Introduction to JavaScript

Hunter Jorgensen

History of JavaScript

- Created in 1995 by Brendan Eich and first called Mocha
- Netscape (recognize that name?) controlled early development, and partnered with Sun Microsystems (creators of Java) against Microsoft
- Mocha became JavaScript as a marketing ploy: it was meant to be the companion language to Java (but that didn’t happen)
- Eventually JavaScript finds it home as server side language

JavaScript

- Back in the early days of JavaScript, different companies (Microsoft, Netscape) had their own rules for how JavaScript should work.
- NetScape worked with the Ecmas Standards Organization to define the features and specifications of a document later known as ECMAScript.
- JavaScript is an IMPLEMENTATION of ECMAScript (ActionScript, JScript also implement this.)
- New features are added to this almost year, though it does follow a release schedule
- This is why you may see reference to something called ES6/ES10, etc. with ES.Next being the most recent release
- Like other cases, not all browsers support all functionality

Coding in JavaScript

4 ways to code along:

1. Use something like JSFiddle/JSBin to code on a browser
2. Use node on your terminal
3. Use <script>...</script> in an HTML document and open it in the browser
4. Developer Tools -> Click Console Tab -> Use terminal at bottom of screen (click multiline input tool for easier reading!)

First Code

Demo: https://gist.github.com/hunterjorgensen167/05f7891a56e147ca3ae9c18b0ee4cad7

For now, just know that odd bits are:

1. Helping to improve performance.
2. Prevent stupid semantics in JavaScript that trip up beginners.
3. Prevent code snippets executed in the console from interacting with one-another (e.g., having something created in one console execution being used for a different console execution)
4. Only used when separated out into second JS file (not used in <style> tags)

Declaring a Variable

- var: declares a variable, but is not commonly used anymore
- let: declares a variable at local level
- const: declares a constant/read only variable at the local level

Scope, or the thing about var...

With `var`, the scope of a variable was harder to track since it took on its execution scope (i.e. the enclosing function or even a global scope). Opposed to this, `let` follows a much more common scoping and exists only at the block level.

Demo

Types

- JavaScript has 7 primitive types:
  1. Boolean. true and false.
  2. null. A special keyword denoting a null value. (Because JavaScript is case-sensitive, null is *not* the same as Null, NULL, or any other variant.)
  3. undefined: A top-level property whose value is not defined.
  4. Number. An integer or floating point number. For example: 42 or 3.14159.
  5. BigInt. An integer with arbitrary precision. For example: 900719925474099 2n.
  6. String. A sequence of characters that represent a text value. For example: "Howdy"
  7. Symbol (new in ECMAScript 2015). A data type whose instances are unique and immutable.
- and Object
  1. This is a container for objects

Dynamically Typed

JavaScript is a dynamically typed language.

If using the let (or even var) to declare a variable, you can change the value, including that value's type. So a string variable can become a number, etc.

Demo

Equality and Sameness

JavaScript uses both == and === to compare different values.

== will attempt to convert the values and then make a comparison

=== will NOT attempt to convert the values before comparing

(!= and !== are the analogues to these)

Why does JavaScript do this?

Demo

This is why you get odd behaviors as seen here!

Objects

Objects in JavaScript allow for the creation and management of more complex data. Objects are declared within brackets, and then using key-value pairings.

Demo

Notice the 2 ways of accessing keys in these objects: array vs dot notation.

Arrays

The other important type of object in JavaScript is the array. Arrays are 0-indexed (do you know what this means?) and are essentially objects where the key is a number between 0 and the size of the array.

Arrays are powerful data structures in JavaScript, but are full of many eccentricities that you can explore on your own.

Demo

We'll go over more functionality of arrays in a bit

Function

There are 2 ways to declare functions in JavaScript. Once a function is declared, it can be invoked like most other programming languages:

Demo

Note that functions can take arguments. Also, function names can be overwritten like other variables.

Functions in JavaScript are first-class objects. What does this mean?

A Comment on Functions

There are a lot of funny unique behaviors about JavaScript functions, that aren't commonly seen in other programming languages: they can be passed as arguments, or variables. Additionally, they are essentially objects. This can get quite complicated, but can allow for some pretty advanced logic.

Demo

Hoisting

JavaScript allows for some unusual behavior, such as using variables or functions before they're explicitly declared (note that this only works with the `var`. What happens if you use `let`/`const`?)

Demo

Template Literal

You can construct a string with a grave accents (`...`), that allows you to perform logic or reference variables.

Demo

Functions: Default Parameters

One feature of functions is that they can accept default parameters: a functionality similar to Python and other languages. This can be helpful if you're expecting `undefined`s. Since JavaScript can only allow one function with a name at a time, this allows an alternative to overloading.

Demo

truthy/falsy

`true` and `false` are keywords in JavaScript, so we use the terms truthy and falsy to indicate code that translates to true or false in a Boolean context. The following things translate to false:

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

This means that ANYTHING ELSE is translated as true (including empty objects)

Demo

Debugging

When testing code, it is very common to print to console or alert. For instance, you can use console.log(...) to print to the browser terminal (Developer Tools -> Console) or alert(...) to cause a popup (does not require going into developer tools, but it's very annoying and harder to parse.)

Even better is using `debugger;`, which will require your Developer Tools console to be open.

Demo

Control Flow and Loops

JavaScript is capable of many standard pieces of control flow that you would be familiar with from languages such as Java, Python, etc. This includes things like:

- if...else statements
- switch statements
- for loops
- while loops
- do...while loops

Demo

JS Unique Loops: for...in

for...in loops are unique loops that all you to iterate over Objects and get their keys. Note that this is NOT meant to be used with Arrays or other iterables.

Demo

JS Unique Loops: for...of

for...of loops in JavaScript are used to iterate over enumerables. This includes Arrays, Sets, Maps and others.

Demo

JavaScript Organization

Another feature of JavaScript is that files can be organized into separate files (specifically JS files). We will not go over this as the frameworks we use will provide this functionality to us for free. But it is important to be aware that it is easy to override commonly named functions in the namespace and always a good reminder to organize your code into small chunks of files.

We'll go into this (ESModules) when we cover React and Node.js

Bonus Points: DnD Dice Roller

Create a dice rolling function to help me play Dungeons and Dragons!

https://gist.github.com/ahjorgen167/82bea9cdad6849d3142cbbcfe6912e75

Worth 2 bonus points, submit to Nandish within 24 hours in a file labeled {firstname}_{lastname}_js_bonus.js