Lesson Weekend

Introduction to Programming (/introduction-to-programming) / JavaScript and Web Browsers (/introduction-to-programming/javascript-and-web-browsers) / Welcome to JavaScript and Web Browsers

Text

Welcome to JavaScript and Web Browsers! In this course section, your web pages are going to go from static to dynamic thanks to JavaScript.

JavaScript

Back in 1995, a developer at Netscape named Brendan Eich introduced JavaScript to the world. Despite Java and JavaScript both containing "Java" in their names, they're two distinct languages that have nothing to do with one another. The official name of JavaScript is ECMAScript which gets its name from the international organization that maintains it, European Computer Manufacturer's Association (ECMA) (http://www.ecma-international.org/).

JavaScript is a programming language used to make web pages interactive. Like HTML and CSS, you do not need to install anything to begin writing JavaScript or see it run on your computer. All modern browsers support JavaScript. This means the JavaScript that you write for your website will automatically run when visitors load your web page in their browsers. In fact, JavaScript is the *only* scripting language that all browsers support so if you want a website to have dynamic content, learning JavaScript is a must.

ECMAscript Versions

We'll often reference "ES6" in our curriculum. ES6 means ECMAscript version 6, which came out in 2015 and made a significant number of changes to JavaScript development. It's helpful to know what "ES6" means, but you don't have to worry about knowing which JS feature came with which ES version. At Epicodus and beyond, we'll be using features from all ECMAscript versions.

Markup, Style Sheet, and Programming Languages

JavaScript is the first programming language we will learn at Epicodus. Later on, depending on the track you've signed up for, you will also learn C# or Ruby. In case it isn't clear, HTML, Markdown, and CSS are not programming languages. Let's review the differences:

- **Markup languages** like HTML and Markdown define the format of data, like separating plain text into headings, paragraphs, lists, and so on. Markup languages define the structure of data.
- **Style Sheet languages** like CSS define the presentation of the data, or how it looks. Style sheet languages style data, like defining what color text should be on a website.
- **Programming languages** like JavaScript enable actions and interactivity. Programming languages have a functional purpose, meaning they *do* things, like evaluate math, handle events (like what to do after a user clicks a button), or handle conditional situations (like being able to tell if someone is able to vote based on whether their age is greater than or lesser than 18). In short, programming languages manipulate data and enable us to interact with data.

HTML, CSS, and JavaScript are all standard technologies used in web development.

Mozilla Developer Network Web Docs

To help us write JavaScript, we will learn how to use the Mozilla Developer Network (https://developer.mozilla.org/en-US/) documentation and reference. According to the site, MDN is made for developers, by developers.

> MDN Web Docs (previously known as MDN — the Mozilla Developer Network) is an evolving learning platform for Web technologies and the software that powers the Web, including CSS, HTML, and JavaScript... MDN's mission is simple: provide developers with the information they need to easily build projects on the open Web. If it's an open technology exposed to the Web, we want to document it.

MDN has excellent documentation on JavaScript, including beginner guides, code examples, and detailed references.

Web Browsers

Let's briefly review how web browsers work in order to build more context around programming for the web. A **web browser** is an application that enables you to explore the web. At Epicodus, we use the Chrome browser from Google. Other popular browsers include Firefox from Mozilla, Edge from Microsoft, and Safari from Apple, among many others. It's important to know that each browser has its own tooling, including its own version of JavaScript based on the ECMAscript standards. Yes, that's right. ECMA only sets the standards for JavaScript, and every browser has its own version of JavaScript.

Web Standards

Similar to ECMA setting the standards for JavaScript, there are other organizations that set the standards for the web in general. Collectively these are called **web standards** and they are the rules that define how the web works (including JavaScript). Having web standards is important, because they explicitly direct how companies should create their software so that browsing the web is the same from web browser to web browser. It also levels the playing field for competition in web browsers since the open and international standards make it so that there are no competing proprietary standards.

Web standards include rules for:

- The internet
- HTML, CSS, and the DOM (which we will learn more about soon)
- Accessibility
- JavaScript
- And MORE!

If you are interested in learning more, check out the Mozilla Developer Network's documentation on Web Standards (https://developer.mozilla.org/en-US/docs/Glossary/Web_standards) to see a non-exhaustive list of the organizations that set these rules and more.

How to Make a Web Browser

There's **a lot** that goes into making a web browser. However, two fundamental pieces of software are the browser engine and the JavaScript engine.

JavaScript Engines

A JavaScript engine executes JavaScript source code, the browser's own version of the ECMAscript standards. The engine is built to perform quickly, and also to meet web standards. Notably, JavaScript engines are not just used in browsers, they can also be used in servers and JavaScript compilers. We won't worry about both of those for now.

Browser Engines

A browser engine transforms the HTML and CSS source code into a visual and interactive representation on your computer's screen. It is also built to perform quickly and meet web standards. If you research browser engines online, take note that you'll see hits for "rendering engines" and "layout engines", two processes that are a part of browser engine functionality.

Examples from Major Companies

Let's look at the largest companies' browser and JavaScript engines. Note that you do not need to retain the following information and we're only reviewing it to demonstrate the different technologies that power different web browsers.

- Google's Chrome browser uses the V8 JavaScript engine (https://v8.dev/) along with the Blink browser engine (https://en.wikipedia.org/wiki/Blink_(browser_engine)).
- Mozilla's Firefox browser uses the SpiderMonkey JavaScript engine (https://spidermonkey.dev/) and Gecko (https://en.wikipedia.org/wiki/Gecko_(software)) is their browser engine.
- Apple's Safari browser uses JavaScriptCore (https://trac.webkit.org/wiki/JavaScriptCore#:%7E:text=JavaScriptCore%20is%20the%20built%2Din,SquirrelFish%2 as their JavaScript engine and WebKit (https://webkit.org/) as their browser engine.
- Microsoft's Edge uses Chakra (https://en.wikipedia.org/wiki/Chakra_(JavaScript_engine)) as their JavaScript engine, and EdgeHTML (https://en.wikipedia.org/wiki/EdgeHTML) as their browser engine.

There's a lot of other browser and JavaScript engines out there, including outmoded technologies. To see a list of JavaScript engines on Wikipedia, check out this link (https://en.wikipedia.org/wiki/List_of_ECMAScript_engines). To see a list of browser engines on Wikipedia, check out this link (https://en.wikipedia.org/wiki/Comparison_of_browser_engines).

Browser Compatibility

Even though companies who make web browsers are expected to meet international web standards, that doesn't mean that every browser out there has met all of those standards. Technology is constantly evolving, and new standards are developed regularly. This means that web browser companies need to keep up with the new standards that are released. This also means that some web browsers are behind on implementing new standards, or just plain outdated (like Internet Explorer).

Because of this, web developers need to consider browser compatibility. **Browser compatibility** is the ability for websites to function on multiple different web browsers.

Let's look at an example. Check out this compatibility table (https://kangax.github.io/compat-table/es6/) that describes features of ECMAscript version 6 (ES6) and whether web browsers support those features. This table is complex, showing ES6 features in each row and listing not only desktop browsers in the columns, but mobile browsers, servers, and compilers that use JavaScript. Any box that is colored green means the ES6 feature is supported. If you select the ECMAscript version "2016+" (at the top left of the menu bar (https://kangax.github.io/compat-table/es2016plus/)), you'll be able to see features from later versions of ECMAscript.

It's not necessary to understand the compatibility table linked above, nor are we going to learn the ins and outs of browser compatibility in this program. We are discussing this now to create context on how web browsers work, and how they differ.

Takeaways

This lesson is meant as both an introduction to technologies that you'll work with in this section and as a context-building exercise. You do not have to remember the names of browser engines or every web standard in existence. You also do not need to worry about browser compatibility.

Just remember these items:

- HTML, CSS, and JavaScript are all fundamental web technologies, and all browsers support these.
- Every browser is unique, made by different companies, and operating on different technology.
- All browsers adhere to the same web standards set by international organizations.

As we go along in this section, we'll reference back to the concepts in this lesson.

Previous (/introduction-to-programming/javascript-and-web-browsers/javascript-and-web-browsers-objectives)

Next (/introduction-to-programming/javascript-and-web-browsers/documentation-and-resources)

Lesson 2 of 75
Last updated more than 3 months ago.