Программирование на JavaScript

Лекция 1
О курсе. Введение.

Hexlet University

Знакомство

Рахим Давлеткалиев
http://freetonik.com

rakhim@hexlet.org
@freetonik

JS в “двух” словах

- LISP/Scheme (почти) с C-синтаксисом (почти)
- нестрогая типизация
- прототипно-ориентированный
- сценарный (scripting)
- чаще всего используется в браузерах
- *«Самый неправильно понятый язык программирования в мире стал самым популярным в мире языком программирования»* – Дуглас Крокфорд

Чего нет в этом курсе

- Работа с браузером, DOM, HTML
- Эффекты, анимация, jQuery, библиотеки, фреймворки
- AJAX, взаимодействие с сервером, node.js
- Отладка, тестирование, среда разработки

Эмм... а что тогда есть?

только **язык**:

- Грамматика
- Объекты, прототипы, наследование
- Регулярные выражения
- Замыкания
- Стиль
- Красивые и отвратительные особенности

Зачем?
Для кого этот курс?

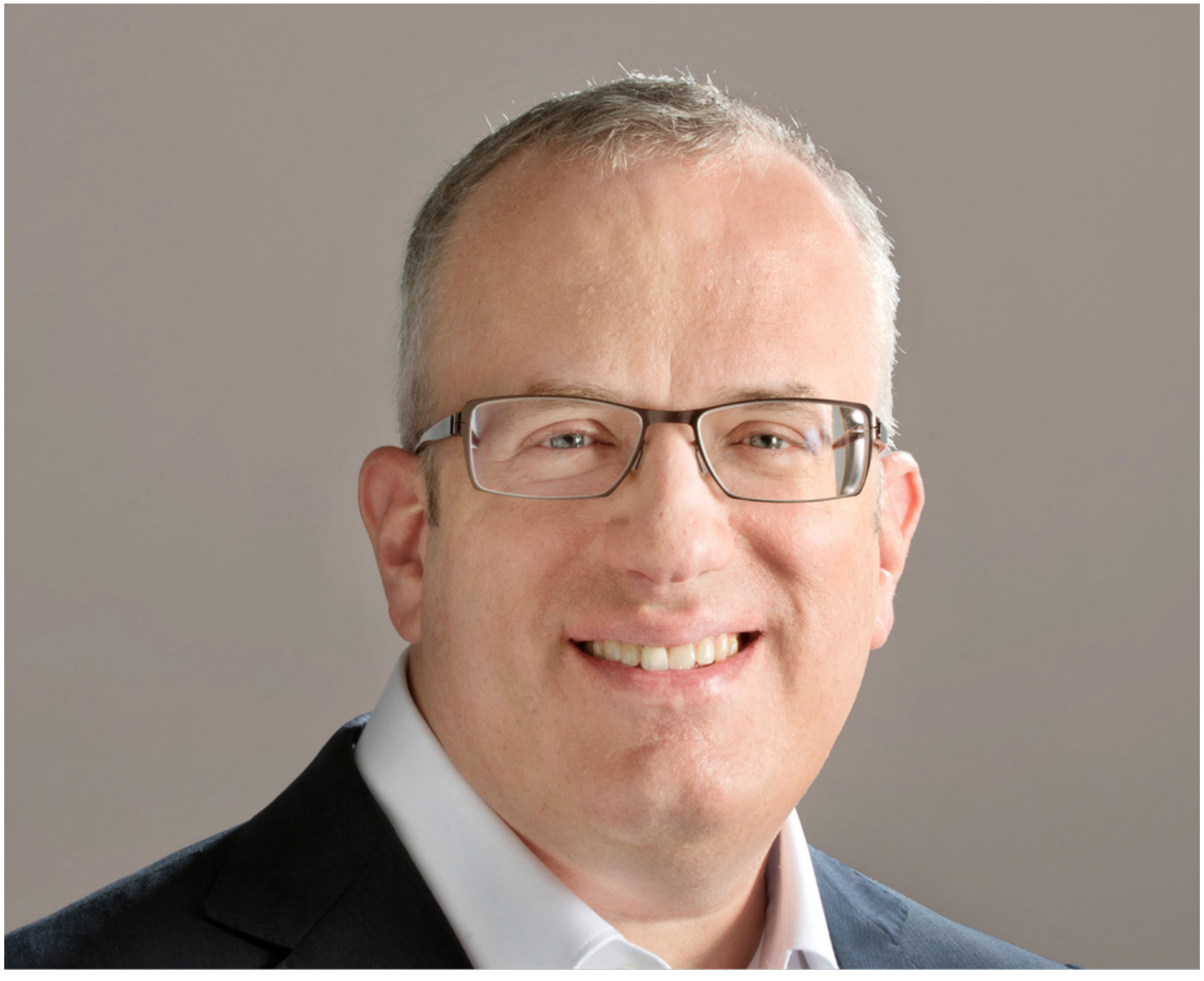
- JS ужасен, но на самом деле нет
- Все дело в применении
- JS прощает плохие вещи, он слишком добрый

На чем основан курс?

- Стандарт ECMA-262
- Лекции Дугласа Крокфорда
- “JavaScript: The Good Parts” Дугласа Крокфорда
- Блоги хороших программистов

Немного истории

Брендан Айк
главный инженер Mozilla Corporation



Пример того, что нам предстоит узнать

```
function foo() {
    return
    {
        foo: 'bar'
    }
}

function bar() {
    return {
        foo: 'bar'
    }
}
```

Пример того, что нам предстоит узнать

```
1 == 1; //true
'foo' == 'foo'; //true
[1,2,3] == [1,2,3]; //false
```

```
new Array(3) == ",,"; //true
new Array(3).toString(); //",,"
```

```
new Array(3) === ",,"; //false
```

```
[+!+[]]+([]+([]+[])[([]+{})[+!+[]+!+[]+!+[]+!+[]+!+[]]+([]+{})[+!+[]]+([]+[][+!+[]])[+!+[]]]
[]]+([
```

Зарезервированные имена

- break
- case
- catch
- continue
- debugger
- default
- delete
- do
- else
- finally
- for
- function
- if
- in
- instanceof
- new
- return
- switch
- this
- throw
- try
- typeof
- var
- void
- while
- with

- class
- enum
- export
- extends
- import
- super

Числа

- Только один тип (float64, 8 байт с плавающей точкой)
- 0.1+0.2 == 0.3; // false
- 1 == 1.0

Ошибочные числа

1/0 = Infinity

-1/0 = -Infinity

NaN = не числовое значение

1. Любая операция с NaN дает NaN
2. NaN != NaN
3. isNaN(...)

Некоторые удобные функции

- `Number(10);` `// 10`
- `Number("42.23");` `// 42.23`
- `Number("71oshi");` `// NaN`
- `parseInt("18");` `// 18`
- `parseInt("19kdjas");` `// 19`
- `parseInt("74.54");` `// 74`
- `parseFloat("74.54");` `// 74.54`

parseInt(num, base)

```
parseInt("ff");              //NaN
parseInt("ff","16");         //255

parseInt(“0x10”);            // 16
parseInt(“0x10”,”10”); // 0
```

parseFloat(num)

Преобразования

```
var x = 123456789;          // undefined
x.toExponential();          //'1.23456789e+8'
x.toExponential(1);         //'1.2e+8'
x.toExponential(2);         //'1.23e+8'
x.toExponential(3);         //'1.235e+8'
```

Преобразования

```
var y = 43.81327;

y.toFixed();        //'44'
y.toFixed(1);       //'43.8'
y.toFixed(2);       //'43.81'
y.toFixed(3);       //'43.813'
```

Битовые операции

```
var n = 7432;
n.toString();       // '7432'
n.toString(2);      //'1110100001000'
```

JavaScript берет целую часть числа и конвертирует
в необходимое количество битов, например:
0 – два бита (знак + 0)
1 – два бита
2 – три бита

Operator	Description
&	AND
|	OR
^	XOR
~	NOT
<<	Shift Left
>>	Shift Right
>>>	Shift Right

```
var a = 5;
var b = 13;

// a | b - OR
a|b;   // 13

// a & b - AND
a&b;   // 5

// a ^ b - XOR
a^b;   // 8

// ~a - NOT
~a;    // -6

// a >> b - RIGHT SHIFT
a>>b;  // 0

// a << b - LEFT SHIFT
a<<b;   // 40960

// a >>> b - ZERO FILLED RIGHT SHIFT
a>>>b;  // 0
```

Округление

```
3.257582347 | 0;      // 3
~~3.257582347;         // 2
```