JavaScript pradžiamokslis

Gintautas Balčiūnas

gbalciunas@kayak.com

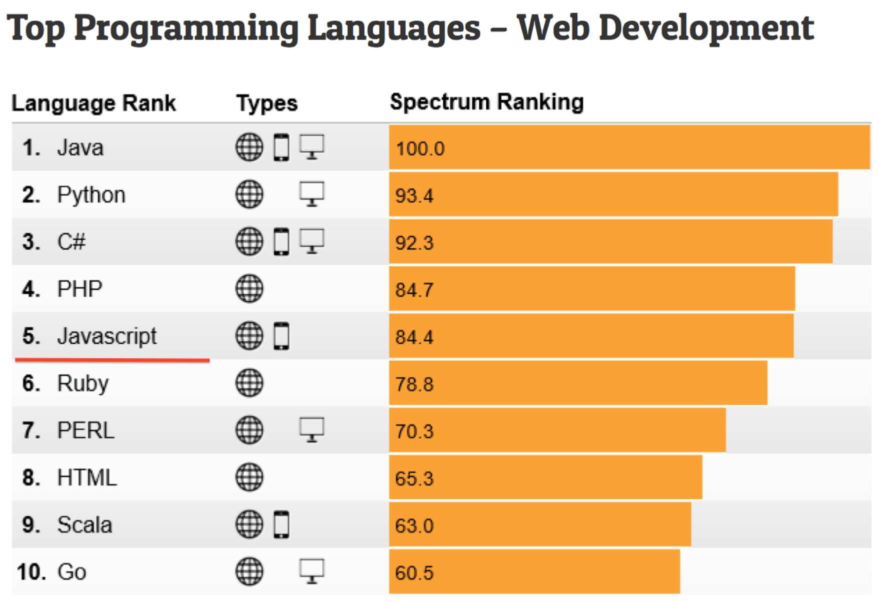
1. HTML
2. CSS
3. **JavaScript**

JavaScript

...there is hardly any website that doesn’t make use of JS.

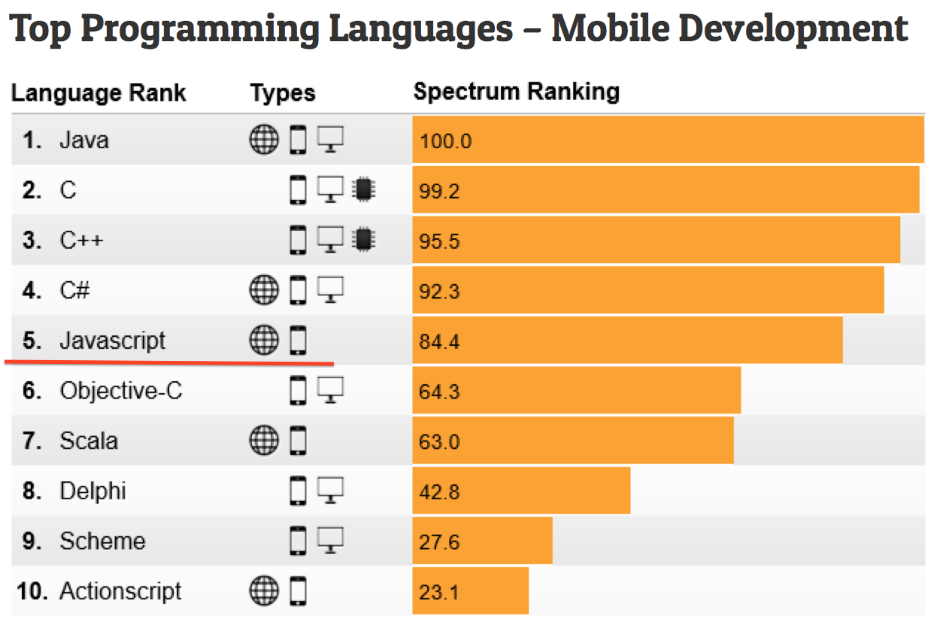
In web development, JS is a must learn language along with some advanced server side language.*

JavaScript



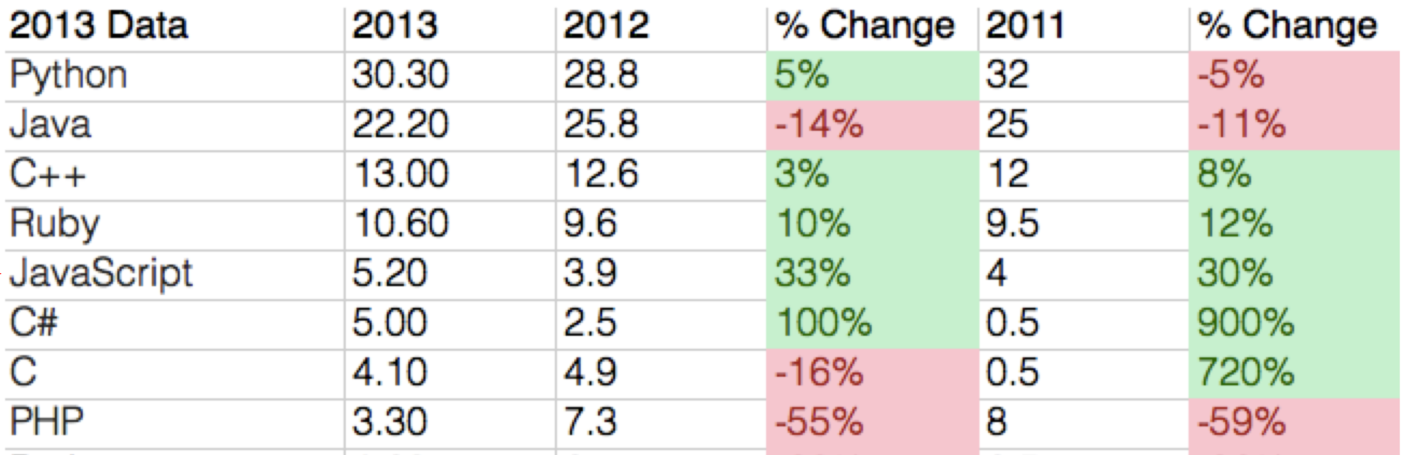
Top Programming Languages – Web Development

Language Rank	Types	Spectrum Ranking
1. Java	🌐 📱 🖥	100.0
2. Python	🌐 🖥	93.4
3. C#	🌐 📱 🖥	92.3
4. PHP	🌐	84.7
5. Javascript	🌐 📱	84.4
6. Ruby	🌐	78.8
7. PERL	🌐 🖥	70.3
8. HTML	🌐	65.3
9. Scala	🌐 📱	63.0
10. Go	🌐 🖥	60.5



Top Programming Languages – Mobile Development

Language Rank	Types	Spectrum Ranking
1. Java	🌐 📱 🖥	100.0
2. C	📱 🖥 ▪	99.2
3. C++	📱 🖥 ▪	95.5
4. C#	🌐 📱 🖥	92.3
5. Javascript	🌐 📱	84.4
6. Objective-C	📱 🖥	64.3
7. Scala	🌐 📱	63.0
8. Delphi	📱 🖥	42.8
9. Scheme	📱 🖥	27.6
10. Actionscript	🌐 📱	23.1



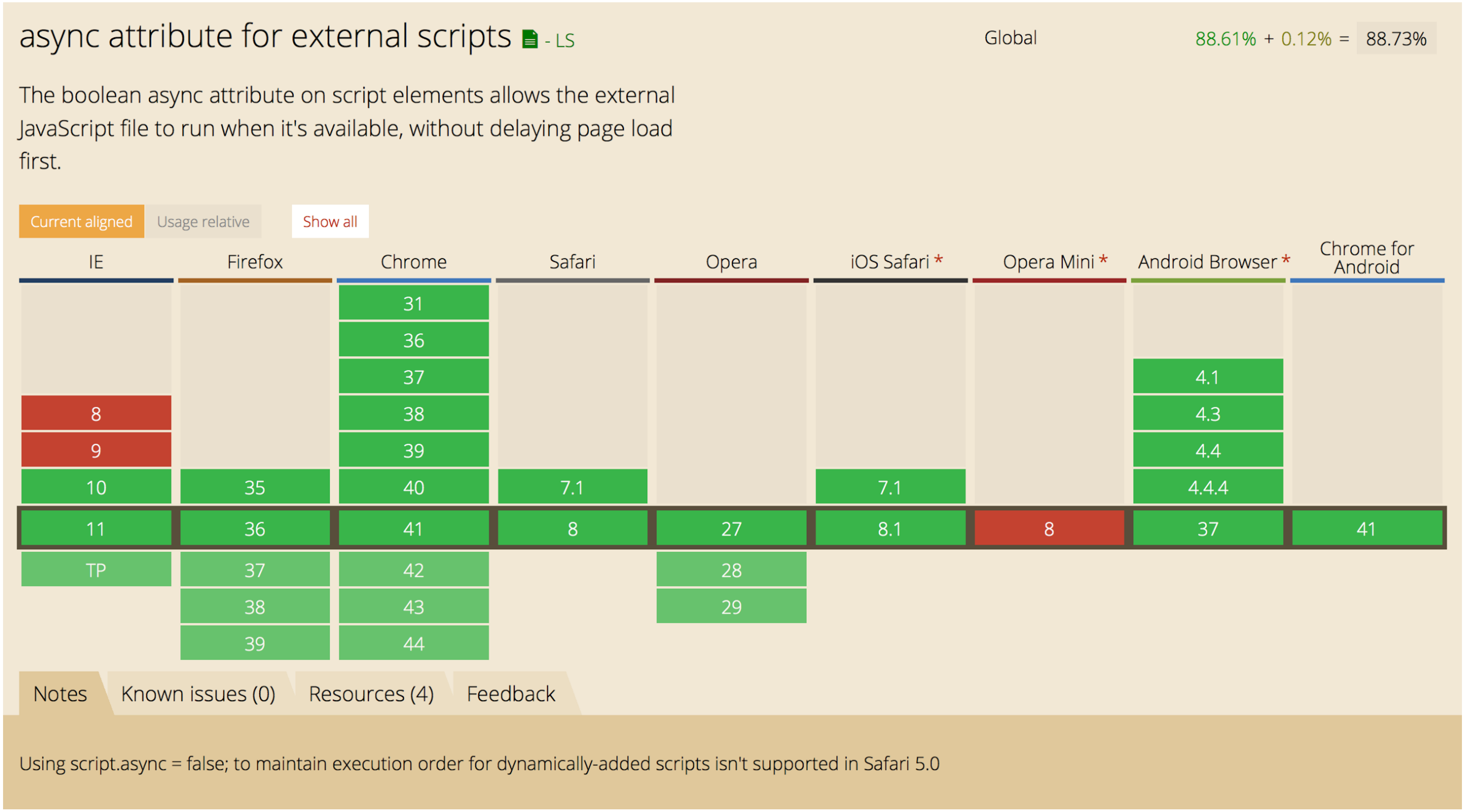
2013 Data	2013	2012	% Change	2011	% Change
Python	30.30	28.8	5%	32	-5%
Java	22.20	25.8	-14%	25	-11%
C++	13.00	12.6	3%	12	8%
Ruby	10.60	9.6	10%	9.5	12%
JavaScript	5.20	3.9	33%	4	30%
C#	5.00	2.5	100%	0.5	900%
C	4.10	4.9	-16%	0.5	720%
PHP	3.30	7.3	-55%	8	-59%

JavaScript

?

```
<script async src=“path/to/script.js“></script>
</body>
</html>
```

http://caniuse.com/#feat=script-async



async attribute for external scripts - LS

Global 88.61% + 0.12% = 88.73%

The boolean async attribute on script elements allows the external JavaScript file to run when it's available, without delaying page load first.

IE	Firefox	Chrome	Safari	Opera	iOS Safari *	Opera Mini *	Android Browser *	Chrome for Android
		31						
		36						
		37					4.1	
8		38					4.3	
9		39					4.4	
10	35	40	7.1		7.1		4.4.4	
11	36	41	8	27	8.1	8	37	41
TP	37	42		28				
	38	43		29				
	39	44						

Using script.async = false; to maintain execution order for dynamically-added scripts isn't supported in Safari 5.0

http://caniuse.com/#feat=script-async

JavaScript data types

- Number
- String
- Boolean
- Object
  - Function
  - Array
  - Date
- Null
- Undefined

JS numbers

2.1 + 5.1 ?

JS numbers

- 2.1 + 5.1 ? = 7.199999999999999
- 0.1 * 0.2 ? = 0.020000000000000004

JavaScript Numbers are Always 64-bit Floating Point

⇩

(2.1 * 10 + 5.1 * 10) / 10

arba ne naudoti kablelio

JS numbers

```
parseInt("123") or +"123"
123
```

```
parseInt("hello")
NaN
```

```
parseInt("010", 10)      ->    parseInt("010", 2)
10                             2
```

JS numbers

```
isNaN(NaN)          isNaN("15")       isNaN("15aaa")
true                false             true
```

```
+"10.3abc"
NaN

parseInt( "10.3abc" )
10

parseFloat( "10.3abc" )
10.3
```

JS numbers

1/0 ?

JS numbers

```
1/0 = Infinity

-1/0 = -Infinity
```

JS strings (primitive)

property:

```
"text".length                                -> 4
```

JS strings (primitive)

property:

```
"text".length                     -> 4
```

methods:

```
"hello".charAt(0)                 -> h
"hello, world".replace("hello", "goodbye")
                                  -> goodbye, world

"hello".toUpperCase()             -> HELLO
```

`etc...`

JS boolean

```
false, 0, the empty string (""),
NaN, null, undefined                    -> false
```

JS boolean

```
false, 0, the empty string (""),
NaN, null, undefined                  -> false


all other values                      -> true
```

JS boolean

```
false, 0, the empty string (""),
NaN, null, undefined                  -> false


all other values                      -> true
```

Test in console:

```
Boolean("")                           -> false
Boolean(234)                          -> true
```

JS variables

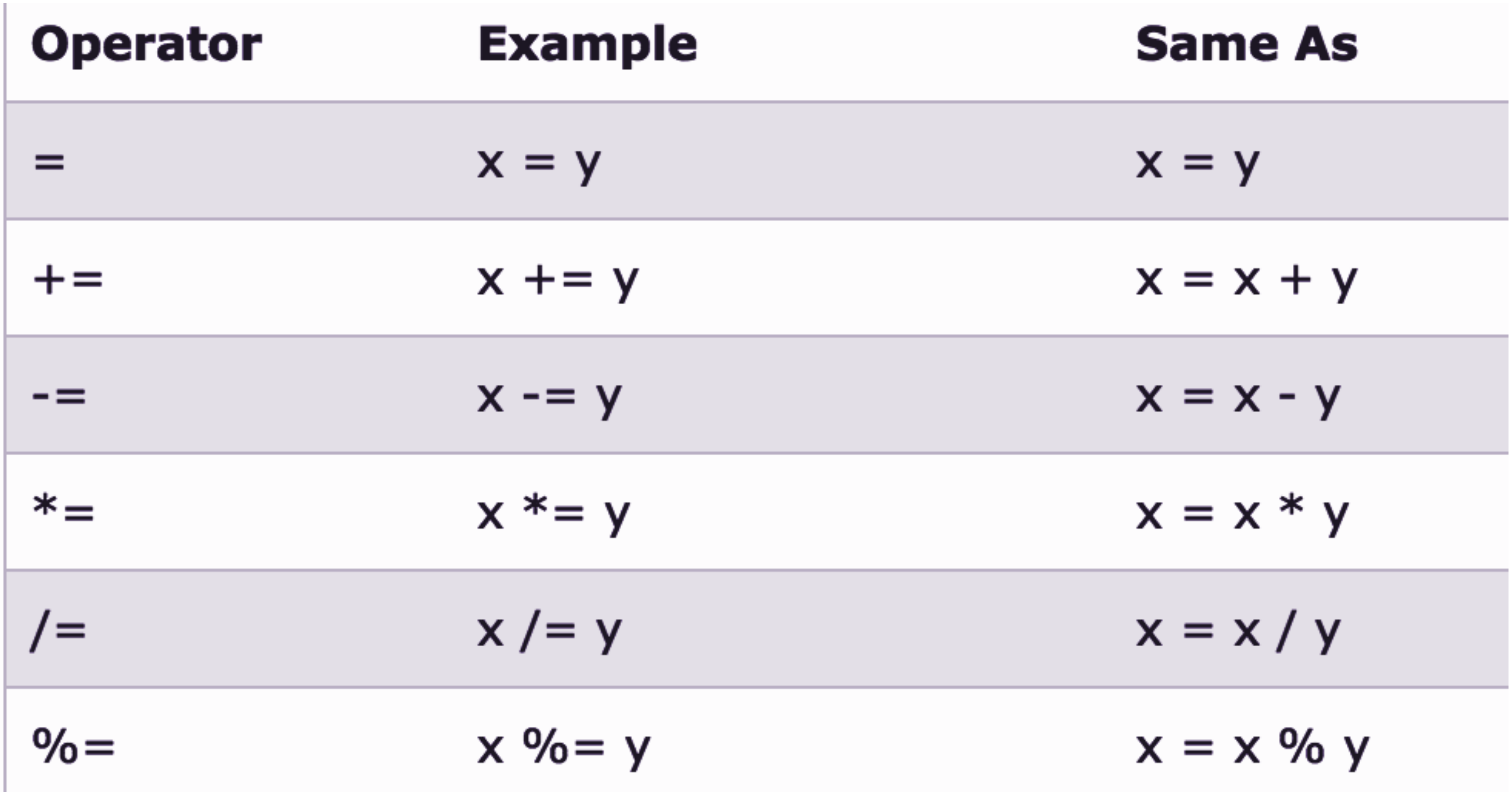
```
var a; alert(a); -> undefined

var name = "Tom"; alert(name);
```

JS operators

+   -   *   /   %   ++   - -

JS assignment operators



Operator	Example	Same As
=	x = y	x = y
+=	x += y	x = x + y
-=	x -= y	x = x - y
*=	x *= y	x = x * y
/=	x /= y	x = x / y
%=	x %= y	x = x % y

JS string operators

```
> "hello" + " world"
hello world
```

```
> "3" + 4 + 5
"345"
> 3 + 4 + "5"
"75"
```

JS comparisons

```
> "cat" == "cat"
true
> 1 == true
true
```

```
> 1 === true
false
> true === true
true
```

JS control structures

```
var name = "kittens";

if (name == "puppies") {
    name += "!";
} else if (name == "kittens") {
    name += "!!";
} else {
    name = "!" + name;
}
-> "kittens!!"
```

JS control structures

```
while (true) {
    // an infinite loop!
}
```

```
var text = ""; var i = 1;
do {
    text += " The number is " + i;
    i++;
} while (i < 10);

console.log(text);
```

JS control structures

```
for (var i = 0; i < 5; i++) {
    console.log(i);
}
```

```
var name = otherName || "default";
```

```
var allowed = (age > 18) ? "yes" :
"no";
```

JS objects

```
var obj = new Object();
var obj = {};
```

```
function Person(name, age) {
  this.name = name;
  this.age = age;
}

var you = new Person("YourName", 42);
```

JS objects

```
you.name = "Tom";
var name = you.name;
```

```
you["name"] = "Tom";
var name = you["name"];
```

JS objects (Accessing Nested Properties)

```
var obj = {
    name: "house",
    details: {
        color: "green",
        size: 120
    }
}
```

```
> obj.details.size -> 120
```

JS arrays

```
var a = new Array();
a[0] = "house";
a[1] = "car";
a[2] = "tree";
> a.length -> 3
```

```
var a = ["house", "car", "tree"];
> a.length -> 3
```

JS arrays

```
var b = ["house", "car", "tree"];
b[100] = "bike";


> b.length -> ?
```

JS arrays

```
var b = ["house", "car", "tree"];
b[100] = "bike";
```

```
> b.length -> 101
```

?!

JS arrays

```
for (var i=0; i<b.length; i++) {
    console.log(b[i])
}
```

```
house

car

tree

(97) undefined ←!!!!

bike
```

JS arrays

```
for (var i in b) {
    console.log(b[i])
}
```

```
house

car

tree

bike
```

JS scope (function-level scoping)

```
function doStuff(condition) {
   if (condition === true) {
      var x = 'hello';
   }
   alert(x);
}

doStuff(true);
-> hello
```

JS scope (global-level scoping)

```
function doStuff(condition) {
    if (condition === true) {
      ~~var~~ x = 'hello';
    }
}
doStuff(true);
alert(x);
-> hello
```

JS functions (arguments)

```
function add(x, y) {
    var total = x + y;
    return total;
}
```

```
> add()
NaN
> add(2, 3, 4)
5
```

JS functions (arguments object)

```
function avg() {
    var sum = 0;
    for (var i in arguments) {
        sum += arguments[i];
    }
    return sum / arguments.length;
}
```

```
> avg(2, 3, 4, 5)
3.5
```

JS anonymous functions

```
var anon = function() {
    alert("I am anonymous");
};


function test() {...}
```

JS anonymous functions

```
flyToTheMoon1(); //OK
function flyToTheMoon1() {
      alert("Zoom! Zoom! Zoom!");
}
flyToTheMoon1(); //OK
```

JS anonymous functions

```
flyToTheMoon1(); //OK
function flyToTheMoon1() {
      alert("Zoom! Zoom! Zoom!");
}
flyToTheMoon1(); //OK
```

```
flyToTheMoon2(); //undefined is not a
function
var flyToTheMoon2 = function () {
      alert("Zoom! Zoom! Zoom!");
}
flyToTheMoon2(); //OK
```

JS custom objects

```
function Person(first, last) {
    this.first = first;
    this.last = last;
    this.getFullName = function() {
        return this.first + ' ' + this.last;
    };
}

var s = new Person("Tom", "Cat");
```

JS custom objects (optimized)

```
function Person(first, last) {
    this.first = first;
    this.last = last;
}

Person.prototype.getFullName = function() {
    return this.first + ' ' + this.last;
};
```

JS inner functions (scope)

```
function betterExampleNeeded() {
    var a = 1;
    function plusTwo() {
        return a + 2;
    }
    return plusTwo();
}

> 3
```

JS closures

```
function makeAdder(a) {
    return function(b) {
        return a + b;
    };
}
adderX = makeAdder(5);
adderY = makeAdder(20);


adderX(6) ←?
adderY(7) ←?
```

Lets
Play

bit.ly/**zaidimas11**

4:46

bit.ly/**manau**