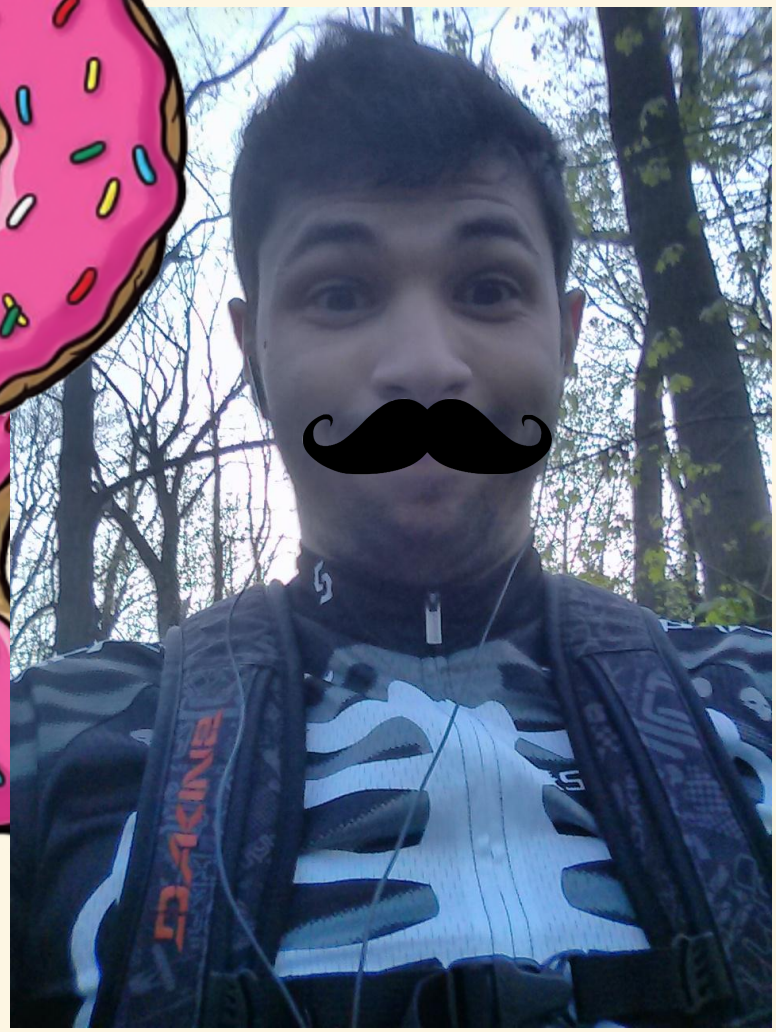
JavaScript



Bozhidar Dryanovski

bdryanovski

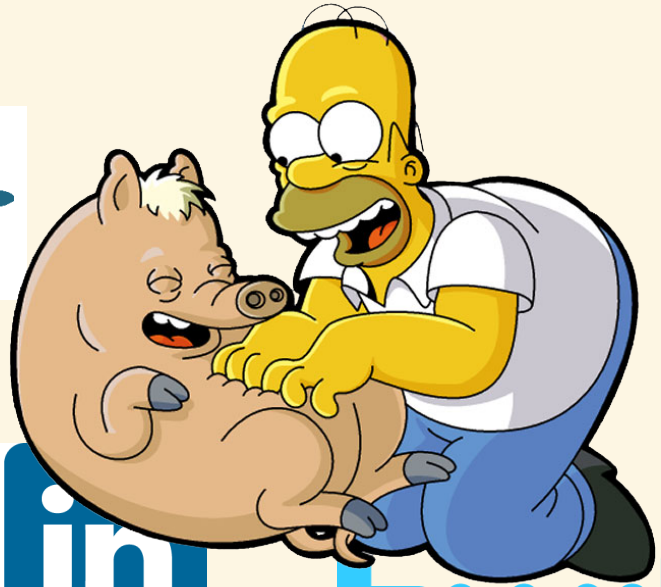
@bdryanovski

bozhidar.dryanovski@gmail.com

Atlassian Bitbucket

Gmail by Google

Trello



LinkedIn

twitter

foursquare

facebook

JavaScript - is everywhere

- Browser
- Desktop
- Server side
- Phones and Tablets

JavaScript != Java

- Functional Language
- Run from top to bottom
- It's all mutable objects
- Prototype based
- Almost everything is **object**

JavaScript built-in types

- Object.prototype
- Array.prototype
- Function.prototype
- Number.prototype
- String.prototype
- Boolean.prototype
- null
- undefined

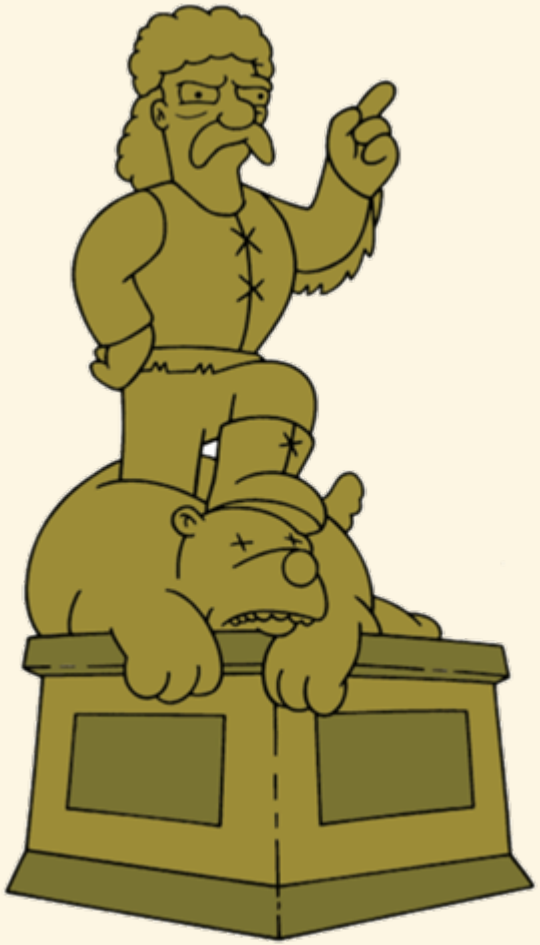
JavaScript - Variables

- Any **var** which is not properly declared is assumed to be global by default
- case sensitive
- can also begin with $ and _

```
var awesome = 'me'
```

Number.prototype

```
// Numbers are 64 bit
var my_number = 1
var your_number = 1.45
// There is a Not a Number
var number = NaN
// is not equal even to itself
number == number // false
// But it's a number
typeof number    // 'number'
```



Number.prototype

```
// But some time sucks at math!

0.1 + 0.2

=> 0.30000000000000004
```

Number.prototype


```
+"42" == 42

=> true


+"0100" == 100

=> true


+"0f100" == 100

=> NaN
```

String.prototype

```
var txt = new String("string");
```

```
var txt = "string";
```

Boolean.prototype

False

```
false

null

undefined

""

0

NaN
```

True

```
true

1

"0"

"false"
```

null && undefined

```
var something

typeof something

=> 'undefined'
```

Object.prototype

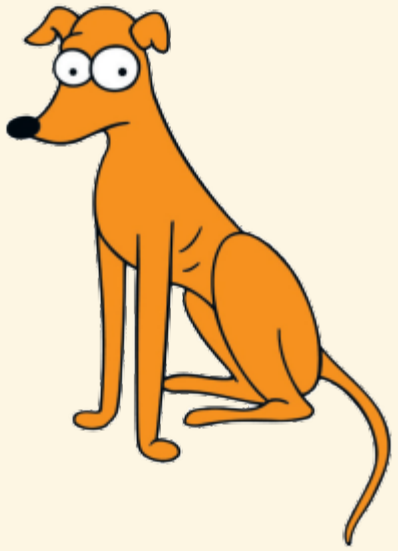
```
{} instenceof Object // true
[] instenceof Object // true
(function() {}) instenceof Object // true
var obj = {
  me_string: 'This is me the String',
  me_number: 1,
  me_array: [1,2,3,4,5,6],
  me_func: function() {
    console.log('Yap, I\'m a function')
  }
}
obj.me_array[2] //  3
obj.me_func()   // 'Yap, I'm a function'
```

Object.prototype

```
// Creating Object
var obj_1 = new Object()

var obj_2 = {}

var obj_3 = Object(Object.prototype)
```



Object.prototype

- Objects can be modified and linked
- Use 'dot' notation or subscript notation to access method/value
- No copy method
- No equal method
- delete object['method/value']

Array.prototype

```
Array instanceof Object

var my_array = [ 'a', 'b', 'c' ]

[ 'a', 'b', 'c' ].length

my_array[2] // 'c'
```

Array.prototype

```
delete my_array[1]
=> ['a', 'undefined', 'c']

Array.prototype.remove = function(from, to) {
  var rest = this.slice((to || from) + 1 || this.
length);
  this.length = from < 0 ? this.length + from : from;
  return this.push.apply(this, rest);
};
```

Function.prototype

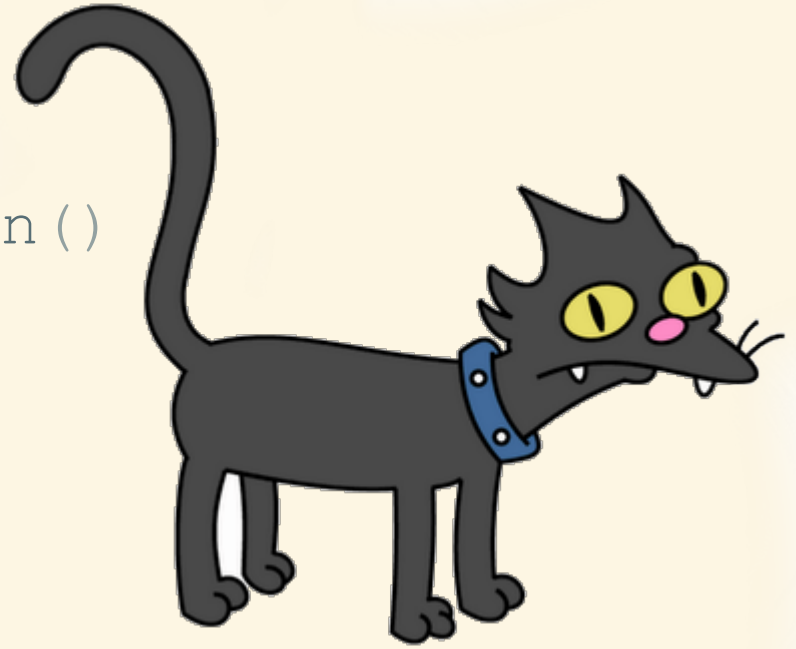
```
function doSomething() {
    /* body */
}
```

```
var doSomething = function(){
    /* body */
}
```

JavaScript Scopes

```
var i = 'What?'
function myFunction() {
  var i = 0
  return function() { console.log(++i) }
}

var iCounter = myFunction()
iCounter() // 1
iCounter() // 2
i          // 'What?'
```



```
jQuery.('a#el').click( function() {
    var self = this

    jQuery.ajax({
      url: jQuery(self).attr('href'),
      data: {}
      success: function(data) { console.log(data) }
    })

    return false
})
```

Function.prototype

```
my_func( arguments ) // this is global

// this is someObject
someObject.my_func( arguments )
someObject['my_func']( arguments )
my_func.apply(someObject,[ arguments ])

// New object is created for this
new my_func( arguments )
```

Function.prototype

- have too many arguments - ignore them
- have less arguments - undefined
- no implicit type checking (default value)

Function.prototype

```
function user ( name, address, company ) {
  var name = name || 'Mickey Mouse'
  var address = address || 'Disney Land'
  var company = company || 'Walt Disney'

  return name + ' is working in ' + company + ' at ' + address
}

console.log( user('Donald Duck') )
```

Extend prototypes

```
Object.prototype.something = function() { }

/* ... */

if ( !Object.hasOwnProperty(something) ) {

    Object.prototype.something = function() { }

}
```

If/else

```
if (condition1) {
  // ...
} else if (condition2) {
  // ...
} else {
  // ...
}
```

for

```
for (var i=0; i < 10 ; i++) {
    if ( i == 3 ) continue
    if ( i == 5 ) break
    console.log( i )
}


// 0, 1, 2, 4
```

labels

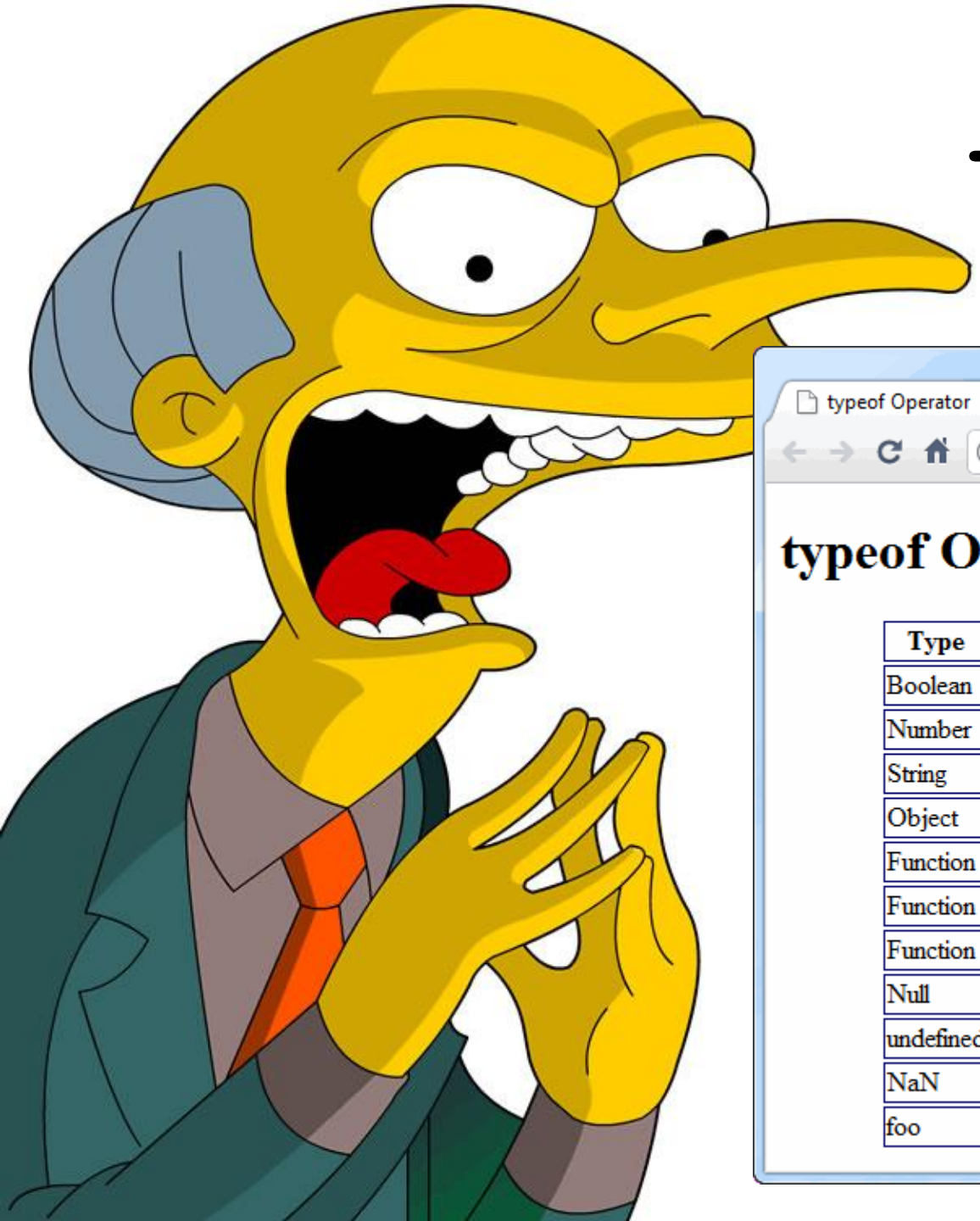
```
counter: for (var i=0; i < 10 ; i++) {
    if ( i == 3 ) continue
    if ( i == 5 ) break counter
    console.log( i )
}

// 0, 1, 2, 4
```

Switch

```
switch(n) {
  case 1:
    // ...
    break;
  case 2:
    // ...
    break;
  default:
    // ...
}
```

typeof



typeof Operator

file:///C:/Webucator/Webuc

typeof Operator

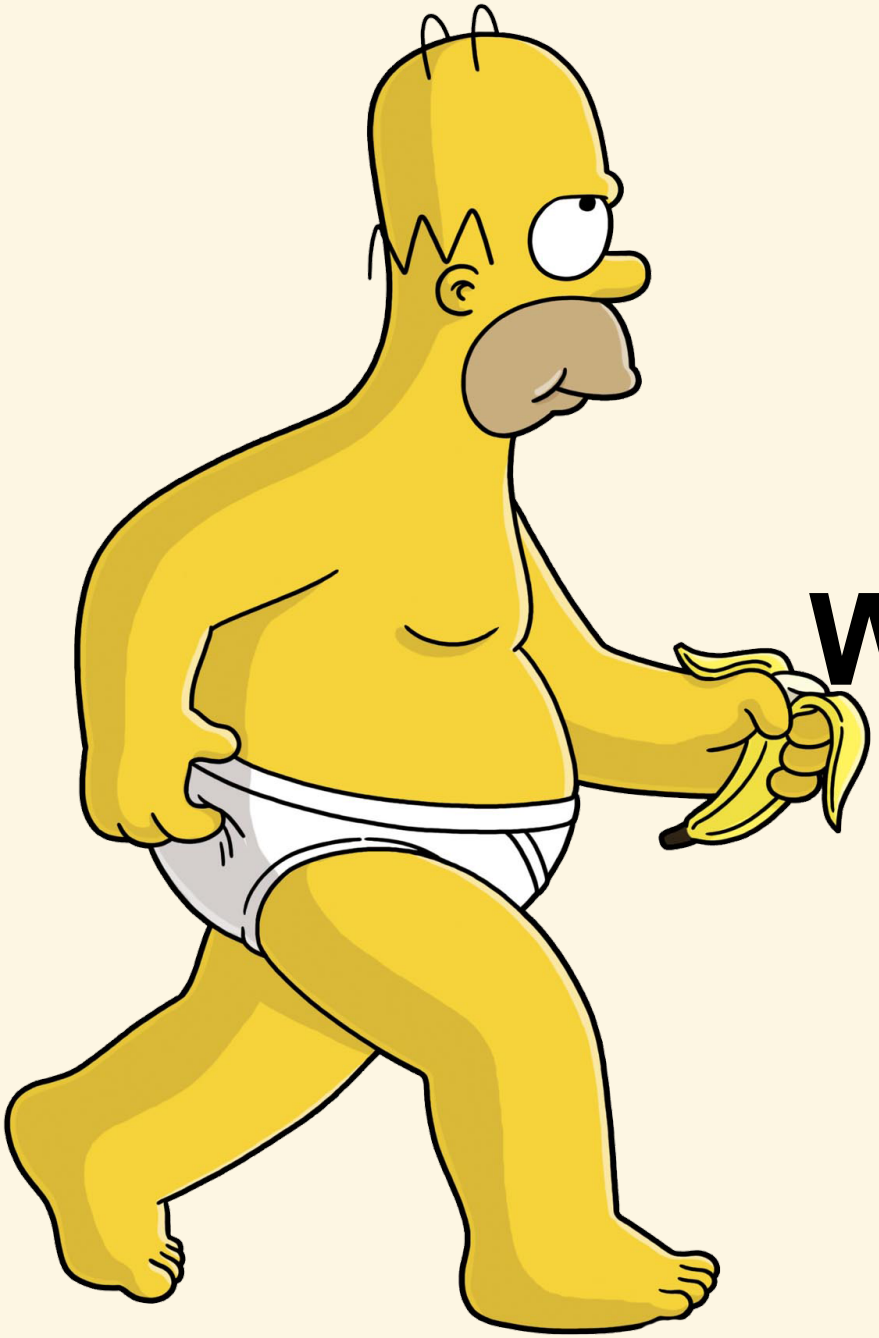
Type	Example	Returns
Boolean	`typeof Boolean()`	boolean
Number	`typeof Number()`	number
String	`typeof String()`	string
Object	`typeof new Array()`	object
Function	`typeof new Function()`	function
Function	`typeof new Date()`	object
Function	`typeof Math`	object
Null	`typeof null`	object
undefined	`typeof undefined`	undefined
NaN	`typeof NaN`	number
foo	`typeof foo`	undefined

Reserved Words

http://mattsnider.com/reserved-words-in-javascript/

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

;



What's next?