



PODSTAWY JavaScript i jQuery

Paweł Kaczmarek

dzień 1

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 - Pełna lista

Osadzanie JavaScript - w pliku HTML

Index.html

```
<script>
```

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

```
</script>
```

Osadzanie JavaScript - w osobnym pliku .js

Index.html

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

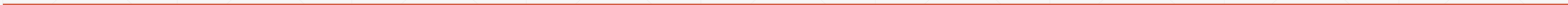
libFile.js

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

tag <script>

Osadzamy wewnątrz:

- body
- head



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>
```

0.Sandbox



jQuery

jQuery?

- Biblioteka JavaScript - JavaScript Query
- Służy do wyszukiwania elementów na stronie i wykonywania na nich akcji
- Czasami upraszcza programowanie w JavaScript
- Działa pod wszystkimi przeglądarkami
- Znajduje się w pojedynczym pliku *.js

Konstrukcja zapytania jQuery

`$(selector).action()`

- `$`
 - zmienna globalna, w której trzymane jest jQuery
- `selector`
 - wartość textowa, string, definiuje jakich elementów szukamy
- `action`
 - akcja wykonana na wszystkich znalezionych elementach

Przykłady zapytań jQuery

`$ (selector).action()`

- `$('p').hide()`
- `$('div').show()`
- `$('div.news').addClass('przeczytany')`

Zapytania jQuery zwracają kolekcje elementów

```
<p class="klikalny">Element 1</p>  
<p class="klikalny">Element 2</p>  
<p class="klikalny">Element 3</p>  
<p class="klikalny">Element 4</p>
```

`$('p')`



```
$( 'p' )  
[  
  <p class="klikalny">Element 1</p>,  
  <p class="klikalny">Element 2</p>,  
  <p class="klikalny">Element 3</p>,  
  <p class="klikalny">Element 4</p>]  
|
```

jQuery - Proste selectory

- `$('div')`
- `$('.artykul')`
- `$('#naglowek')`
- `$('div.artykul')`
- `$('div, p')`
- `$('div p')`
- `$('div p.artykul')`

<http://api.jquery.com/category/selectors/> - Pełna lista

1.jq.warmup



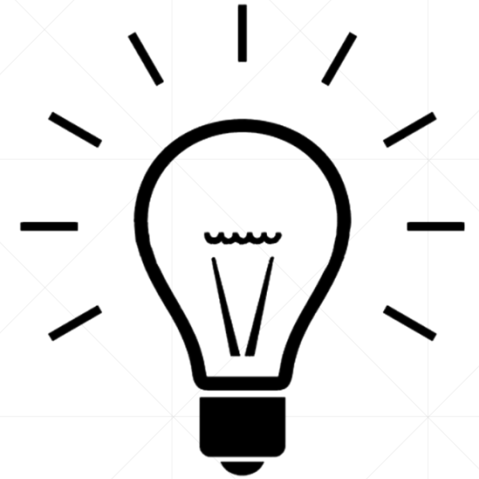
Events (zdarzenia)

Eventy HTML

- Zdarzenia przytrafiające się elementom HTML
- onchange, onclick, onmousedown....
- **asynchroniczne !!**

http://www.w3schools.com/jsref/dom_obj_event.asp - Pełna lista eventów

Obsługa eventów – przez atrybut HTML



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

jQuery - Obsługa eventów

- click
- dblclick
- change
- mousedown
- hover
- keydown

`$(selector).action()`

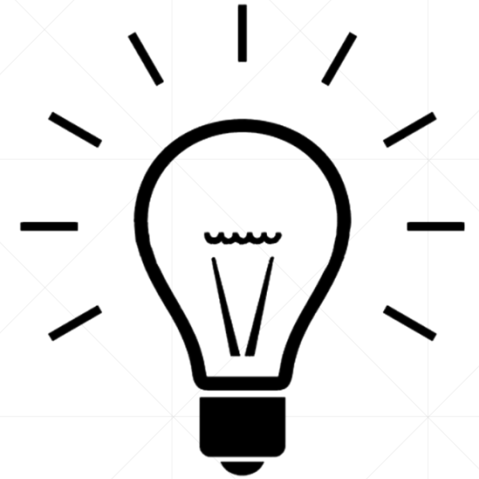
`$('.nazwaKlasy').click (function() { });`

`$('.nazwaKlasy').dblclick(function() { });`

`$('.nazwaKlasy').mousedown (function() { });`

`$('.nazwaKlasy').change(function() { });`

jQuery - Obsługa eventów



```
$( '.nazwaKlasy' ).click ( function() {  
    $( this ).hide();  
} );
```

anonymous functions and callbacks

Funkcja nazwana

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

```
nazwaFunkcji ();
```

Funkcja anonimowa

```
function () { }
```

```
...
```


Funkcja nazwana

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

```
nazwaFunkcji ();
```

Funkcja anonimowa

```
function () { }
```

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

```
nazwaFunkcji ();
```

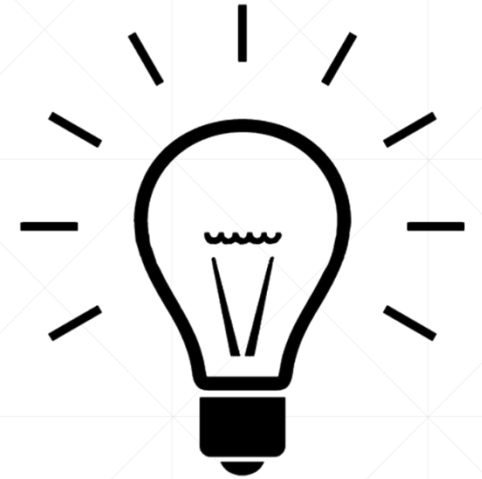
callback

- funkcja przekazana do innej funkcji jako parametr
- bardzo często jest funkcją anonimową

2.jq.events



jQuery - funkcja .find()



```
$( '.nazwaKlasy' ).click ( function() {  
    $( this ).find('.podelement').hide();  
} );
```

jQuery - pokazywanie, ukrywanie elementów

```
var el = $('naszSelector');
```

- `el.show()`
- `el.hide()`
- `el.fadeIn(400)`
- `el.fadeOut(400)`
- `el.fadeToggle(400)`
- `el.fadeTo(400, 0.5)`
- `el.slideUp(500)`
- `el.slideDown(500)`
- `el.slideToggle(500)`
- `el.stop()`

jQuery - Atrybut class

```
var els = $('naszSelector');
```

- **els**.hasClass('klikalny')
 - **true**, jeżeli **przynajmniej jeden** element posiada klasę 'klikalny'
- **els**.addClass('klikalny')
 - dodaje klasę 'klikalny' do wszystkich elementów
- **els**.removeClass('klikalny')
 - usuwa klasę 'klikalny' do wszystkich elementów
- **els**.toggleClass('klikalny')
 - dodaje/usuwa klasę 'klikalny' do/z wszystkich elementów

3.jq.classes



Uruchamiamy zapytania jQuery dopiero po załadowaniu całego dokumentu

```
$( document ).ready( function() {  
    $( 'p' ).click(....)  
})
```


jQuery - tworzenie nowych elementów HTML

```
var nowyDiv = $('<div>'),  
    nowyParagraf = $('<p>');
```



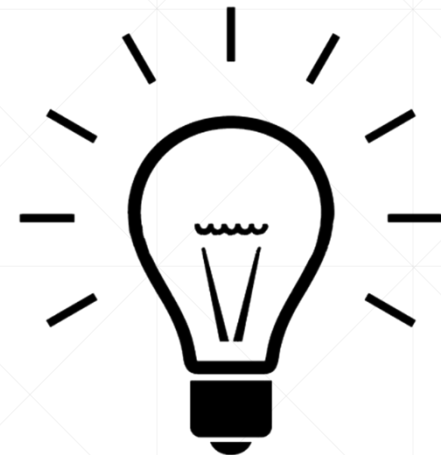
jQuery - Kolejowanie funkcji - method chaining

```
var newElement = $('<div>').addClass('hero').text('Batman');
```

jQuery - funkcja append()

```
var newElement = $('<div>').addClass('hero').text('Batman');  
$('#heroes').append(newElement);
```

- dodawanie dynamicznie stworzonych elementów do strony



jQuery- dokumentacja

<http://api.jquery.com>

Array (tablica)

Array - tablice

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

```
var liczby = [ 10 , 11, 12 ];
```

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

array.forEach()

```
var heroes = ['Batman', 'Robin', 'Gordon'];  
  
heroes.forEach(function(hero, index) {  
    console.log(hero, index);  
})
```

- wywołuje funkcję dla każdego elementu tablicy
- za każdym wywołaniem przekazuje do funkcji dwa parametry:
 - pierwszy - aktualny element tablicy
 - index - index aktualnego elementu tablicy

array.map()

```
var heroes = ['Batman', 'Robin', 'Gordon'];  
  
var modifiedHeroes = heroes.map(function(hero) {  
    return 'Hero - ' + hero;  
});
```

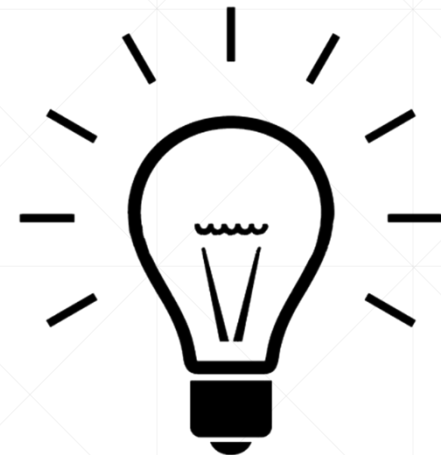
- tworzy nową tablicę
- nowa tablica zawiera nowe elementy, stworzone w oparciu o elementy ze starej tablicy
- stara tablica pozostaje niezmienną

Array - Kolejowanie funkcji - method chaining

```
[ 'Batman', 'Robin', 'Gordon' ]  
  .map(function (hero) {  
    return 'Hero - ' + hero;  
  })  
  .forEach(function (hero) {  
    console.log(hero);  
  })
```

4.Array.forEach





Array - dokumentacja

<https://developer.mozilla.org>

Array

Do elementów tablicy dostajemy się po indeksie

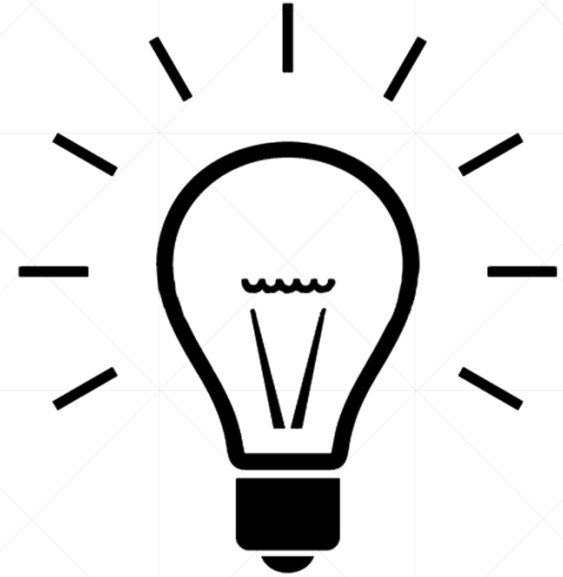
Array – odczyt

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

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

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

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



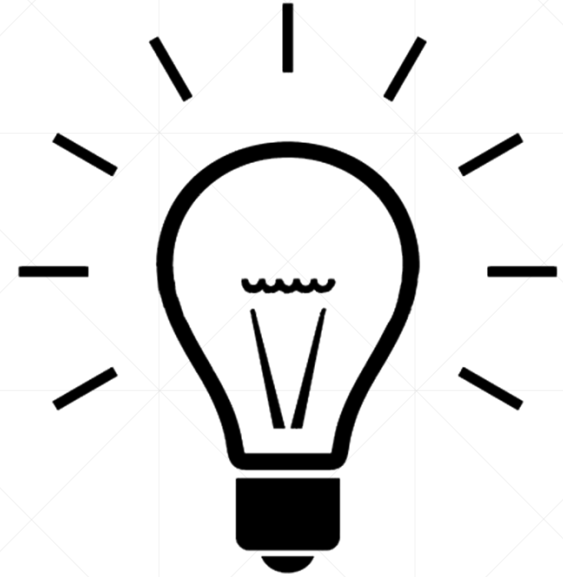
Array – zapis

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

```
cars [ 2 ] = 'Niemiecki wóz'
```

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

```
cars [ 4 ] = 'Czarna Wołga'
```



Array.length - liczba elementów w tablicy

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

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

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



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

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

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

Array.concat - sklejanie tablic

```
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('Catwoman')`



`push('Catwoman')`



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

`shift()`

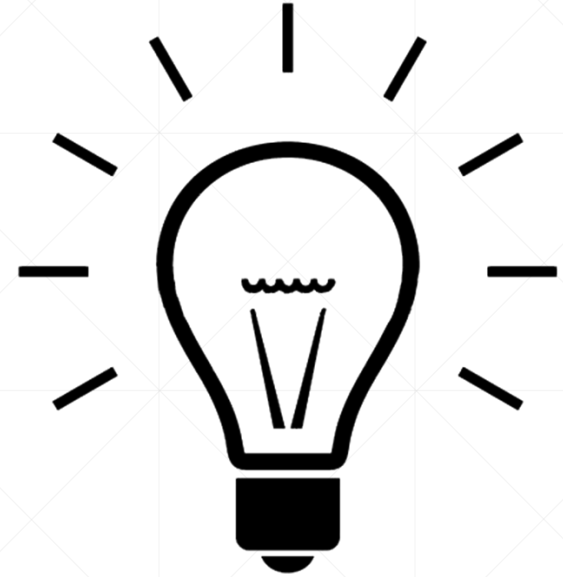


`pop()`



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" ] );
```





PODSTAWY JavaScript i jQuery

Paweł Kaczmarek

dzień 2

Objects (obiekty)

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

Obiekty

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



Obiekty

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

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

Obiekty

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

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


Obiekty

Do elementów obiektów dostajemy się po nazwie

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>

1.objects



array.filter()

```
var numbers = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10];
```

```
numbers.filter(function(num) {  
    return num > 9  
});
```



[10]

```
numbers.filter(function(num) {  
    return num > 5 && num < 9  
});
```



[6, 7, 8]

```
numbers.filter(function(num) {  
    return num < 3 || num < 8  
});
```



[1, 2, 9, 10]

2.people



jQuery - animacje

```
$(selector).animate(  
    {  
        parametr_1_css:    wartość_docelowa_1 ,  
        parametr_2_css:    wartość_docelowa_2 ,  
        parametr_3_css:    wartość_docelowa_3  
    },  
    czasTrwaniaWMilisekundach  
);
```

Animacje - przykład

```
$('#mojIdentyfikator') . animate( {  
    opacity: 0.6,  
    width: 400  
}, 3000 );
```

3.jq.animacije



Atrybuty HTML

```
var el = $('naszSelector');
```

- `el.attr('attrName')` - zwraca wartość atrybutu `'attrName'` **pierwszego ze znalezionych elementów**
- `el.attr('attrName', 'newValue')` - ustawia wartość atrybutu `'attrName'` **wszystkich elementów**
- `el.removeAttr('attrName')` - usuwa atrybut `'attrName'` ze **wszystkich elementów**

Modyfikowanie HTML

```
var elements = $('naszSelector');
```

- `elements.html()`
 - zwraca zawartość HTML pierwszego ze znalezionych elementów
- `elements.html('Nowy HTML')`
 - ustawia zawartość HTML wszystkich elementów

Modyfikowanie zawartości tagów HTML

```
var elements = $('nazSelector');
```

- `elements.text()`
 - zwraca tekst stanowiący agregację tekstu ze wszystkich znalezionych elementów
- `elements.text('Nowa zawartość')`
 - ustawia tekst we wszystkich znalezionych elementach

Modyfikowanie zawartości tagów typu input

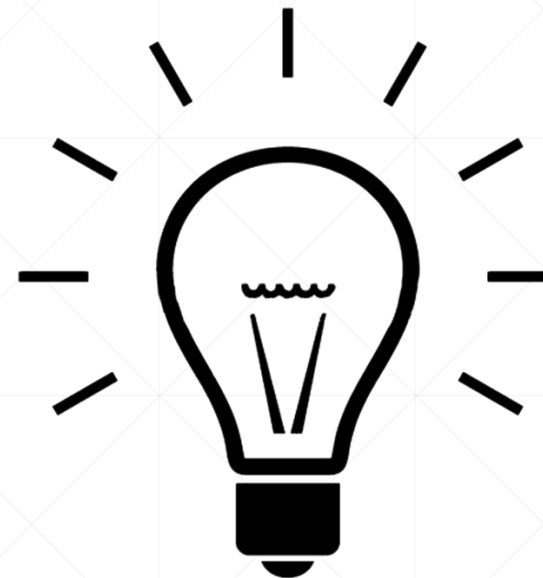
```
var elements = $('nazSelector');
```

- `elements.val()`
 - zwraca wartość atrybutu value **pierwszego** ze znalezionych elementów
- `elements.val('Nowa wartość')`
 - ustawia atrybut value we wszystkich znalezionych elementach

Trawersowanie dokumentu

```
var el = $('naszSelector');
```

- `el.children()`
- `el.parent()`
- `el.siblings()`
- `el.next()`
- `el.prev()`



Obiekty wbudowane (Standard built-in objects)

https://developer.mozilla.org/en/docs/Web/JavaScript/Reference/Global_Objects

Obiekty pomocnicze

- document
- window
- Math
- Date

window.navigator

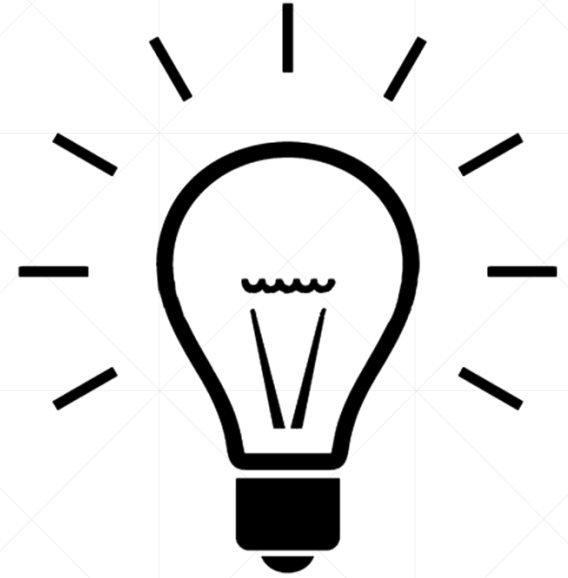
Informację o przeglądarce

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



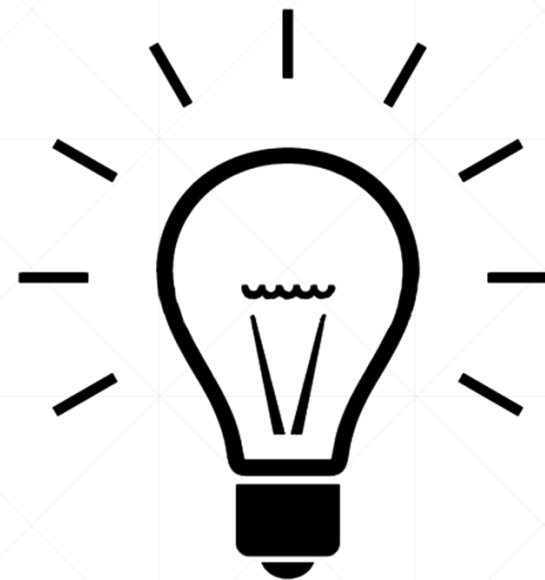
window.location

- `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()

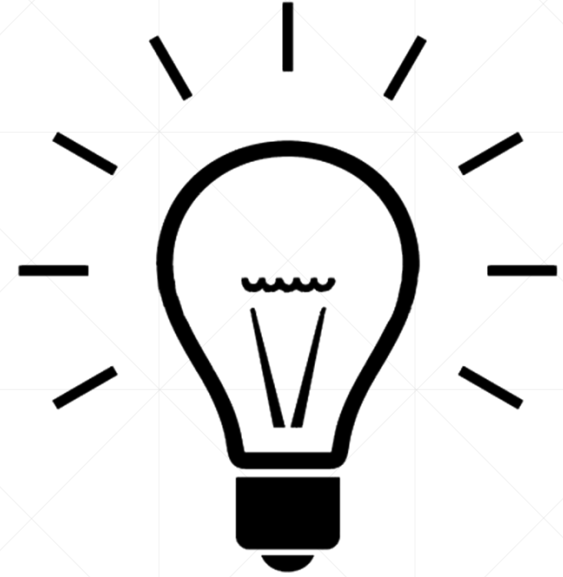
▶ sinh: function sinh()

▶ sqrt: function sqrt()

▶ tan: function tan()

▶ tanh: function tanh()

▶ trunc: function trunc()



Date - odczyt

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

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

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

4.date



Pozostałe typy danych

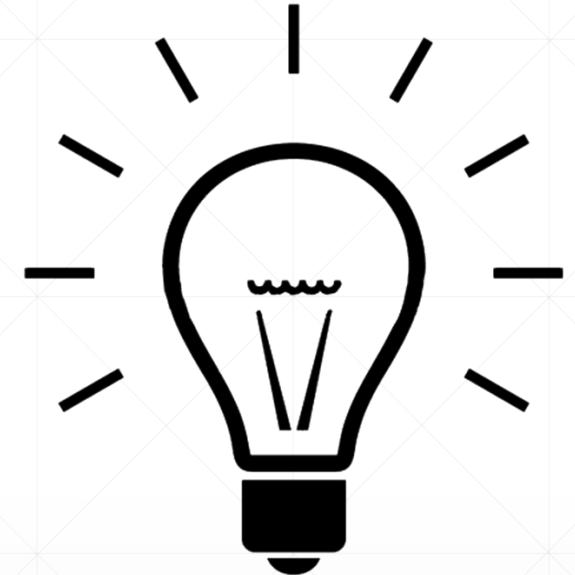
Typy danych

- **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;`

typeof - sprawdza 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



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()

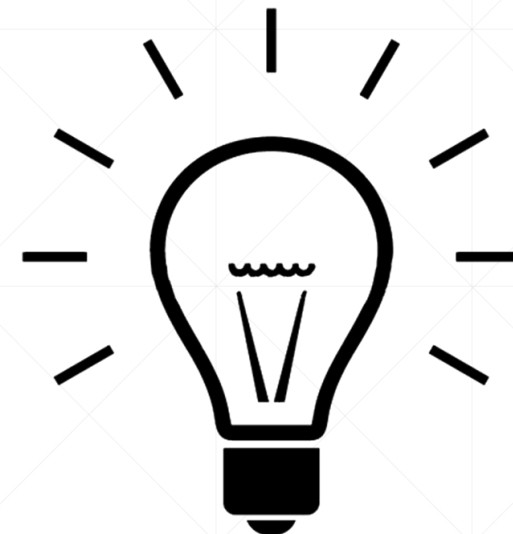
(12).toFixed(2) -> '12.00'

12.987.toFixed(2) -> '12.99'

- toString()

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

12.55.toString() -> '12.55'



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

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



Typ danych - Boolean

Boolean (*expression*)



TRUE

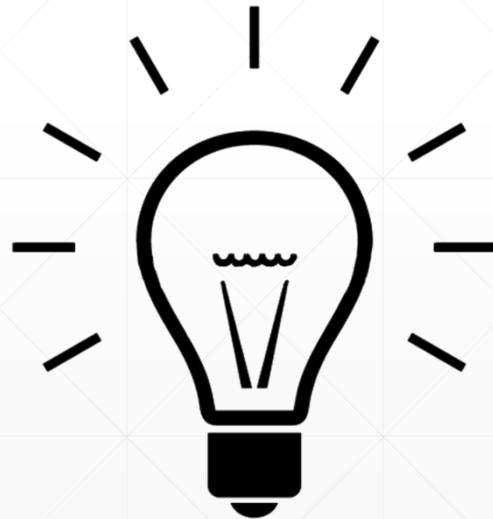
- true
- 100, 3.14, -15
- 'Text', 'false'
- 1+2+3
- 5 < 6
- [], {}

FALSE

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

JavaScript Koans

<https://github.com/mrdavidlaing/javascript-koans>





PODSTAWY JavaScript

Dziękuję za uwagę!