programming with JavaScript and node.js

## Notes by

## Siddharth Ramawat

# What is JavaScript?

*JavaScript* was initially created to *“make web pages alive”*.

The programs in this language are called *scripts*. They can be written right in a web page’s HTML and run automatically as the page loads.

Scripts are provided and executed as plain text. They *don’t* need special preparation or compilation to run.

In this aspect, JavaScript is very different from another language called [Java](https://en.wikipedia.org/wiki/Java_(programming_language)).

But as it evolved, JavaScript became a fully independent language with its own specification called ECMAScript, and now it has no relation to Java at all.

# Why JavaScript?

When JavaScript was created, it initially had another name: “LiveScript”. But Java was very popular at that time, so it was decided that positioning a new language as a “younger brother” of Java would help.

**JavaScript** is a ***must*** for students and working professionals to become a great Software Engineer especially when they are working in Web Development Domain. I will list down some of the key advantages of learning JavaScript:

* JavaScript is the most popular **programming language** in the world and that makes it a programmer’s great choice. Once you learnt JavaScript, it helps you developing great front-end as well as back-end software using different JavaScript based frameworks like jQuery, Node.JS etc.
* JavaScript is everywhere, it comes installed on every modern web browser and so to learn JavaScript you really do not need any special environment setup. For example Chrome, Mozilla Firefox, Safari and every browser you know as of today, supports JavaScript.
* JavaScript helps you create really beautiful and crazy fast websites. You can develop your website with a console like look and feel and give your users the best Graphical User Experience.
* JavaScript usage has now extended to mobile app development, desktop app development, and game development. This opens many opportunities for you as JavaScript Programmer.
* Due to high demand, there is tons of job growth and high pay for those who know JavaScript. You can navigate over to different job sites to see what having JavaScript skills looks like in the job market.
* Great thing about JavaScript is that you will find tons of frameworks and Libraries already developed which can be used directly in your software development to reduce your time to market.

There could be 1000s of good reasons to learn JavaScript Programming. But one thing for sure, to learn any **programming language**, not only JavaScript, you just need to code, and code and finally code until you become expert.

Today, JavaScript can execute not only in the browser, but also on the server, or actually on any device that has a special program called [the JavaScript engine](https://en.wikipedia.org/wiki/JavaScript_engine).

The browser has an embedded engine sometimes called a “JavaScript virtual machine”.

Different engines have different “codenames”. For example:

* [V8](https://en.wikipedia.org/wiki/V8_(JavaScript_engine)) – in Chrome and Opera.
* [SpiderMonkey](https://en.wikipedia.org/wiki/SpiderMonkey) – in Firefox.
* …There are other codenames like “Trident” and “Chakra” for different versions of IE, “ChakraCore” for Microsoft Edge, “Nitro” and “SquirrelFish” for Safari, etc.

The terms above are good to remember because they are used in developer articles on the internet. We’ll use them too. For instance, if “a feature X is supported by V8”, then it probably works in Chrome and Opera.

# How do engines work?

Engines are complicated. But the basics are easy.

1. The engine (embedded if it’s a browser) reads (“parses”) the script.
2. Then it converts (“compiles”) the script to the machine language.
3. And then the machine code runs, pretty fast.

The engine applies optimizations at each step of the process. It even watches the compiled script as it runs, analyses the data that flows through it, and applies optimizations to the machine code based on that knowledge. When it’s done, scripts run quite fast.

# What can in-browser JavaScript do?

Modern JavaScript is a “safe” programming language. It does not provide low-level access to memory or CPU, because it was initially created for browsers which do not require it.

JavaScript’s capabilities greatly depend on the environment it’s running in. For instance, [Node.js](https://wikipedia.org/wiki/Node.js) supports functions that allow JavaScript to read/write arbitrary files, perform network requests, etc.

In-browser JavaScript can do everything related to webpage manipulation, interaction with the user, and the webserver.

For instance, in-browser JavaScript is able to:

* Add new HTML to the page, change the existing content, and modify styles.
* React to user actions, run on mouse clicks, pointer movements, and key presses.
* Send requests over the network to remote servers, download and upload files (so-called [AJAX](https://en.wikipedia.org/wiki/Ajax_(programming)) and [COMET](https://en.wikipedia.org/wiki/Comet_(programming)) technologies).
* Get and set cookies, ask questions to the visitor, show messages.
* Remember the data on the client-side (“local storage”).

# What CAN’T in-browser JavaScript do?

JavaScript’s abilities in the browser are limited for the sake of the user’s safety. The aim is to prevent an evil webpage from accessing private information or harming the user’s data.

Examples of such restrictions include:

* JavaScript on a webpage may not read/write arbitrary files on the hard disk, copy them or execute programs. It has no direct access to OS system functions.

Modern browsers allow it to work with files, but the access is limited and only provided if the user does certain actions, like “dropping” a file into a browser window or selecting it via an <input> tag.

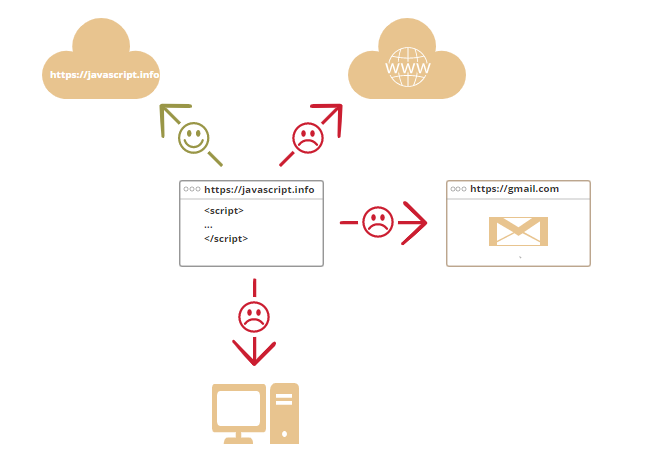
There are ways to interact with camera/microphone and other devices, but they require a user’s explicit permission. So a JavaScript-enabled page may not sneakily enable a web-camera, observe the surroundings and send the information to the [NSA](https://en.wikipedia.org/wiki/National_Security_Agency).

* Different tabs/windows generally do not know about each other. Sometimes they do, for example when one window uses JavaScript to open the other one. But even in this case, JavaScript from one page may not access the other if they come from different sites (from a different domain, protocol or port).

This is called the “Same Origin Policy”. To work around that, *both pages* must agree for data exchange and contain a special JavaScript code that handles it. We’ll cover that in the tutorial.

This limitation is, again, for the user’s safety. A page from *http://anysite.com* which a user has opened must not be able to access another browser tab with the URL*http://gmail.com* and steal information from there.

* JavaScript can easily communicate over the net to the server where the current page came from. But its ability to receive data from other sites/domains is crippled. Though possible, it requires explicit agreement (expressed in HTTP headers) from the remote side. Once again, that’s a safety limitation.



Such limits do not exist if JavaScript is used outside of the browser, for example on a server. Modern browsers also allow plugin/extensions which may ask for extended permissions.

# What makes JavaScript unique?

There are at least *three* great things about JavaScript:

* Full integration with HTML/CSS.
* Simple things are done simply.
* Support by all major browsers and enabled by default.

JavaScript is the only browser technology that combines these three things.

That’s what makes JavaScript unique. That’s why it’s the most widespread tool for creating browser interfaces.

That said, JavaScript also allows to create servers, mobile applications, etc.

# Languages “over” JavaScript

The syntax of JavaScript does not suit everyone’s needs. Different people want different features.

That’s to be expected, because projects and requirements are different for everyone.

Recently a plethora of new languages appeared those are *transpiled* (converted) to JavaScript before they run in the browser.

Modern tools make the transpilation very fast and transparent, actually allowing developers to code in another language and auto-converting it “under the hood”.

Examples of such languages:

* [CoffeeScript](http://coffeescript.org/) is a “syntactic sugar” for JavaScript. It introduces shorter syntax, allowing us to write clearer and more precise code. Usually, Ruby devs like it.
* [TypeScript](http://www.typescriptlang.org/) is concentrated on adding “strict data typing” to simplify the development and support of complex systems. It is developed by Microsoft.
* [Flow](http://flow.org/) also adds data typing, but in a different way. Developed by Facebook.
* [Dart](https://www.dartlang.org/) is a standalone language that has its own engine that runs in non-browser environments (like mobile apps), but also can be transpiled to JavaScript. Developed by Google.

There are more. Of course, even if we use one of transpiled languages, we should also know JavaScript to really understand what we’re doing.

# Hello World using JavaScript.

In any JavaScript program we can either use the console or the html page to print our text. The console is usually used by developers and the output to the console cannot be seen by the clients on the web page, but can use the console tab in the inspect element section to view the output, or the output can be directly seen using a normal command prompt / terminal.

JavaScript can be implemented using JavaScript statements that are placed within the **<script>... </script>** HTML tags in a web page.

You can place the **<script>** tags, containing your JavaScript, anywhere within your web page, but it is normally recommended that you should keep it within the **<head>** tags.

<html>

<body>

<script language = "javascript" type = "text/javascript">

<!--

document.write("Hello World!")

//-->

</script>

</body>

</html>

The <script> tag alerts the browser program to start interpreting all the text between these tags as a script.

<script ...>

JavaScript code

</script>

The script tag takes two important attributes −

* **Language** − this attribute specifies what scripting language you are using. Typically, its value will be javascript. Although recent versions of HTML (and XHTML, its successor) have phased out the use of this attribute.
* **Type** − this attribute is what is now recommended to indicate the scripting language in use and its value should be set to "text/javascript".

So your JavaScript segment will look like −

<script language = "javascript" type = "text/javascript">

JavaScript code

</script>

JavaScript Frameworks

There are many useful **Javascript frameworks** and libraries available:

* Angular
* React
* jQuery
* Vue.js
* Ext.js
* Ember.js
* Meteor
* Mithril
* Node.js
* Polymer
* Aurelia
* Backbone.js

It is really impossible to give a complete list of all the available Javascript frameworks and libraries. The Javascript world is just too large and too much new is happening.

# Applications of JavaScript Programming

As mentioned before, **JavaScript** is one of the most widely used **programming languages** (Front-end as well as Back-end). It has its presence in almost every area of software development. I'm going to list few of them here:

* **Client side validation** - This is really important to verify any user input before submitting it to the server and JavaScript plays an important role in validating those inputs at front-end itself.
* **Manipulating HTML Pages** - JavaScript helps in manipulating HTML page on the fly. This helps in adding and deleting any HTML tag very easily using JavaScript and modify your HTML to change its look and feel based on different devices and requirements.
* **User Notifications** - You can use JavaScript to raise dynamic pop-ups on the webpages to give different types of notifications to your website visitors.
* **Back-end Data Loading** - JavaScript provides Ajax library which helps in loading back-end data while you are doing some other processing. This really gives an amazing experience to your website visitors.
* **Presentations** - JavaScript also provides the facility of creating presentations which gives website look and feel. JavaScript provides RevealJS and BespokeJS libraries to build a web-based slide presentations.
* **Server Applications** - Node JS is built on Chrome's JavaScript runtime for building fast and scalable network applications. This is an event based library which helps in developing very sophisticated server applications including Web Servers.

This list goes on, there are various areas where millions of software developers are happily using JavaScript to develop great websites and others software.

# Advantages of JavaScript

The merits of using JavaScript are −

* **Less server interaction** − you can validate user input before sending the page off to the server. This saves server traffic, which means less load on your server.
* **Immediate feedback to the visitors** − they don't have to wait for a page reload to see if they have forgotten to enter something.
* **Increased interactivity** − you can create interfaces that react when the user hovers over them with a mouse or activates them via the keyboard.
* **Richer interfaces** − you can use JavaScript to include such items as drag-and-drop components and sliders to give a Rich Interface to your site visitors.

# Limitations of JavaScript

We cannot treat JavaScript as a full-fledged programming language. It lacks the following important features −

* Client-side JavaScript does not allow the reading or writing of files. This has been kept for security reason.
* JavaScript cannot be used for networking applications because there is no such support available.
* JavaScript doesn't have any multi-threading or multiprocessor capabilities.

Once again, JavaScript is a lightweight, interpreted programming language that allows you to build interactivity into otherwise static HTML pages.

# JavaScript Development Tools

One of major strengths of JavaScript is that it does not require expensive development tools. You can start with a simple text editor such as Notepad. Since it is an interpreted language inside the context of a web browser, you don't even need to buy a compiler.

To make our life simpler, various vendors have come up with very nice JavaScript editing tools. Some of them are listed here −

* **Microsoft FrontPage** − Microsoft has developed a popular HTML editor called FrontPage. FrontPage also provides web developers with a number of JavaScript tools to assist in the creation of interactive websites.
* **Macromedia Dreamweaver MX** − Macromedia Dreamweaver MX is a very popular HTML and JavaScript editor in the professional web development crowd. It provides several handy prebuilt JavaScript components, integrates well with databases, and conforms to new standards such as XHTML and XML.
* **Macromedia HomeSite 5** − HomeSite 5 is a well-liked HTML and JavaScript editor from Macromedia that can be used to manage personal websites effectively.

# Where is JavaScript Today?

The ECMAScript Edition 5 standard will be the first update to be released in over four years. JavaScript 2.0 conforms to Edition 5 of the ECMAScript standard, and the difference between the two is extremely minor.

The specification for JavaScript 2.0 can be found on the following site: <http://www.ecmascript.org/>

Today, Netscape's JavaScript and Microsoft's JScript conform to the ECMAScript standard, although both the languages still support the features that are not a part of the standard.