JavaScript

Basics

Since its release in December 4, 1995 (six months after Java, which was released in May 23, 1995), JavaScript has gone through many changes. JavaScript began as a client-side programming language (which runs inside a web browser via a built-in JavaScript engine) for adding interactive contents to the web pages. It became more robust with DHTML (1997) and Ajax (1999). With Node.js (released in May 27, 2009), JavaScript can be used to program server-side and build full-stack web applications, as well as standalone utility scripts. In 2015, the ECMAScript 6 (ES6 or ES2015) introduces major update to the language, such as classes and modules, and make JavaScript a general-purpose programming language capable of complex software tasks.

Introduction

It began as a Client-side Programming Language run inside a web browser via a Built-in JavaScript Engine

JavaScript is the most widely used client-side programming language that lets you supercharge your HTML with interactivity, animation and dynamic visual effect for better User Interface and User Experience (UI/UX). It is:

* a small, lightweight, object-oriented, cross-platform, special-purpose scripting language meant to be run under a host environment (typically a web browser).
* a client-side scripting language to enrich web user-interfaces and create dynamic web pages (e.g., for...input validation, and immediate response to user's actions).
* the engine that supports AJAX (Asynchronous JavaScript and XML - that can be used to update one part of the web page asynchronously), which generate renew interest in JavaScript.

**JavaScript works together with HTML/CSS. HTML provides the contents; CSS specifies the presentation; and JavaScript programs the behavior. Together, they enrich the UI/UX of the web users.**

History and Versions

JavaScript, originally called LiveScript, was created by Brendan Eich at Netscape in 1995. Soon after, Microsoft launched its own version of JavaScript called JScript. Subsequently, Netscape submitted it to ECMA (formerly "European Computer Manufacturers Association", now "Ecma International - European association for standardizing information and communication systems") for standardization, together with Microsoft's JScript.

The ECMA Specification is called "ECMA-262 ECMAScript Language Specification" @ http://www.ecma-international.org/publications/standards/Ecma-262.htm (also approved as "ISO/IEC 16262"):

* ECMA-262 version 1 (June 1997): First edition
* ECMA-262 version 2 (August 1998)
* ECMA-262 version 3 (December 1999): Added regular expressions, try/catch, switch, do-while, etc.
* ECMA-262 version 4 - Abandon due to political differences. In 2007, the TC-39 (the committee responsible for ECMAScript) put up a draft specification for ECMAScript 4, which was massive in scope and introduced many new syntax and features. But a group of developers from Yahoo, Google and Microsoft felt that was too much and created an alternate proposal called ECMAScript 3.1. ECMAScript 4 was never finalized.
* ECMA-262 version 5 and 5.1 (June 2011): ECMAScript 3.1 was eventually standardized as ECMAScript 5. Added "strict mode", JSON, Array iteration methods, etc.
* **ECMA-262 2015: most popularly known as ECMAScript 6 or ES6.**

**This version added significant new syntax for writing complex application, including class declaration, let for local declarations, const for constant local declaration, default parameter values, iterators and for...of loops, Python-style generators, arrow function expression (() => {...}), binary data, typed arrays, new collections (maps, sets and WeakMap), promises, reflection, proxies, template literals for strings, and many more.**

* ECMAScript 2016 (ES7): Since ES6, ECMAScript standards are on yearly release cycles in June.
* This version added exponential operator (\*\*), block-scoping of variables and functions, await and async keywords for asynchronous programming, etc.
* ECMAScript 2017 (ES8): Added string padding, async functions which use generators and promises, Object.values, Object.entries for easy object manipulation, etc.
* ECMAScript 2018 (ES9): Added rest parameters (...), spread operator, asynchronous iteration, additions to regular expression.
* ECMAScript 2019 (ES10): Added Array.prototype.flat, Array.prototype.flatMap, changes Array.sort and Object.fromEntries, catch binding becomes optional, etc.
* ECMAScript 2020 (ES11): introduces a BigInt primitive type for arbitrary-size integers, the nullish coalescing operator (??) and the gloablThis object.
* ECMAScript 2021 (ES12): Added enhancement to strings (replaceAll), promises (Promise.any), and object references. Logical assignment operators (??=, &&=,||=,)

JavaScript vs. Java

Java is a full-fledged general-purpose programming language. It was created by James Gosling at Sun Microsystems (now part of Oracle) and released in August 1995.

JavaScript was created by Brendan Eich at Netscape, also in 1995. Originally called LiveScript, it was a small and lightweight special-purpose language for writing client-side program running inside a browser to create active user-interface and generate dynamic web pages. It was renamed to JavaScript in an ill-fated marketing decision to try to capitalize on the popularity of Java language, when Netscape released its Navigator 2 in 1996.

Java and JavaScript are totally different languages for different programming purposes. However, in the early days, some efforts were made to adopt Java syntax and convention into JavaScript, such that JavaScript seems to be a subset of Java. In reality, they have very little in common.

Client-Side JavaScript by Examples

I shall assume that you know HTML and CSS (read my HTML/CSS articles otherwise). I shall also assume that you understanding some programming basics (computational thinking) such as variables, if-else and for-loop constructs.

Client-side JavaScripts run inside a browser via a built-in JavaScript engine. There are standards on JavaScript. But the Big-5 (Chrome, Firefox, IE/Edge, Safari and Opera), in particular the IE, does not adhere to all the standards strictly. Furthermore, they create their own extensions. Hence, the behavior of JavaScript could be different in different browsers. You may need to test your JavaScripts on more than one browsers.

JavaScript also run standalone (and in the server). To run JavaScript standalone, you need to install Node.js, which is a JavaScript engine. I will present the examples in the next section.

**1  Client-side JS EG 1: Functions alert() and document.write()**

Let us write our first client-side JavaScript to print the message "Hello, world".

Start with a new file and enter the following codes. Do not enter the line numbers, which is used to aid in explanation. Take note that:

* JavaScript is *case sensitive*. A *rose* is NOT a *ROSE* and is NOT a *Rose*.
* "Extra" white spaces (blanks, tabs and newlines) are ignored. That is, multiple white spaces is treated as a single blank character. You could use them liberally to make your program easier to read.

Save the file as "JSExAlertWrite.html" (or any filename that you prefer, with file extension of ".html" or ".htm"). Run the script by loading the HTML file into a JavaScript-capable browser (e.g., One of the BIG FIVE - Chrome, Firefox, Internet Explorer/Edge, Safari or Opera).

<!DOCTYPE html>

<!-- JSExAlertWrite.html -->

<html lang="en">

<head>

<meta charset="utf-8">

<title>JavaScript Example: Functions alert() and document.write()</title>

<script>

alert("Hello, world!");

</script>

</head>

<body>

<h1>My first JavaScript says:</h1>

<script>

document.write("<h2><em>Hello world, again!</em></h2>");

document.write("<p>This document was last modified on "

+ document.lastModified + ".</p>");

</script>

</body>

</html>

How it Works?

1. JavaScripts are programming codes that are embedded inside an HTML document. The codes are contained between a pair of <script> and </script> tags, as follows:
2. <script>
3. // Your JavaScript programming codes here!
4. </script>
5. NOTE: In HTML4/XHTML, you need to include attribute type="text/JavaScript" to the <script> opening tag.
6. You could place the scripts in either the HEAD section (called header script) or BODY section (called body script) of an HTML document. You are free to embed as many scripts into a single document as you like, using multiple <script>...</script> elements. Lines 7-9 and Line 13-17 are two pieces of JavaScripts, placed in the HEAD and BODY sections, respectively.
7. JavaScript statements are terminated by a semicolon ';' (like Java/C/C++/C#).
8. The alert(*str*) function (Line 8) pops out a *dialog box* displaying the *str* and a OK button. Strings are enclosed by a pair of *double quotes* or *single quotes*, e.g., "hello" or 'hello'.
9. The current web page is represented by the so-called document object in the JavaScript. The document.lastModified (Line 16) property stores the last modified date of the current document. The document.write(*str*) function (Line 14 to 16) can be used to write the specified *str* to the current document, as part of the current HTML document.
10. The '+' operator (Line 16) can be used to *concatenates* two strings (like Java).
11. As a result of the document.write(), the BODY section of this document contains:

<h1>My First JavaScript says</h1>

<h2><em>Hello world, again!</em></h2><p>This document was last modified on mm/dd/yyyy hh:mm:ss.</p>

The alert(str) and document.write(str) are some of the commonly-used built-in functions provided in JavaScript.

TRY: Print the document's title and the URL location. (Hints: use document.title and document.location properties.)