

JavaScript Crash Course

Jonathan Sumner Evans

November 8, 2016



Colorado School of Mines
Linux Users Group

JavaScript is **NOT** Java ¹

- JavaScript was written was created in 10 days in May 1995 by Brendan Eich.
- JavaScript was originally called Mocha and was renamed to LiveScript before being renamed again to JavaScript.
- Why **Java**Script? Because Java happened to be popular then (that was before people realized how awful it is) and JavaScript looks syntactically similar at a glance.
- JavaScript is standardized² by Ecma International and there have been a number of ECMAScript versions. The latest is ECMAScript 6, but it is not fully supported by any browsers, including Firefox which only has partial support.

¹ Lots of this slide's information is from: https://www.w3.org/community/webed/wiki/A_Short_History_of_JavaScript

² JavaScript standards aren't actually that standard.

JavaScript is **NOT** Java ¹

- JavaScript was written was created in 10 days in May 1995 by Brendan Eich.
- JavaScript was originally called Mocha and was renamed to LiveScript before being renamed again to JavaScript.
- Why **JavaScript**? Because Java happened to be popular then (that was before people realized how awful it is) and JavaScript looks syntactically similar at a glance.
- JavaScript is standardized² by Ecma International and there have been a number of ECMAScript versions. The latest is ECMAScript 6, but it is not fully supported by any browsers, including Firefox which only has partial support.

¹ Lots of this slide's information is from: https://www.w3.org/community/webed/wiki/A_Short_History_of_JavaScript

² JavaScript standards aren't actually that standard.

JavaScript is **NOT** Java ¹

- JavaScript was written was created in 10 days in May 1995 by Brendan Eich.
- JavaScript was originally called Mocha and was renamed to LiveScript before being renamed again to JavaScript.
- Why **Java**Script? Because Java happened to be popular then (that was before people realized how awful it is) and JavaScript looks syntactically similar at a glance.
- JavaScript is standardized² by Ecma International and there have been a number of ECMAScript versions. The latest is ECMAScript 6, but it is not fully supported by any browsers, including Firefox which only has partial support.

¹ Lots of this slide's information is from: https://www.w3.org/community/webed/wiki/A_Short_History_of_JavaScript

² JavaScript standards aren't actually that standard.

JavaScript is **NOT** Java ¹

- JavaScript was written was created in 10 days in May 1995 by Brendan Eich.
- JavaScript was originally called Mocha and was renamed to LiveScript before being renamed again to JavaScript.
- Why **Java**Script? Because Java happened to be popular then (that was before people realized how awful it is) and JavaScript looks syntactically similar at a glance.
- JavaScript is standardized² by Ecma International and there have been a number of ECMAScript versions. The latest is ECMAScript 6, but it is not fully supported by any browsers, including Firefox which only has partial support.

¹ Lots of this slide's information is from: https://www.w3.org/community/webed/wiki/A_Short_History_of_JavaScript

² JavaScript standards aren't actually that standard.

JavaScript is an **untyped** language. I don't know what that means and I don't think that Brendan did either when he wrote the language.

Variables are declared using the `var` keyword¹.

Examples:

- `var name;` - creates variable name of type undefined.
- `var name = 'Sumner';` - you can initialize a variable when you declare it.

¹Sometimes you don't need to use `var`.

JavaScript has six primitive types:

- Boolean (true or false)
- Null
- Undefined (yes, this is a type)
- Number (can be a number between $-(2^{53} - 1)$ and $2^{53} - 1$, NaN, -Infinity, or Infinity).
- String (single or double quotes declares a string literal²)
- Symbol (new in ECMAScript 6)

Everything else is an object.

¹Info on this slide from: https://developer.mozilla.org/en-US/docs/Web/JavaScript/Data_structures

²Single quotes is recommended by Douglas Crockford because HTML normally uses double quotes and to avoid conflicts when manipulating DOM objects, single quotes should be used.

Scope I

There are two scopes in JavaScript: global and function.¹

Variables are *hoisted* to the top of the function they are declared in. Thus, the following is entirely valid.

```
function () {  
    b = 5;  
    console.log(b); // logs 5  
    var b = 3  
    console.log(b); // logs 3  
}
```

Thus, it is recommended that you declare all of your variables at the top of your functions (one exception to this rule is counter variables).

¹In ES6, variables declared with `let` are actually block scope.

Scope II

Variables declared outside of a function are automatically in the global scope.

Variables declared within a function *without* the `var` keyword are also in the global scope.

```
var a = 2;
(function() {
  b = 3
})(); // this creates and invokes the function
      immediately

console.log(a); // logs 2
console.log(b); // logs 3
```

Because your code could coexist with other people's code, on the same HTML page, it is recommended that you reduce your *global footprint* by creating only a few global objects and then putting everything into that.

Arrays

JavaScript arrays are basically vectors.

Example:

```
var arr = [1, 'a', {}, [], true];  
arr[0] = 'not a number';  
arr.push('this is basically a vector');  
console.log(arr);
```

Output:

```
[ 'not a number', 'a', {}, [], true, 'this is  
  basically a vector' ]
```

Note that the elements of an array do not have to be the same type.

Objects

JavaScript objects are basically maps.

Functions

Functions are just objects with two special properties: a context (scope) and the function code.

Functions: Closure

JavaScript is Pseudoclassical.

Truthy, Falsy and == vs ===

JavaScript has the notion of being *truthy* and *falsy*.

JavaScript has two equality operators:

- == compares without checking variable type.
- === compares and checks variable type.

Pitfalls

There are **lots** of JavaScript libraries. One of the most widely used is jQuery (<http://jquery.com/>). It has good documentation and is really good for DOM manipulation.

I also have a few nice prototype overloads in a repository at <https://github.com/sumnerevans/js-utils> (MIT License).

DOM Manipulation

The *Document Object Model* is an API used by JavaScript to interact with the elements of an HTML document.¹

¹https://en.wikipedia.org/wiki/Document_Object_Model

Additional Resources

A lot of this presentation was based off of *JavaScript: The Good Parts* by Douglas Crockford. This is an essential read for anyone interested in learning JavaScript for anything more than writing a few simple scripts.

MDN is the best resource for JavaScript documentation (<https://developer.mozilla.org/en-US/>).