

JavaScript

dobre praktyki ;)

Dawid Dziurdzia

JavaScript

- Stworzony przez Netscape w latach 90
- W tej chwili mamy wersję 1.8.5 (ECMA Script 5)
 - Wersja 2.0 jest w produkcji
- V8, Nitro, SpiderMonkey, Rhino, Carakan
- Node.js, Apache couchDB

Javascript i HTML

- `<script async src="jquery.min.js"></script>`
- `<script async src="leaflet.min.js"></script>`
- `<script async src="underscore.js"></script>`
- ...
- `<script async src="main.min.js"></script>`

Nie mamy gwarancji, że skrypty wykonają (załadują) się w takiej kolejności w jakiej zostały zadeklarowane.

Require.js

```
<script  
  data-main="scripts/main"  
  src="scripts/require.js">  
</script>
```

Require.js

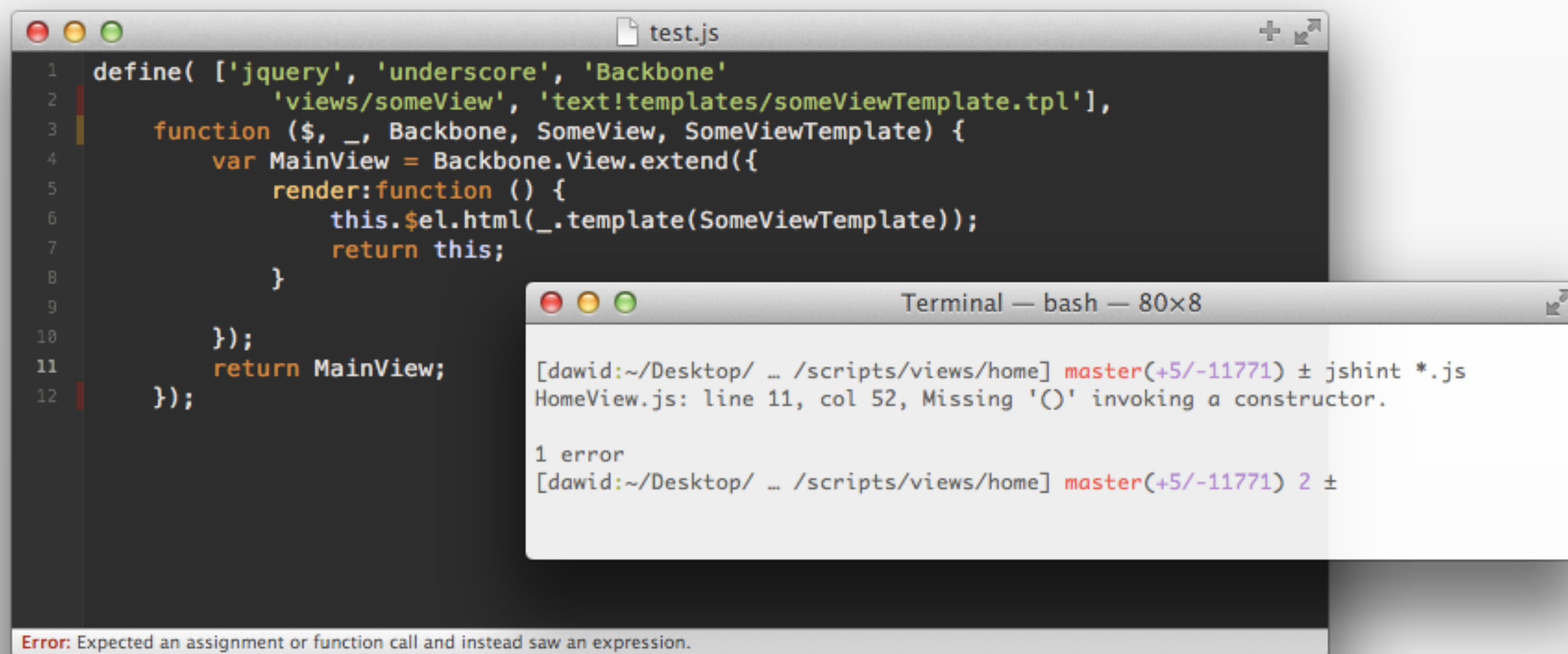
- Asynchronous module definition (AMD)
- Cross-origin resource sharing (CORS)
- r.js

<http://requirejs.org/docs/whyamd.html>

Jak programować w JS?

na co zwracać uwagę...

Używajmy JSLint / JSHint



The image shows a code editor window titled 'test.js' with the following JavaScript code:

```
1 define( ['jquery', 'underscore', 'Backbone'
2         'views/someView', 'text!templates/someViewTemplate.tpl'],
3         function ($, _, Backbone, SomeView, SomeViewTemplate) {
4             var MainView = Backbone.View.extend({
5                 render:function () {
6                     this.$el.html(_.template(SomeViewTemplate));
7                     return this;
8                 }
9             });
10            });
11            return MainView;
12        });
```

Below the code editor is a terminal window titled 'Terminal — bash — 80x8' showing the output of running JSHint:

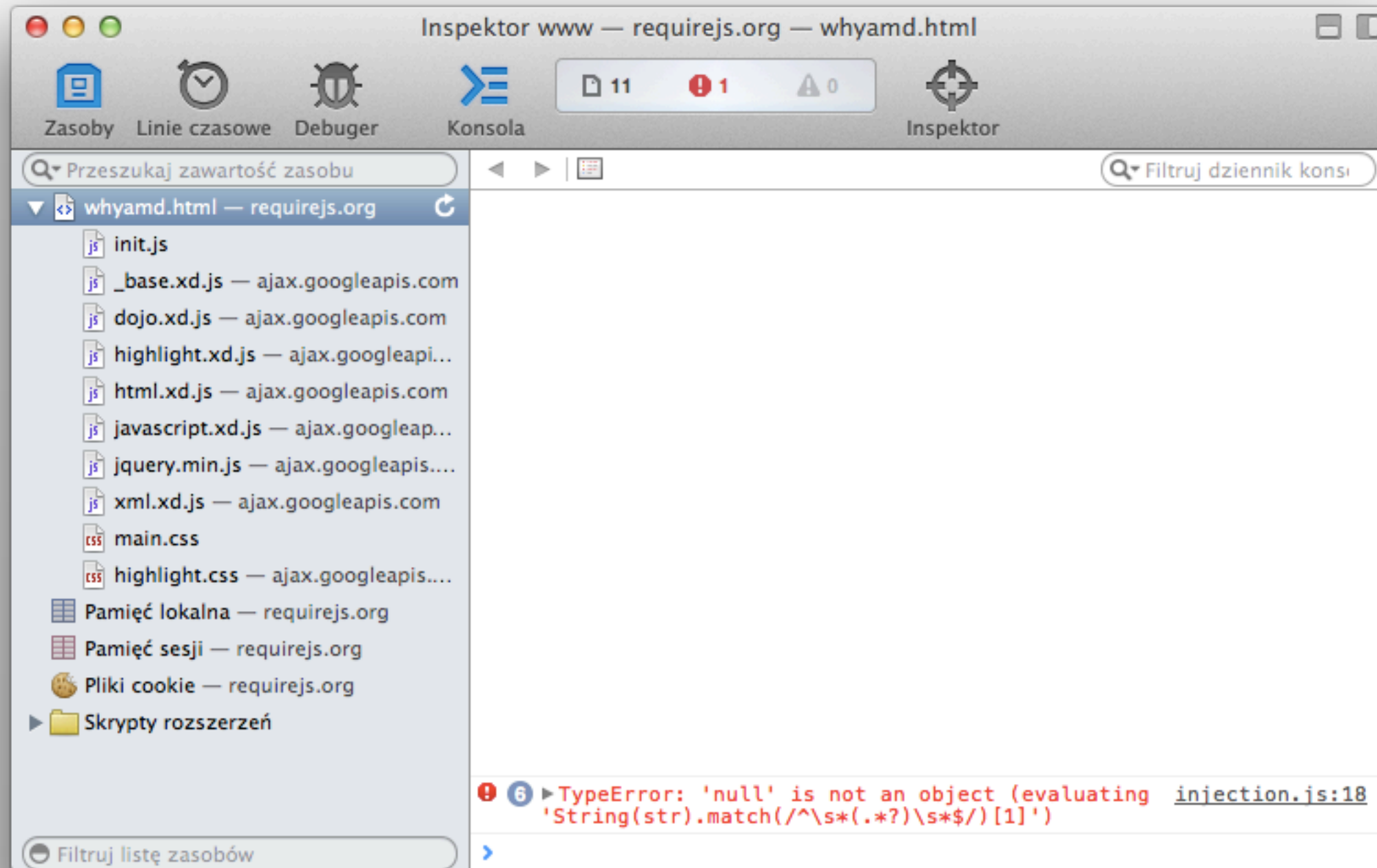
```
[dawid:~/Desktop/ ... /scripts/views/home] master(+5/-11771) ± jshint *.js
HomeView.js: line 11, col 52, Missing '()' invoking a constructor.

1 error
[dawid:~/Desktop/ ... /scripts/views/home] master(+5/-11771) 2 ±
```

At the bottom of the code editor window, there is a red error message: **Error:** Expected an assignment or function call and instead saw an expression.

<http://www.jshint.com/docs/>

i debuggera!



'use strict';

- Pomyłki są traktowane jako błędy
 - NaN = 5;
- Dokładniejsze sprawdzanie składni

Browsers don't reliably implement strict mode yet, so don't blindly depend on it. Strict mode changes semantics. Relying on those changes will cause mistakes and errors in browsers which don't implement strict mode.

https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Functions_and_function_scope/Strict_mode

Porównania

```
var test = true;
```

```
if(test=="1")
```

```
↳ console.log( 'whaaat?' );
```

```
if(test=== "1")
```

```
↳ console.log( ':' );
```

Porównania

```
typeof(5);  
typeof('a');  
typeof(NaN);  
typeof(new Object());  
typeof([]);  
typeof(null);  
typeof(myVar);  
// inne typy: 'function', 'boolean'
```

isArray?

Więc jak sprawdzić, czy zmienna to tablica...?

```
Object.prototype.toString.call( someVar )  
=== '[object Array]'
```

eval() = evil

- wolniejsze wykonywanie
- trudniejsze w debugowaniu
- `eval(`
 `'$(".facebook_like").click();'`
 `);`
- zawsze jest inne rozwiązanie :)

```
setInterval(  
    "console.log('a')",  
    3000  
);
```

Konwencja

```
if ( ... ) {  
    workToDo();  
}
```

```
if ( ... ) workToDo();
```

```
( ... ) ? workToDo() :  
↪ console.log( 'false' );
```

Konwencja

```
var obj = {  
    ...  
};
```

```
var Obj = (function(){  
    this.a = 1;  
})();
```

```
function workToDo (paramOne, paramTwo) {  
    ...  
}
```

```
var arr = [1, 2, 3];
```

```
var _privateVariable;  
var normalVariable;
```


Tworzenie obiektów

```
var car = new Object();  
car.vendor = 'BMW';  
  
...  
car.name = function () {  
  
...  
}
```

VS.

```
var car = {  
  vendor = 'BMW';  
  name = function () {  
    ...  
  }  
}
```

...analogicznie z tablicami

```
var arr = new Array();  
var arr =  
    ['some object', 2, 4];  
console.log(arr.join());
```

Zmienne, zmienne...

```
var jakasZmienna;  
var innaZmienna  
var jeszczeJednaZmienna1234;  
  
var jakasZmienna,  
    innaZmienna  
    jeszczeJednaZmienna1234;
```

Namespacing

```
var Car = Car || {  
  vendor : 'BMW',  
  model : 'x5',  
  name : function() {  
    ↪    return this.vendor + ' ' +  
        this.model;  
  }  
}  
console.log(Car.name());
```

```
var man = (function () {  
    var _man = {  
        name: 'some',  
        lastName: 'name'  
    };  
    return _man;  
})();
```

Fabryka obiektów

```
var man = function (name, surname) {  
  var obj = {};  
  obj.name = name;  
  obj.surname = surname;  
  obj.sayHi = function() {  
    console.log(obj.name + ' ' + obj.surname);  
  }  
  
  var oPublic = {  
    sayHi: obj.sayHi  
  };  
  
  return oPublic;  
};  
  
var firstMan = man('John', 'Smith');  
firstMan.sayHi();  
console.log(firstMan.name);
```

<http://www.yarpo.pl/2011/01/11/tworzenie-obiektow-w-js/>

Zabezpieczanie obiektu

```
Object = function() {  
    console.log( 'Gotcha!' )  
};  
var a = new Object();  
  
delete Object;  
var obj1 = new Object(); // !!!  
  
var obj2 = {};
```


Zabezpieczanie obiektu

```
Object.preventExtensions(obj);  
Object.isExtensible(obj);
```

```
Object.seal(obj);  
Object.isSealed(obj);
```

```
Object.freeze(obj);  
Object.isFrozen(obj);
```

METODA OBJECT.*\OPERACJE	ZMIANA WARTOŚCI WŁASNOŚCI	{PRZE- }KONFIGUROWANIE WŁASNOŚCI	DODAWANIE NOWYCH WŁASNOŚCI
(brak)	TAK	TAK	TAK
preventExtensions()	TAK	TAK	NIE
seal()	TAK	NIE	NIE
freeze()	NIE	NIE	NIE

Zabezpieczanie właściwości obiektu

Deskryptor własności w ECMAScript 5

<http://blog.marcoos.com/2011/03/20/ecmascript-5-deskryptory-wlasnosci/>

```
var obj = {};  
Object.defineProperty(obj, "name",  
{ value: function(){return 'a';},  
writable: false });  
obj.name =  
    function(){return 'b';} // !!!
```

Unobtrusive JavaScript

```
<div id="x" onchange="alert('X  
hovered!')"/>
```

Separacja kodu HTML od logiki

```
var x = document.getElementById('x');  
if (x) {  
    x.onmouseover = alert('X hovered!');  
}
```

DOM

- Zmiany w DOM są wolne!
- Więc unikajmy ich
 - Elementy dodajmy raz.
- Szukanie elementu po **ID** jest najszybsze

Zewnętrzne biblioteki

\$, _, json, ...

\$

\$

\$

\$

jQuery

\$

\$

\$

\$

\$

\$

Jeśli możesz coś zrobić
bez jQuery ...

...to zrób to bez jQuery!

DOM Ready

```
( $(function($) {  
    //application entry point  
})) (jQuery);
```


Selektory CSS

```
$('#menu li:not(active)  
↳ a[href^=http]')[, context]);
```

Możemy używać selektorów takich jak w CSS:

#id

.class

[attr=]

:not(class)

Function chaining

// No chaining

```
$("#menu").fadeIn('fast');  
$("#menu").addClass("active");  
$("#menu").css('marginRight',  
  '10px');
```

// vs. with chaining

```
$("#menu").fadeIn('fast')  
  .addClass("active")  
  .css('marginRight', '10px');
```

problem?

```
$('#a').click(function(){...})  
↳ .fadeIn('fast').css('background', '#fff')  
↳ .toggleClass('active')
```

A co, jeśli mamy na stronie kilkaset odnośników?

– *Każda funkcja to osobna pętla*

```
$('#a').each(function(index, value){  
    value.click(function(){...});  
    value.fadeIn('fast');  
    ...  
});
```

Object caching

```
$('#menu li:not(active)  
↳ a[href^=http]').hide();  
$('#menu li:not(active)  
a[href^=http]').addClass('icn-arrow');  
...
```

vs.

```
var menu_outgoing_links = $('#menu  
↳ li:not(active) a[href^="http://"]');  
menu_outgoing_links.hide();  
menu_outgoing_links.addClass('icn-arrow');  
...
```

Zdarzenia

```
$('#elem').on('nazwa_zdarzenia', function () { ... });  
$('#elem').one('nazwa_zdarzenia', function () { ... });
```

```
$(document).on('click', 'button.accept',  
↳ function () { ... });
```

```
$('#elem').trigger('click');  
$('#elem').click();
```

```
$('#elem').off( [ 'nazwa_zdarzenia' ] );
```

```
$('#elem').click(function () { ... });  
$('#elem').dblclick(function () { ... });  
$('#elem').scroll(function () { ... });  
$('#elem').dblclick(function () { ... });
```

AJAX

```
$.getJSON('path/to/json', function(results) {  
    // callback  
    // results contains the returned json  
});
```

```
$.ajax({  
    type: 'GET',  
    url : 'path/to/json',  
    data : yourData,  
    dataType : 'json',  
    success : function( results ) {  
        console.log('success');  
    })  
});
```

underscore

Przecież mamy już „\$”

Underscore daje nam dostęp do funkcji, które znamy (bądź nie ;)) z języków takich jak Python i Ruby

Zajmuje tylko około **4kB**.

_.select

_.map

_.reduce

_.pluck

_.all

_.uniq

_.range

_.intersection

_.keys

_.values

_.defaults

_.bind

_.template

Nie trzeba rezygnować z jQuery

Przykłady użycia

```
_(cars).select(function (cars) { return horsepower > 115; } );
```

```
var cars = [{vendor : 'BMW', model : '3 Gran Turismo'},  
↪ {vendor : 'Porsche', model : '911'}];  
var vendors = _(cars).pluck('vendor');
```

```
var carNames = _(cars).map(function (value) { return value.vendor + ' ' +  
↪ value.model; });
```

```
_.range(0,1000,100); // [0, 100, 200, 300, 400, 500, 600, 700, 800, 900]
```

```
_.defaults(params, defaults);
```

```
var o = { name: 'name' };  
var f = function () { return this.name };  
var binded0 = _.bind(f, o);  
var n = binded0(); // n = 'name'
```

```
_.template(htmlCode, dataHash) // lightweight alternative to handlebars
```

Parsowanie JSON

JSON

- Nie używamy do tego eval()!
- Są do tego odpowiednie narzędzia
 - `JSON.parse(jsonData, reviver);`
 - `JSON.stringify(myObject, replacer);`
 - `json2.js`
- W JavaScript 2.0 ma to być wbudowane...

<https://github.com/douglascrockford/JSON-js>



<http://caniuse.com/>

Więcej informacji

- <https://github.com/stevekwan/best-practices/blob/master/javascript/best-practices.md>
- <http://net.tutsplus.com/tutorials/javascript-ajax/getting-cozy-with-underscore-js/>
- <http://net.tutsplus.com/tutorials/javascript-ajax/24-javascript-best-practices-for-beginners/>
- <https://developer.mozilla.org/en-US/docs/Web/JavaScript>
- https://developer.mozilla.org/en-US/docs/Web/JavaScript/New_in_JavaScript