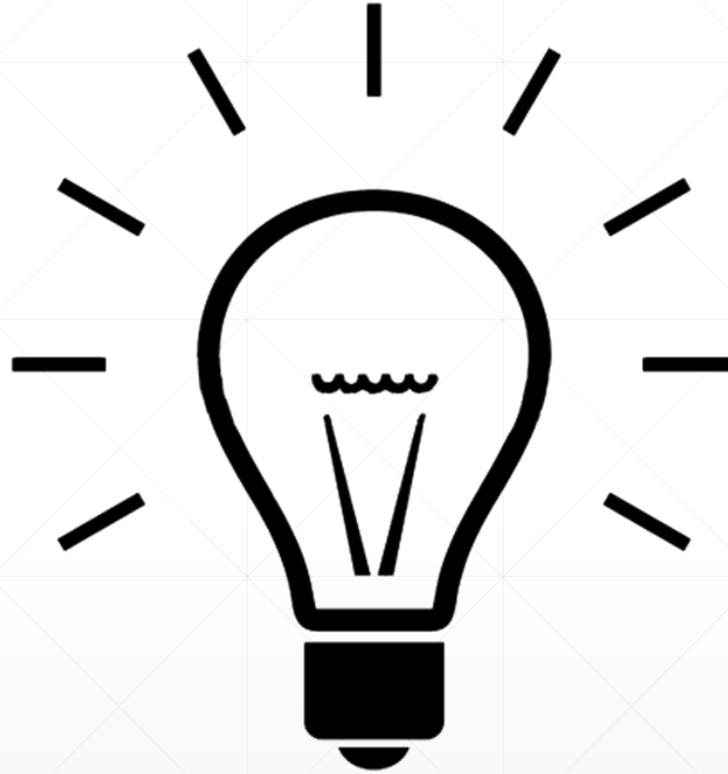
```
car . type           // 'Fiat'  
car . model          // 500  
car . color           // 'white'
```

# Obiekty

```
var car = {  
  type : 'Fiat',  
  model : 500,  
  color : 'white',  
  start : function() { console.log('bruum') },  
  fullModel : function() {  
    return this.type + ' ' + this.model;  
  }  
};
```

```
car.type           // 'Fiat'  
car.model          // 500  
car.color          // 'white'  
car.start()        // undefined  
car.fullModel()    // 'Fiat 500'
```

**1c.object**



## 2. OBIEKTY POMOCNICZE



# Obiekty pomocnicze

- document
- window
- Math
- Date

# window.navigator

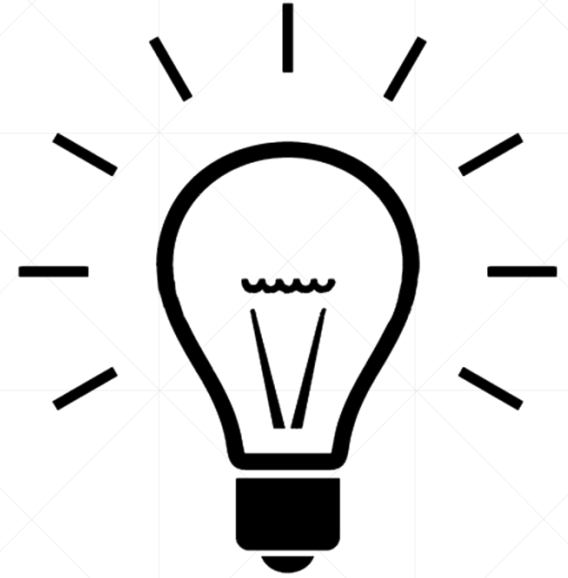
## Informację o przeglądarce

- navigator.language
- navigator.cookieEnabled
- navigator.userAgent
- navigator.appVersion
- navigator.platform



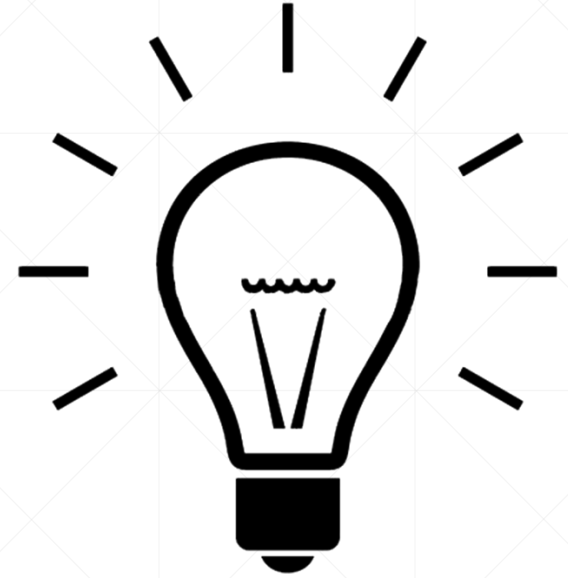
# window.location

- `location.href`
- `location.hostname`
- `location.pathname`
- `location.protocol`
- `location.reload()`
- `location.assign('http://wp.pl')`



# window.history

- `history.back()`
- `history.forward()`



# Math

> Math

< ▼ MathConstructor {E: 2.718281828459045,  
0.4342944819032518...} ⓘ

E: 2.718281828459045

LN2: 0.6931471805599453

LN10: 2.302585092994046

LOG2E: 1.4426950408889634

LOG10E: 0.4342944819032518

PI: 3.141592653589793

SQRT1\_2: 0.7071067811865476

SQRT2: 1.4142135623730951

▶ abs: function abs()

▶ acos: function acos()

▶ acosh: function acosh()

▶ asin: function asin()

▶ asinh: function asinh()

▶ atan: function atan()

▶ atan2: function atan2()

▶ atanh: function atanh()

▶ cbrt: function cbrt()

▶ ceil: function ceil()

▶ clz32: function clz32()

▶ cos: function cos()

▶ cosh: function cosh()

▶ exp: function exp()

▶ expm1: function expm1()

▶ floor: function floor()

▶ fround: function fround()

▶ hypot: function hypot()

▶ imul: function imul()

▶ log: function log()

▶ log1p: function log1p()

▶ log2: function log2()

▶ log10: function log10()

▶ max: function max()

▶ min: function min()

▶ pow: function pow()

▶ random: function random()

▶ round: function round()

▶ sign: function sign()

▶ sin: function sin()

▶ sinh: function sinh()

▶ sqrt: function sqrt()

▶ tan: function tan()

▶ tanh: function tanh()

▶ trunc: function trunc()



# Date - odczyt

```
var now = new Date();
```

- `now.toUTCString();`
- `now.toString();`

<code>getDate()</code>	Get the day as a number (1-31)
<code>getDay()</code>	Get the weekday as a number (0-6)
<code>getFullYear()</code>	Get the four digit year (yyyy)
<code>getHours()</code>	Get the hour (0-23)
<code>getMilliseconds()</code>	Get the milliseconds (0-999)
<code>getMinutes()</code>	Get the minutes (0-59)
<code>getMonth()</code>	Get the month (0-11)
<code>getSeconds()</code>	Get the seconds (0-59)
<code>getTime()</code>	Get the time (milliseconds since January 1, 1970)

# Date - modyfikowanie

```
var now = new Date();
```

```
var newYear = now . getFullYear() + 1;
```

```
now . setFullYear ( newYear );
```

setDate()	Set the day as a number (1-31)
setFullYear()	Set the year (optionally month and day)
setHours()	Set the hour (0-23)
setMilliseconds()	Set the milliseconds (0-999)
setMinutes()	Set the minutes (0-59)
setMonth()	Set the month (0-11)
setSeconds()	Set the seconds (0-59)
setTime()	Set the time (milliseconds since January 1, 1970)

**2b.date**





# 3. TYPY DANYCH

# Typy danych

- **Boolean**      `true, false`
- **Number**      `225, 23.4`
- **String**      `'Napis', ''`
- **Array**      `[ 1, 12, 3 ], [ 'test', 'napis' ], [ 12, 'napis', true ]`
- **Object**      `{ name: 'John', age: 34 }`
- ***undefined***      `var x;`

# Zmienne raz jeszcze

- Przechowują wartości, tablice, obiekty i..... funkcje (funkcje anonimowe)
- Silne porównania (===, !==)

# Number

- 12, 12.00, 12e5, 12e-5, 0xFF

- **NaN**

5 \* 'one' , 5 + NaN

- **Infinity**

- Number.parseFloat()

'10', '1.12', '1.12aa', 'a12.22'

- Number.parseInt()

'10', '1.12', '1.12aa', 'a12.22'

- toFixed()

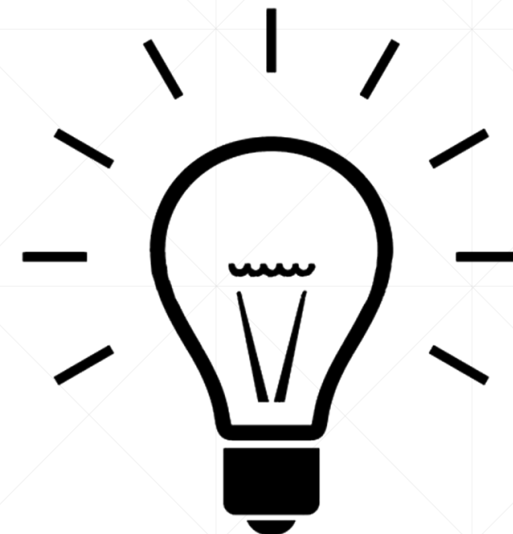
(12).toFixed(2) -> '12.00'

12.987.toFixed(2) -> '12.99'

- toString()

(12).toString() -> '12'

12.55.toString() -> '12.55'



**3a.number**



# string

```
var text = 'Moj text'
```

- `text . length`
- `text . indexOf( 'text' )`
- `text . toUpperCase()`
- `text . toLowerCase()`
- `text . trim()`

[https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global\\_Objects/String#Methods\\_2](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/String#Methods_2)

## string – znaki specjalne

- `var text = "Maciek powiedział \" Nic się nie stało \" "`
- `var text = " Bardzo długa linia kodu, która będzie się ciągnąć, \`  
a którą chcielibyśmy złamać "
- `var text = " Pierwsza część " + " Druga część "`

**3b.string**





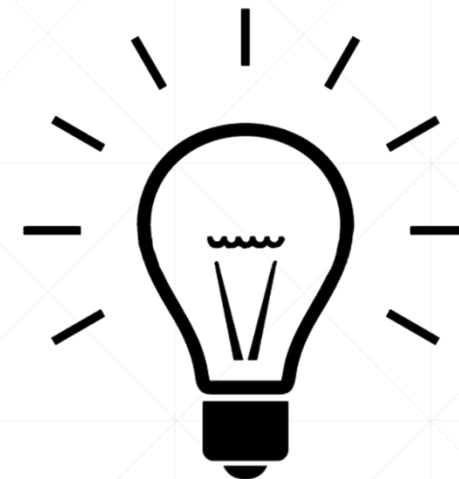
## 4. EVENTY HTML

# Eventy HTML

- Zdarzenia przytrafiające się elementom HTML
- onchange, onclick, onmousedown....
- Kilka metod obsługi

[http://www.w3schools.com/jsref/dom\\_obj\\_event.asp](http://www.w3schools.com/jsref/dom_obj_event.asp) - Pełna lista eventów

# Obsługa eventów HTML – przez atrybut



- `<div onclick="console.log ( 'kliknięcie' ) ">Click on this text!</div>`

# addEventListener – przypisanie funkcji bez parametru

```
function  zmienTekst ( ) { ... }
```

```
var element = document . getElementById ( 'myId' )
```

## addEventListener – przypisanie funkcji bez parametru

```
function  zmienTekst ( ) { ... }
```

```
var element = document . getElementById ( 'myId' )
```

- *element* . addEventListener ( 'click' , **zmienTekst** ) ;

# addEventListener – przypisanie funkcji bez parametru

```
function  zmienTekst ( ) { ... }
```

```
var element = document . getElementById ( 'myId' )
```

- *element* . addEventListener ( 'click' , **zmienTekst** ) ;
- *element* . attachEvent ( 'onclick' , **zmienTekst** ) ;      // (< IE9 )

## ... i jeszcze raz funkcje

### Funkcje nazwane

```
function nazwaFunkcji () { }
```

```
nazwaFunkcji ();
```

### Funkcje anonimowe

```
function () { }
```

```
var nazwaFunkcji = function () { }
```

```
nazwaFunkcji ();
```

# addEventListener – przypisanie funkcji z parametrem

```
function zmienTekst ( nowyTekst ) { ... }  
var element = document . getElementById ( 'myId' )
```



# addEventListener – przypisanie funkcji z parametrem

```
function zmienTekst ( nowyTekst ) { ... }
```

```
var element = document . getElementById ( 'myId' )
```

- `element . addEventListener ( 'click' ,  
function () { zmienTekst( 'jakiś tekst' ) } );`

# addEventListener – przypisanie funkcji z parametrem

```
function zmienTekst ( nowyTekst ) { ... }
```

```
var element = document . getElementById ( 'myId' )
```

- `element . addEventListener ( 'click' ,  
function () { zmienTekst( 'jakiś tekst' ) } );`
- `element.attachEvent ( 'onclick' ,  
function () { zmienTekst( 'jakiś tekst' ) } ); // ( < IE9 )`

## addEventListener – przekazanie elementu wywołującego zdarzenie do funkcji

```
function zmienKlase ( kontener , nowaKlasa ) {  
    kontener . getAttribute ( 'class' );  
    kontener . setAttribute ( 'class' , nowaKlasa );  
}  
  
element . addEventListener ( 'click' ,  
    function () { zmienKlase ( this , 'mojaKlasa' ) } );
```

## 4.eventy



**5.people**





# PODSTAWY JavaScript

---

Dziękuję za uwagę!

## 5. BONUS =)

# Typ danych - Boolean

**Boolean** ( *expression* )



TRUE

- true
- 100, 3.14, -15
- 'Text', 'false'
- 1+2+3
- 5 < 6
- [], {}

FALSE

- false
- 0, -0
- "" – pusty string
- *undefined, null, NaN*



# Obiekty – alternatywna metoda adresacji pól

<code>car.type</code>	<code>==</code>	<code>car [ 'type' ]</code>
<code>car.model</code>	<code>==</code>	<code>car [ 'model' ]</code>
<code>car.color</code>	<code>==</code>	<code>car [ 'color' ]</code>
<code>car.start()</code>	<code>==</code>	<code>car [ 'start' ] ()</code>
<code>car.fullModel()</code>	<code>==</code>	<code>car [ 'fullModel' ] ()</code>

# typeof - sprawdź typ zmiennej/wartości

Wpiszmy na konsolę:

- `typeof "John"` // string
- `typeof 3.14` // number
- `typeof false` // boolean
- `typeof [1,2,3,4]` // object
- `typeof {name:'John', age:34}` // object

