JavaScript

03.03.2016

Who am I?

- **Andrej Skok**

  andrejskoki@gmail.com
- **Vacuumlabs**

  Front-end developer, JavaScript (ReactJS, ReactNative), backend NodeJS, Python
- **FMFI UK, computer science**

What is JavaScript

- Interpreted OOP language with strong functional background
- ECMA script standard
- From 2011 ECMA5
- From 2015 ECMA6
- New version every year...

Why JavaScript?

- In every browser
- Only client side language supported in browser
- Managing what is happening on the screen
- From 2011 => NodeJS, React, Angular
- Mobile platforms
- CoffeeScript, TypeScript, PureScript

Power of JS

- Manipulating DOM
- Client side validaton and effects
- Async data fetching
- Single-Page
- Ananlytics

Whats wrong with JS?

- Community solving many problems
- Biggest issue

WHAT IS WRONG WITH MY PROGRAM?

How to run JS in browser?

```
<script>
    document.write(“Hello world”)
</script>
```

Example time!

Syntax (1)

- Case-sensitive, Unicode
- Statements delimited with semicolon
- Semicolon are not mandatory
- … but recommended

Syntax (2)

- // comment
- /* comment */

Declaration of variables

- var a = 5;
- Dont care about types
- Function-scoped (or global)
- Hoisting
- Always declare variables at the top of their scope

Example time!

Types

- **6 types:**
- **Boolean**
- **Null**
- **Undefined**
- **Number**
- **String**

+ object (functions, arrays...)

- **Dynamic types**

  var a = 42;

  a = 'Ahoj';
- **Wrapping**

  var a = 5; // primitive

  var a = new Number(5); // object
- **Everything acts like object**

Object

- **“An object is a collection of properties, and a property is an association between a name (or key) and a value.”**
- **Set of key/value pairs**
- **var a = {count: 4 , title: “JavaScript”}**

  console.log(a.count) // 4

Number type

- **64-bit floating point double**
- **1 type**
  - 0, 117 and -345 (decimal, base 10)
  - 015, 0001 and -0o77 (octal, base 8)
  - 0x1123, 0x00111 and -0xF1A7 (hexadecimal, "hex" or base 16)
  - 0b11, 0b0011 and -0b11 (binary, base 2)
  - Infinity, -Infinity, NaN

Little bit of Math

Math object .abs(), .floor(), .max(), .min(), .random(), .sin(), .cos() …

Math.abs(-5); //5

Why not?

number NaN

12 + NaN; // NaN

NaN + NaN; // false

- **Floating-point:**
  - -.123456789
  - -3.1E+12
  - .1e-23
  - 3.141592
- **Exactly:**

  [(+|-)][digits][.digits][(E|e)[(+|-)]digits]
- **Fun with numbers example!**

Strings

- 16-bit Unicode
- No char
- var a = “hello”;
- Immutable

- Reference:

  https://developer.mozilla.org/en-US/docs/Web/JavaScript/Refere

Arrays

- var a = [0, 1, 'hi'];
- var a = new Array(length);
- var a = new Array(0, 1, 2);
- Methods:

  pop(), push(), reverse(), shift(), sort(), splice()

  …

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

T/F

Eval as False

- False , null, undefied, empty string, 0, NaN

Everything else is True

- [], {} ...

Operators

=== vs. == (!== vs. !=)

=== porovnáva aj typ

```
3 == '3'  // true
3 === '3' // false
```

Recommended to use always ===

+, -, <, >, &&, ||, &, |

Little hacking with && and ||

```
var tmp = myPrettyObject || {};
var tmp = thisIsTrue && whatever; // whatever
```

Expression equals the last value which was being evaluated!

Ternary operator

Always use ternary operator when possible!

var b = expression ? this: that;

Control structures

- **Like Java, PHP etc.**
- **If-else, while, for, switch**

```
for (var i=0; i < 10; i++) {
    ...
}
```

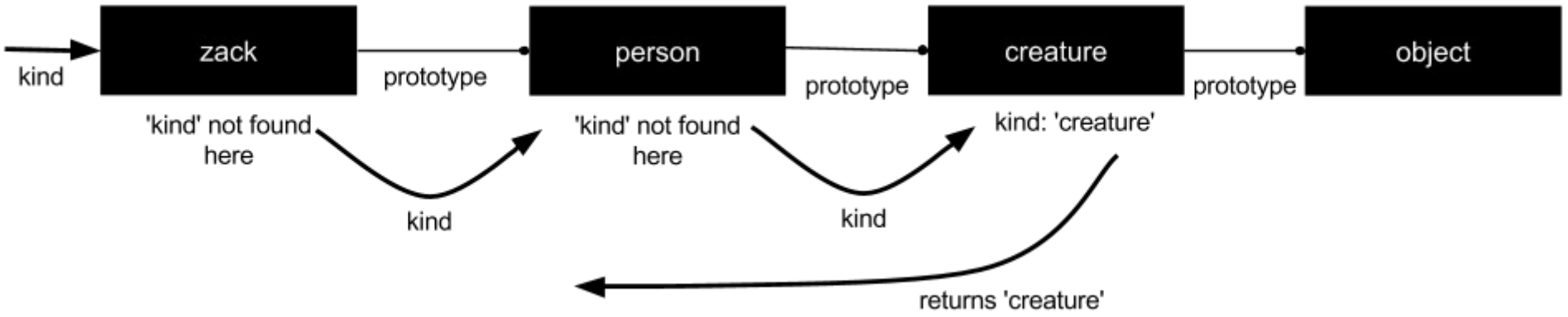
var in 'var i = 0' super important!

What about OOP?

- No classes
- Just objects
- Prototype-based language
- Polymorphism for free!
- Inheritance on the level of objects

Prototypes

- Every object has a prototype (internal link)
- Query an object, looking through chain
- Single parent inheritance
- Long chains



The __proto__ property

Let's do some coding!

Lets add function to all objects!

Lets add function to all types!

Do whatever you want!

(then debug all day)

Objects vol. 2

How to create objects?

var obj = {};

var obj = Object.create(prototype);

var obj = new Object(); ???

No Problem with encapsulation

Polymorphism for free

Inheritance by chains

Constructor function

- engines are highly optimized for constructor functions
- Function as constructor
- Function called with new keyword magically creates object
- Let me show you the magic!

Objects vol.3, usefull stuff

. notation or [] notation

for (key in object)

borrowing functions, Function.call Function.apply

Object.defineProperty → writable, configurable, enumerable

Object .keys, .preventExtensions, .freeze, . isFrozen

What does 'this' mean?

- **Problems with 'this'**

  Example time!
- **3 meanings**

  - object method, refers to object

  - constructor, refers to new object

  - standalone function → global object

Functions

- First-class citizen
- Arguments as many as you like
- arguments variable inside function
- 2 types, declarations and expressions
- Seems like nothing but there is a difference
- Higher-order functions

Modules

- **Have everything global is not a good idea**
- **Example time for module management!**

DOM manipulations

Objects in the DOM tree may be addressed and manipulated by using methods on the objects

The good /*pun intended*/ old way

```
document.getElementById("content");

document.getElementsByTagName("p");

document.getElementsByClassName("awesomeClass");

document.querySelectorAll("p.intro");
```

& chain them as you wish

Html collections

```
var x = document.forms["form1"];
```

```
document.images .link .scripts….
```

I want change!

element.innerHTML

element.attribute (.src, .href) // .setAttribute

element.style.property

document.createElement(element)

document.removeChild(element)

document.appendChild(element)

document.replaceChild(element)

Events

- Let's listen to user
- .addEventListener(event, callback);
- Events = {mouseover, focus, blur, hashchange, click, ….}
- Callback = function(event) { /*do something*/}

Exercise 1

Web calculator

+ - * / power = C

Power function callable on Number function

Memory function if you are bored?

Exercise 2

- TODO list
- Input box with name of TODO
- Add todo to list, delete, or mark as completed

General tips

- Don't use excessive comments
- Lines of code
- Beware of tight coupling
- Not too many variables per function, use more functions when needed
- Arguments should be logically grouped
- Don't do deep nesting