# Introduction to JavaScript

* Short history (really short)
  + What’s JavaScript?
  + Creator
* Set up development environment
  + Text editor
  + JavaScript Interpreter
  + Debug tool
  + Frameworks
* Basic concept
  + **Number**
  + **String**
  + **Boolean**
  + **Object** 
    - **Function**
    - **Array**
    - **Date**
    - **RegExp**
    - **null**
    - **undefined**

# Short history

## What’s JavaScript?

JavaScript (JS) is a programming language primarily used by web browsers for dynamic scripting of web pages. It can also be used on the server side to perform any **sort** of action. Unlike most programming languages, the JavaScript language has no concept of input or output. It is designed to run as a scripting language in a host environment, and it is up to the host environment to provide mechanisms for communicating with the outside world. The most common host environment is the browser, but JavaScript interpreters can also be found in Adobe Acrobat, Photoshop, SVG images, Yahoo!'s Widget engine, as well as server side environments such as node.js. However the list of the areas where JavaScript is used just begins here. It also includes NoSQL databases, like the open source Apache CouchDB, embedded computers, or complete desktop environments, like GNOME (one of the most popular GUIs for GNU/Linux operating systems).

## Creator

JavaScript was created in 1995 by Brendan Eich, an engineer at Netscape, and first released with Netscape 2 early in 1996. And has several names like: Mocha, LiveScript, JScript, ECMAScript and JavaScript.

# Set up develoment enviroment

## Text editor

* Visual Studio Community Edition (<http://www.visualstudio.com/en-us/products/visual-studio-community-vs.aspx>)
* SublimeText (<http://www.sublimetext.com/>)
* Notepad++ (<http://www.notepad-plus-plus.org/>)

## JavaScript Interpreter

* Any popular browser
* Nodejs (<http://nodejs.org/>)

## Debugs tool

* Developers tools
* Firebug (<http://getfirebug.com/>)

## Frameworks

* JQuery (<http://http://jquery.com>)
* Prototype (<http://prototypejs.org>)
* MooTools (<http://mootools.net>)

# Basic Concept

JavaScript is an object oriented dynamic language; it has types and operators, standard built-in objects, and methods. Its syntax comes from the Java and C languages, so many structures from those languages apply to JavaScript as well. One of the key differences is that JavaScript does not have classes; instead, the class functionality is accomplished by object prototypes. The other main difference is that functions are objects, giving functions the capacity to hold executable code and be passed around like any other object.

* Number
* String
* Boolean
* Object
* Function
* Array
* Date
* RegExp
* null
* undefined

## Numbers

In JavaScript are "double-precision 64-bit format IEEE 754 values", according to the spec.

0.1 + 0.2 == 0.30000000000000004

Some build-in function:

* **Math** object
* **parseInt** and **parseFloat** (functions)
* unary **+** operator for convert
* **NaN** and **isNaN** (function)
* **Infinity**, minus **Infinity** and **isFinite**

## Strings

Strings in JavaScript are sequences of characters. More accurately, they are sequences of Unicode characters, with each character represented by a 16-bit number. This should be welcome news to anyone who has had to deal with internationalization.

Some build-in function:

* **chartAt** (function)
* **replace** (function)
* **toUpperCase** (function)

## Others types

JavaScript distinguishes between null, which is a value that indicates a deliberate non-value (and is only accessible through the null keyword), and undefined, which is a value of type 'undefined' that indicates an uninitialized value — that is, a value hasn't even been assigned yet.

(undefined is a constant)

JavaScript has a boolean type, with possible values true and false (both of which are keywords). Any value can be converted to a boolean according to the following rules:

* false, 0, the empty string (""), NaN, null, and undefined all become false.
* all other values become true.

(Use Boolean function for checking)

## Variables

New variables in JavaScript are declared using the **var** keyword:

var a;

var name = "John";

An important difference from other languages like Java is that in JavaScript, blocks do not have scope; only functions have scope. However, starting with ECMAScript Edition 6, let and const declarations allow you to create block-scoped variables.

## Operators

JavaScript's numeric operators are +, -, \*, / and % - which is the remainder operator. Values are assigned using =, and there are also compound assignment statements such as += and -=. These extend out to x = x *operator* y.

x += 5;

x = x + 5;

You can use ++ and -- to increment and decrement respectively. These can be used as prefix or postfix operators.

The + operator also does string concatenation:

"hello" + " world"; // "hello world"

If you add a string to a number (or other value) everything is converted in to a string first. This might catch you up:

"3" + 4 + 5; // "345"

3 + 4 + "5"; // "75"

Adding an empty string to something is a useful way of converting it.

Comparisons in JavaScript can be made using <, >, <= and >=. These work for both strings and numbers. Equality is a little less straightforward. The double-equals operator performs type coercion if you give it different types, with sometimes interesting results:

"dog" == "dog"; // true

1 == true; // true

To avoid type coercion, use the triple-equals operator:

1 === true; // false

true === true; // true

There are also != and !== operators.

JavaScript also has bitwise operations.

## Control Structure

JavaScript has a similar set of control structures to other languages in the C family.

var name = "kittens";

if (name == "puppies") {

name += "!";

} else if (name == "kittens") {

name += "!!";

} else {

name = "!" + name;

}

name == "kittens!!"

JavaScript has while loops and do-while loops

while (true) {

// an infinite loop!

}

var input;

do {

input = get\_input();

} while (inputIsNotValid(input))

JavaScript's for loop is the same as that in C and Java

for (var i = 0; i < 5; i++) {

// Will execute 5 times

}

The && and || operators use short-circuit logic, which means whether they will execute their second operand is dependent on the first. This is useful for checking for null objects before accessing their attributes:

var name = o && o.getName();

Or for setting default values:

var name = otherName || "default";

JavaScript has a ternary operator for conditional expressions:

var allowed = (age > 18) ? "yes" : "no";

The switch statement can be used for multiple branches based on a number or string:

switch (action) {

case 'draw':

drawIt();

break;

case 'eat':

eatIt();

break;

default:

doNothing();

}

## Objects

JavaScript objects can be thought of as simple collections of name-value pairs. As such, they are similar to:

* Dictionaries in Python
* Hashes in Perl and Ruby
* Hash tables in C and C++
* HashMaps in Java
* Associative arrays in PHP

There are two basic ways to create an empty object:

var obj = new Object();

And:

var obj = {};

And....

function Person(name, age) {

this.name = name;

this.age = age;

}

// Define an object

var You = new Person("You", 24); // We are creating a new person named "you" // (that was the first parameter, and the age..)

Once created, an object's properties can again be accessed in one of two ways:

obj.name = "Simon";

var name = obj.name;

And...

obj["name"] = "Simon";

var name = obj["name"];

These are also semantically equivalent. The second method has the advantage that the name of the property is provided as a string, which means it can be calculated at run-time though using this method prevents some JavaScript engine and minifier optimizations being applied. It can also be used to set and get properties with names that are reserved words:

obj.for = "Simon"; // Syntax error, because 'for' is a reserved word obj["for"] = "Simon"; // works fine

Object literal syntax can be used to initialize an object in its entirety:

var obj = {

name: "Carrot",

"for": "Max",

details: {

color: "orange",

size: 12

}

}

Attribute access can be chained together:

obj.details.color; // orange

obj["details"]["size"]; // 12

## Arrays

Arrays in JavaScript are actually a special type of object. They work very much like regular objects (numerical properties can naturally be accessed only using [] syntax) but they have one magic property called 'length'.

One way of creating arrays is as follows:

var a = new Array();

a[0] = "dog";

a[1] = "cat";

a[2] = "hen";

a.length; // 3

A more convenient notation is to use an array literal:

var a = ["dog", "cat", "hen"];

a.length; // 3

Note that array.length isn't necessarily the number of items in the array. Consider the following:

var a = ["dog", "cat", "hen"];

a[100] = "fox";

a.length; // 101

Remember — the length of the array is one more than the highest index.

If you query a non-existent array index, you get undefined:

typeof a[90]; // undefined

If you take the above into account, you can iterate over an array using the following:

for (var i = 0; i < a.length; i++) {

// Do something with a[i]

}

This is slightly inefficient as you are looking up the length property once every loop. An improvement is this:

for (var i = 0, len = a.length; i < len; i++) { // Do something with a[i]

}

An even nicer idiom is:

for (var i = 0, item; item = a[i++];) {

// Do something with item

}

Another way to iterate is to use the for...in loop. Note that if someone added new properties to Array.prototype, they will also be iterated over by this loop:

for (var i in a) {

// Do something with a[i]

}

**Source:** [**https://developer.mozilla.org/en-US/docs/Web/JavaScript/A\_re-introduction\_to\_JavaScript**](https://developer.mozilla.org/en-US/docs/Web/JavaScript/A_re-introduction_to_JavaScript?redirectlocale=en-US&redirectslug=JavaScript%2FA_re-introduction_to_JavaScript)

**Other sources:** [**https://msdn.microsoft.com/es-co/library/ie/d1et7k7c%28v=vs.94%29.aspx**](https://msdn.microsoft.com/es-co/library/ie/d1et7k7c%28v=vs.94%29.aspx)

[**https://developer.mozilla.org/es/docs/Web/JavaScript/Referencia**](https://developer.mozilla.org/es/docs/Web/JavaScript/Referencia)

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