Ever wondered how computer programming works, but haven't done anything more complicated on the web than upload a photo to Facebook?

Then you're in the right place.

To someone who's never coded before, the concept of creating a website from scratch -- layout, design, and all -- can seem really intimidating. You might be picturing the Harvard students from *The Social Network*, sitting at their computers with gigantic headphones on and hammering out code, and think to yourself, *I could never do that.*

But you can!

Anyone can learn to code, just like anyone can learn a new language. In fact, programming is kind of like speaking a foreign language -- which is exactly why they're called programming *languages.* Each one has its own rules and syntax that need to be learned step by step. Those rules are ways to tell your computer what to do. More specifically, in web programming, they're ways of telling your *browsers* what to do.

The goal of this post is to, in plain English, teach you with the basics of HTML, CSS, and one of the most common programming languages, JavaScript. But before we begin, let's get an idea of what programming languages actually are.

**A Brief Intro to Programming Languages**

Programming, or coding, is like solving a puzzle. In a human language like French or Arabic, the puzzle might be the translate a sentence perfectly. In programming, the puzzle could be to make a web page look a certain way, or to make an object on the page move.

So, when a web designer is given an end goal like "create a webpage that has this header, this font, these colors, these pictures, and an animated unicorn walking across the screen when users click on this button," the web designer's job is to take that big idea and break it apart into tiny pieces, and then translate these pieces into instructions that the computer can understand -- including putting all these instructions in the correct order or syntax.

Every page on the web that you visit is built using a sequence of separate instructions, one after another. Your browser (Chrome, Firefox, Safari, and so on) is a big actor in translating code into something we can see on our screens and even interact with. It can be easy to forget that code without a browser is just a text file -- it's when you put that text file into a *browser* that the magic happens. When you open a web page, your browser fetches the HTML and other programming languages involved and interprets it.

HTML and CSS are actually not technically programming languages; they're just page structure and style information. But before moving on to JavaScript and other true languages, you need to know the basics of HTML and CSS, as they are on the front end of every web page and application.

In the very early 1990s, HTML was the only language available on the web. Web developers had to painstakingly code static sites, page by page. A lot's changed since then: Now there are many computer programming languages available. In this post, I'll talk about HTML, CSS, and one of the most common programming languages: JavaScript.

**The TL:DR Version**

An overview:

* **HTML**provides the *basic structure* of sites, which is enhanced and modified by other technologies like CSS and JavaScript.
* **CSS** is used to control *presentation, formatting, and layout*.
* **JavaScript** is used to control the *behavior* of different elements.

Now, let's go over each one individually to help you understand the roles each plays on a website and then we'll cover how they fit together. Let's start with good ol' HTML.

**HTML**

HTML is at the core of every web page, regardless the complexity of a site or number of technologies involved. It's an essential skill for any web professional. It's the starting point for anyone learning how to create content for the web. And, luckily for us, it's surprisingly easy to learn.

HTML stands for HyperText Markup Language. The "markup language" part means that, rather than being a programming language that uses a programming language to perform functions, it uses tags to identify content.

Let me show you what I mean. Take a look at the article below. If I were to ask you to label the types of content on the page, you'd probably do pretty well: There's the header at the top, then a subheader below it, the body text, and some images at the bottom followed by a few more bits of text.

Markup languages work in the same way as you just did when you labeled those content types, except they use code to do it -- specifically, they use HTML tags, also known as "elements." These tags have pretty intuitive names: Header tags, paragraph tags, image tags, and so on.

Every web page is made up of a bunch of these HTML tags denoting each type of content on the page. Each type of content on the page is "wrapped" in, i.e. surrounded by, HTML tags.

For example, the words you're reading right now are part of a paragraph. If I were coding this web page from scratch (instead of using the WYSIWG editor in [HubSpot's COS](https://www.hubspot.com/products/content-optimization-system" \t "_blank)), I would have started this paragraph with an opening paragraph tag: <p>. The "tag" part is denoted by open brackets, and the letter "p" tells the computer that we're opening a paragraph instead of some other type of content.

Once a tag has been opened, all of the content that follows is assumed to be part of that tag until you "close" the tag. When the paragraph ends, I'd put a closing paragraph tag: </p>. Notice that closing tags look exactly the same as opening tags, except there is a forward slash after the left angle bracket. Here's an example:

<p>This is a paragraph.</p>

Using HTML, you can add headings, format paragraphs, control line breaks, make lists, emphasize text, create special characters, insert images, create links, build tables, control some styling, and much more.

To learn more about coding in HTML, I recommend [checking out our guide to basic HTML](https://offers.hubspot.com/html-hacks-for-marketers), and using the free classes and resources on [codecademy](http://www.codecademy.com/en/tracks/web" \t "_blank) -- but for now, let's move on to CSS.

## ****CSS****

Whereas HTML was the basic structure of your website, CSS is what gives your entire website its style. Those slick colors, interesting fonts, and background images? All thanks to CSS. It affects the entire mood and tone of a web page, making it an incredibly powerful tool -- and an important skill for web developers to learn. It's also what allows websites to adapt to different screen sizes and device types.

To show you what CSS does to a website, look at the following two screenshots. The first screenshot is one of my colleague's [blog posts](https://blog.hubspot.com/blog/tabid/6307/bid/23454/The-Ultimate-Cheat-Sheet-for-Mastering-LinkedIn.aspx) in Basic HTML, and the second screenshot is that same blog post with HTML and CSS.

#### ****Screenshot with HTML (no CSS):****

Notice all the content is still there, but the visual styling isn't. This is what you might see if the style sheet doesn't load on the website, for whatever reason. Now, here's what the same web page looks like with CSS added.

#### ****Screenshot with HTML + CSS:****

Isn't that prettier?

Put simply, CSS is a list of rules that can assign different properties to HTML tags, either specified to single tags, multiple tags, an entire document, or multiple documents. It exists because, as design elements like fonts and colors were developed, web designers had a lot of trouble adapting HTML to these new features.

You see, HTML, developed back in 1990, was not really intended to show any physical formatting information. It was originally meant only to define a document's structural content, like headers versus paragraphs. HTML outgrew these new design features, and CSS was invented and released in 1996: All formatting could be removed from HTML documents and stored in separate CSS (.css) files.

So, what exactly does CSS stand for? It stands for Cascading Style Sheets -- and "style sheet" refers to the document itself. Ever web browser has a default style sheet, so every web page out there is affected by at least one style sheet -- the default style sheet of whatever browser the web page visitor is using -- regardless whether or not the web designer applies any styles. For example, my browser's default font style is Times New Roman, size 12, so if I visited a web page where the designer didn't apply a style sheet of their own, I would see the web page in Times New Roman, size 12.

Obviously, the vast majority of web pages I visit don't use Times New Roman, size 12 -- that's because the web designers behind those pages started out with a default style sheet that had a default font style, and then they overrode my browser's defaults with custom CSS. That's where the word "cascading" comes into play. Think about a waterfall -- as water cascades down the fall, it hits all the rocks on the way down, but only the rocks at the bottom affect where it will end up flowing. In the same way, the last defined style sheet informs my browser which instructions have precedence.

To learn the specifics of coding in CSS, I'll point you again to the free classes and resources on [codecademy](http://www.codecademy.com/en/tracks/web" \t "_blank). But for now, let's talk a bit about JavaScript.

## ****JavaScript****

JavaScript is a more complicated language than HTML or CSS, and it wasn't released in beta form until 1995. Nowadays, JavaScript is supported by all modern web browsers and is used on almost every site on the web for more powerful and complex functionality.

In short, JavaScript is a programming language that lets web developers design interactive sites. Most of the dynamic behavior you'll see on a web page is thanks to JavaScript, which augments a browser's default controls and behaviors.

One example of JavaScript in action is boxes that pop up on your screen. Think about the last time you entered your information into an online form and a confirm box popped up, asking you to press "OK" or "Cancel" to proceed. That was made possible because of JavaScript -- in the code, you'd find an "if ... else ..." statement that tells the computer to do one thing if the user clicks "OK," and a different thing if the user clicks "Cancel."

Another example of JavaScript in action is a [slide-in call-to-action](https://blog.hubspot.com/marketing/how-to-add-a-slide-in-call-to-action-to-your-blog-posts), like the ones we put on our blog posts, which appears on the bottom right-hand side of your screen when you scroll past the end of the sidebar. Here's what it looks like:

Other uses for JavaScript include security password creation, check forms, interactive games, and special effects. It's also used to build mobile apps and create server-based applications. You can add JavaScript to an HTML document by adding these "scripts," or snippets of JavaScript code, into your document's header or body.

If you want to learn more about JavaScript, [codecademy has free online courses](http://www.codecademy.com/en/tracks/javascript" \t "_blank) you can take, too.

The hardest part of coding is getting started -- but once you learn the basics, it'll be easier to learn more advanced programming languages. And if you ever feel overwhelmed, head over to [hackertyper.net](http://hackertyper.net/) and hit random keys on your keyboard. Trust me, you'll feel better.