

Let's Talk JavaScript



Agenda

- ✳ JavaScript Is Awesome
- ✳ Simple Programs
- ✳ JavaScript Functions
- ✳ Arrays and Objects

JavaScript Is Awesome

- * It's everywhere
- * It survives
- * Everyone's doing it
- * It's great at parties
- * It's amazingly simple

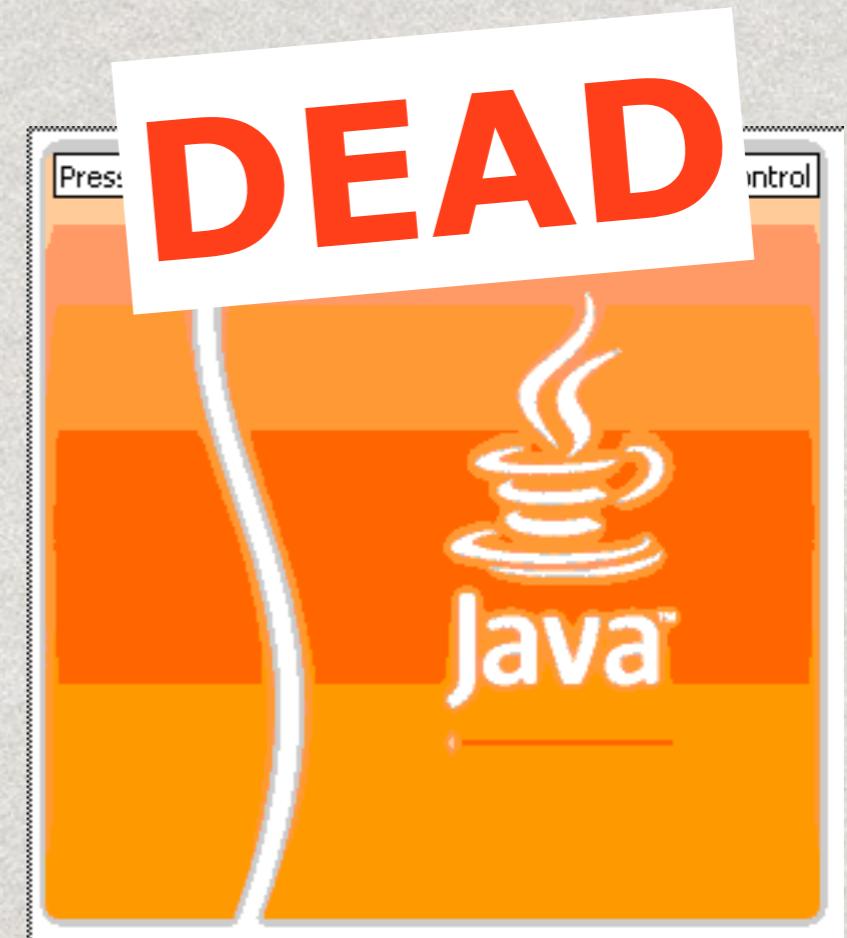


JS Is Everywhere



JS Survives

- * Around since 1995
- * Outlived Java Applets



A Short History

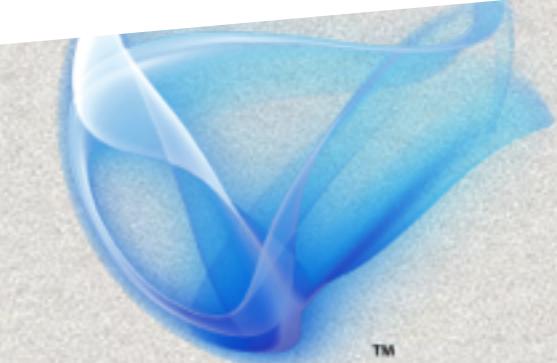
- * 1995 Brendan Eich started developing a new language for Netscape Navigator 2.0
- * Original name was LiveScript
- * Announced on Dec 1995 as JavaScript
- * 1996 Microsoft responded with JScript



JS Survives

- * Around since 1995
- * Outlived Silverlight

DEAD



Microsoft®
Silverlight™

JS Survives

- * Around since 1995
- * Outlived Flash



Everyone's There

- * No need to reinvent the wheel
- * Plenty of libraries
- * Plenty of docs

JavaScript's Really Simple

- ✳ Few key concepts
- ✳ Minimalistic Syntax
- ✳ Top-to-Bottom semantics

JavaScript Key Ideas

- * Interpreter based (no compilation)
- * Loosely typed language
- * Objects are just hash tables

JavaScript Key Ideas

- * Interpreter based (no compilation)
- * Loosely typed language
- * Objects are just hash tables

JavaScript Key Ideas

- * Interpreter based (no compilation)
- * Loosely typed language
- * Objects are just hash tables

Q & A



Running JavaScript

- * Easy way: jsbin
- * Completely offline using node.js
- * Using a simple HTML file

Demo

JavaScript Basic Syntax



JavaScript Syntax

- * Defining variables
- * Loops and branches
- * Defining functions

JavaScript tl;dr

- * Use **var** to declare a variable
- * Use MDN to find information
- * Everything else works
- * Lab: <http://ynonperek.com/javascript-exer.html>

JavaScript Core Types

- * Numbers
- * Strings
- * Booleans
- * null
- * undefined
- * Objects

Numbers

- * JavaScript has only one number type called number
- * number is a 64-bit floating point (double)
- * Same arithmetical problems double have ($0.1+0.2 \neq 0.3$)
- * A special value NaN represents errors

Numeric Functions

- * `Number(string)`
 - converts string to number
 - returns NaN on error
- * `parseInt(string, radix)`
 - converts string to number
 - tries its best (so '7hello' is OK)
- * `Math.random()`
 - returns a random between 0 and 1

Numeric Functions

```
x = 3;  
y = Number(7);  
z = Number('9');  
console.log(x + y + z);
```

Q & A

- * Numbers
- * Strings
- * Booleans
- * null
- * undefined
- * Objects

Strings

- * Strings are unicode 16-bit chars (like in Java).
- * No char class. Characters are strings of length 1
- * Strings are immutable (like in Java)
- * Both Single and Double quotes make a string

String Examples

```
'hello'.length      === 5
```

```
String(10).length      === 2
```

```
'hello'.index0f('l')      === 2
```

```
'hello'.lastIndex0f('l') === 3
```

```
'hello'.toUpperCase()      === 'HELLO'
```

Q & A

- * Numbers
- * Strings
- * Booleans
- * null
- * undefined
- * Objects

Boolean Type

- * JavaScript supports ‘true’ and ‘false’ as boolean values
- * Boolean(value) is a function returning the truthness of a value
- * returns false if value is falsy, and true if value is truthy
- * !!value has the same meaning

null

- * represents the "nothing" of JavaScript
- * usually used to mark errors
- * JavaScript will not give null to a variable. It's always the result of an assignment performed on purpose

undefined

- * Not even the nothing
- * JavaScript puts undefined in anything that hasn't yet been assigned a value

JavaScript False/True

- * These are all falsy:
 - * `false`, `null`, `undefined`
 - * `""` (the empty string)
 - * `0`, `NaN`
- * Everything else is truthy

Objects & Arrays



Objects

- * Everything else is an object
- * An object is a collection of key/value pairs.
- * Objects are fully dynamic, so new methods and fields can be added at runtime
- * Objects have no classes
- * Each object has a prototype. We'll talk about that later.

Objects

name	Ynon Perek
email	<u>ynonperek@yahoo.com</u>
web	<u>http://ynonperek.com</u>

```
var me = {  
    name : 'Ynon Perek',  
    email : 'ynonperek@yahoo.com',  
    web : 'http://ynonperek.com'  
};
```

Object Functions

```
var o = { x: 5, y: 7 };

delete o.x;

// undefined
console.log( o.x );
```

Object Iteration

```
var o = { x: 5, y: 7 };

for ( var key in o ) {
  if ( ! o.hasOwnProperty(key) ) {
    return;
  }

  console.log(key + ' => ' + o[key] );
}
```

JavaScript Arrays

- * Ordered collections of variables
- * Can hold anything

JavaScript Arrays

```
var arr = [ 1, 3,  
            'foo',  
            { x: 5, y: 7 },  
            1  
        ] ;
```

1

3

‘foo’

{ x: 5, y: 7 }

1

Array Functions

- * Use **push** to add elements to the end
- * Use **pop** to remove elements from the end

```
var a = [1, 2, 3];
a.push('foo');

// now a is [1, 2, 3, 'foo']

a.pop();
a.pop();

// now a is [1, 2]
```

Array Functions

- * Use **shift** to remove the first element
- * Use **unshift** to insert elements in the beginning

```
var a = [10, 20, 30];  
  
a.shift();  
// a is now [20, 30]  
  
a.unshift('foo', 'bar');  
// a is now ['foo', 'bar', 20, 30]
```

Array Splicing



a. `splice(3, 2);`

Array Splicing



a. `splice(3, 2);`



Array Splicing



a. `splice(3, 2);`



Array Subscripting



Array Subscripting

- * Use brackets to subscript
- * Can take string or number

```
arr = [1,2,10];  
  
// prints 1  
console.log(arr[0]);  
  
// prints 10  
console.log(arr[2]);  
  
// prints 10  
console.log(arr['2']);
```

Array Iteration

- * Use **for** loop to iterate an array
- * Note the magic **length** property

```
var arr = [10, 20, 30];  
  
for ( var i=0; i < arr.length; i++ ) {  
    console.log( arr[i] );  
}
```

Arrays Are Just Refs

```
var arr = ['foo', 'bar'];
var b = arr;

b.push('buz');

console.log( arr );
```

Lab: Arrays And Objects

- * In Node.JS the variable `process.argv` holds an array of all command line arguments
- * Write a program that prints the unique arguments passed to it. For example running:
`node uniq.js 10 20 20 20 30 10`
should print:
`10 20 30`

Arrays Combinators

- * `forEach(f)`: calls $f(x)$ for every x in array
- * `every(f)`: returns true if for every x : $f(x)$ is true
- * `some(f)`: returns true if $f(x)$ is true for at least one x in array

Arrays Combinators

- * filter(f):
 - * Calls f(x) for every x in array.
 - * Returns a new array only with the true values

```
var a = [10, 15, 17, 20];
var even = a.filter( function(el) {
  return el % 2 == 0;
});
```

```
console.log( even );
```

```
var primes = a.filter( is_prime );
```

Arrays Combinators

- * map(f)
- * Calls f(x) for every x in array
- * Returns an array of the results

```
var a = [10.8, 20.17, 55.22, 77.1, 2];
var rounded = a.map(function(el) {
  return Math.round(el);
});

// Same as
var alt = a.map( Math.round );

console.log( rounded );
```

Q & A



Working With Functions

Spot The Bug

```
var a = Math.floor( Math.random() * 10000 );
var b = Math.floor( Math.random() * 10000 );
var sum_a = 0, sum_b = 0;

while (a) {
    var next = a % 10;
    sum_a += next;

    a = Math.floor( a / 10 );
}

while (b) {
    var next = a % 10;
    sum_b += next;

    b = Math.floor( b / 10 );
}
```

Why Functions

- * Save us from Copy-Paste bugs
- * Make cleaner code
- * Easy to read => Easy to debug

Functions: How

```
var a = Math.floor( Math.random() * 10000 );
var b = Math.floor( Math.random() * 10000 );

var sum_a = sum_of_digits( a );
var sum_b = sum_of_digits( b );
```

Functions: How

```
function sum_of_digits( number ) {  
    var result = 0;  
  
    while ( number > 0 ) {  
        result += number % 10;  
        number = Math.floor( number / 10 );  
    }  
    return result;  
}
```

It Gets Better

- * Replace function implementation without changing outside program

```
function sum_of_digits( number ) {  
    return Number(String(number).split('')).reduce(  
        function(a,b) { return Number(a) + Number(b); }  
    );  
}
```

Function Variables

- * Each variable is bound to an object
- * Calling a function creates an Activation object, so var inside function is lexical
- * Outside every function, the global object is used
- * Demo

Function Objects

- * A function in JavaScript is just an object
- * Can store inside objects or arrays

Function Objects

```
var o = { x: 5, y: 7 };

o.moveRight = function( steps ) {
    o.x += steps;
};

o.moveLeft = function( steps ) {
    o.x -= steps;
};
```

Argument Validation

- * Functions can take any number of arguments of any type
- * Idiomatic code on the right does type checking

```
function got_2_numbers(x, y) {  
    if ( typeof(x) !== 'number' ) {  
        return 1;  
    }  
  
    if ( typeof(y) !== 'number' ) {  
        return 1;  
    }  
  
    return x * y;  
}
```

Default Arguments

- * The idiomatic code on the right assigns default value to arguments

```
function print_times( text, times ) {  
  if ( text == null ) {  
    return;  
  }  
  
  times = times || 5;  
  
  for ( var i=0; i < times; i++ ) {  
    console.log( text );  
  }  
}
```

Variadic Functions

- * The idiomatic JavaScript code below takes a variable number of arguments into an array (starting at the second)

```
function sum(start_value) {  
  var params = Array.prototype.slice.call(arguments, 1);  
  
  return params.reduce(function(a, b) {  
    return a + b;  
  });  
}  
  
// returns 14  
sum(10, 5, 2, 7);
```

Q & A



Functions Lab

- * Write a function that takes many strings, and prints the total number of characters
- * Write a function that takes a list of numbers and prints the sum of all even numbers
- * Write a function that takes a string as input and prints how many digits it has

DOM Scripting

Using JS To Manipulate the web page



<http://www.flickr.com/photos/jram23/3088840966/>

The DOM

- * Stands for Document Object Model
- * Every HTML element has a JS object “bound” to it in a special bond

`div`

`HTML`

`HTMLDivElement`

`JS`

The DOM

- * Can use `getElementById` to find a specific element
- * Can use `getElementsByName` to get all elements with a specified name

```
<p id="text"></p>
```

```
var t = document.getElementById('text');  
t.innerHTML = "Hello World";
```

DOM Agenda

- * jQuery Introduction
- * Handling Events
- * Writing a simple web app

The Facts

- * Every browser is different. Really.
- * W3 Standard
- * Webkit
- * Mozilla
- * Microsoft

Ajax Libraries

- * Developer tools that wrap common functionality and create a standard
- * Many options to choose from. We'll use jQuery

jQuery Basics

```
$( 'p' ).html( 'hello jQuery' );
```

```
<p></p>
```

- * Wrap a normal DOM element inside a jQuery object
- * All operations on the element are performed by jQuery
- * Unified and cross browser API

jQuery Basics: Selectors

- * Powerful element selection and manipulations
- * Works “the same” for all mobile & desktop browsers

```
<p id="text"></p>
```

```
$( 'p#text' ).html("Hello World");
```

Selector returning a
jQuery Object

An action to perform
on the object

jQuery Actions

- * Each jQuery object supports a wide range of functions to alter the elements

```
$(' #foo ').addClass('wide');  
$(' #foo ').removeClass('wide');  
  
$(' div#foo ').css('background', 'red');  
  
$(' div#foo ').hide();  
$(' div#foo ').show();
```

jQuery Actions Example

- * Zebra stripe a table using jQuery

```
$('table tr:nth-child(2n)').css('background', '#ccc');
```

jQuery Events

- * Define what to do when something happens
- * in jQuery, “bind” a function to an event
- * After the bind, every time the event happens
your callback is called, with the object as the “this” argument

jQuery Events

- * document.ready
- * click
- * hover
- * keypress
- * mousemove
- * resize
- * scroll
- * select
- * submit
- * Full list: <http://api.jquery.com/category/events/>

jQuery Demo

- * Build a red spotter game
- * Game should present the user with 4 colored squares, only one is red
- * User should click on the red square

jQuery API

- * Other useful methods:
- * attr - change an attribute value
- * css - change a style attribute
- * html - change innerHTML content
- * data - store arbitrary data inside an element

Q & A



jQuery Lab

- * Write a web application that splits the screen into four sections. Each click on a section should display a sentence inside the clicked section
- * Write a web app to convert time units. User should enter time in seconds, minutes or hours, and convert to all the others
- * Use HTML5 Boilerplate and jQuery

Thank You

- * Ynon Perek
- * ynonperek@yahoo.com
- * ynonperek.com