**Introduction to JavaScript**

**JavaScript** is a lightweight, case sensitive, [cross-platform](../Platform%20languages.docx) and interpreted programming ([scripting](../programming%20and%20scripting%20languages.docx)) language which uses the **Unicode** character set. It was initially created to “make web pages alive”, as it was used as a **Client-side** developments, but letter it evaluated to be used in **Server-side** developments (Node.js and other frameworks) and also many **non-browser environments** like: gaming and applications. JavaScript contains a standard library of objects, like [**Array**](../Data%20type/Reference/Array/Array.docx), [**Date**](../date%20object.docx), and [**Math**](../Math%20object/Math%20object.docx), and a core set of language elements like [**operators**](../Operators.docx), **control structures**, and [**control** **statements**](../Control%20Statements/Intro/Control%20Statements.docx) so it’s considered an objected or prototype based programming language.

Anciently we are write “Vanilla JS” or “Row JS” or “Native JS” codes which means we was written pure JS codes without any libraries or platforms.

JavaScript is a “Just-in-time (JIT) “OR “Dynamic translation” OR “run-time compilations” OR “interpreted” that means no compiler needed in JS because the host environment translate (compile) JS codes at [**run time**](https://en.wikipedia.org/wiki/Run_time_(program_lifecycle_phase)) **(during)** execution of the codes no before it.

JavaScript is a “**dynamically typed**” language which means you can change data type of variables several times without need to repeat variable keyword (in other words there are existing data types, but variables are not bound to any of them).

Doing mathematical operators are “safe” in JavaScript. We can do anything: divide by zero, treat non-numeric strings as numbers; etc. The script will never stop with a fatal error (“die”). At worst, we’ll get **NaN** as the result.

**JavaScript** is designed to run as a scripting language in a **host environment**, the most common host environment is the browser.

Today, JavaScript has a unique position as the most widely-adopted browser language with full integration with HTML/CSS.

**History of JavaScript**

JavaScript was created in 1995 by ***Brendan Eich*** while he was an engineer at Netscape.



JS was initially called “LiveScript”, but it was later renamed to JavaScript. Unlike most other languages, JavaScript has no concept of input & output.

**JavaScript platforms**

1. Web Development 🡺 By **AngularJS**
2. Web Applications 🡺 By **ElectronJs & ReactJs**
3. Server Applications 🡺 By **Node.js**
4. Games 🡺 By **EaseJS** library
5. Smart watches applications 🡺 By **PebbleJS** library
6. Art 🡺 By Html5 Canvas & **p5.js** library.
7. Machine Learning 🡺 By **ml5.js** library

**JS engine**

JavaScript engine is a computer program that executes JavaScript (JS) code. So any device has JS engine program can run JavaScript codes on it.

The browser has this program inside it (as built-in) and it sometimes called a “**JavaScript virtual machine**”.

As browser has many types also JavaScript virtual machine (JS engine) has many types called “**codename**”, for instance:

**Note:** **JavaScript virtual machine === JS engine === codename**

|  |  |
| --- | --- |
| **Browser type** | **codename** |
| **Firefox** | **SpiderMonkey** |
| **Chrome** | **V8** |
| **Opera** | **V8** |
| **IE** | **Chakra** |
| **Microsoft Edge** | **ChakraCore** |
| **Safari** | **Nitro & SquirrelFish** |

So if we have a feature called “**X**” and this feature is supported by “**V8**”, then it probably works in **Chrome and Opera**.

**How do engines work?**

1. The engine reads (“parses”) the script codes **line by line** from **top** to **down** and from the **most inner** to the **most outer**.
2. Then it converts (“compiles”) the script to the machine language (0, 1).
3. And then the machine (computer) runs this code fast.

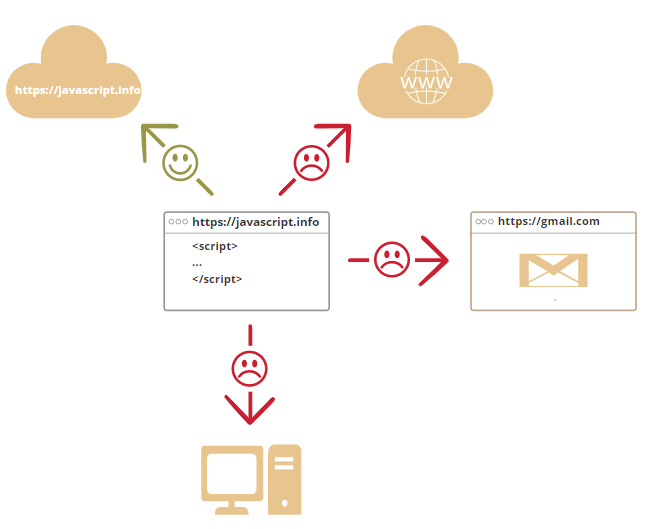
**JS features**

With JS we can:

1. Add new HTML to the page, change the existing content, and modify styles.
2. React to user actions, run on mouse clicks, pointer movements, and key presses.
3. 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).
4. Get and set cookies, ask questions to the visitor, show messages.
5. Remember the data on the client-side (“local storage”).

**JS limitations**

1. 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 functions.
   1. 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.
   2. 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).
2. 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).
   1. 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.
   2. 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.
3. 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.



**Note:** In necessary you can disable JS from your browser setting. Follow, <https://javascript.info/intro#languages-over-javascript>

**Languages over JavaScript**

As JS is the core language for web, all applications and websites should built in JavaScript language. Because JS is not suitable for all users and all needs there are another new language that you can use to build your application or website–without any JS line- then compile (convert) it to JS language to run it in the browser.

There are many Languages can converted to JS:

1. **Dart**
2. **CoffeScript**
3. **TypeScript**
4. **Flow**
5. **Brython**
6. **Elm**
7. **PureScript**
8. **ClojureScript**
9. **Scala.js**
10. **Reason**
11. **Haxe**
12. **Nim**

**Tip:** Of course there are other languages, and it is recommended to take a look at them, at least briefly, after mastering JavaScript