**The Internet and Web Development**

**The Ever-Expanding Network**

So how did the internet start? In 1969, the United States Department of Defense funded the creation of ARPANET, a precursor network to the internet. ARPANET stands for Advanced Research Projects Agency Network. ARPANET connected supercomputing centers run by government agencies and universities.

These institutions wanted to connect their individual networks for large-scale information transfer. However, many of them followed different standards and technical implementations. In the 1970s, the transmission control protocol and internet protocol, otherwise known as TCP/IP, were created to provide standards around the transfer of data that would allow these early networks to communicate with each other.

TCP/IP was researched and specified throughout the 1970s and adopted in the early 1980s. As different networks adopted TCP/IP, the interconnected global network of networks that is today known as the internet was formed.

A map of the united states

AI-generated content may be incorrect.

**The World Wide Web**

While people today often use the terms internet and world wide web interchangeably, they actually refer to quite different things.

The internet refers to the actual network of connected computing devices. Although the internet was around in the 1980s, there was not an intuitive way for most people to browse the internet. The internet just sent messages produced by one computer and presented them to another computer.

Engagement with the internet changed in 1989 when Tim Berners-Lee

invented the world wide web. The world wide web is a collection of interlinked websites and other web resources. The world wide web, in combination with the rise of web browsers in the 1990s, introduced a user-friendly interface that enabled users to browse multimedia content and interact with other users.

The invention of the world wide web led to the use of the internet in wider society through the 1990s and the creation of a variety of websites that are still in use today.

**Browsers and Servers**

As we’ve seen, the internet is a network that links computer devices worldwide, enabling people to share information with one another despite vast distances. But how is information sent from one device to another?

One way of understanding this process is to look at the client-server model. In this model, the client refers to the user’s device or program that is making a request for data. A client can be a browser or application running on a user’s laptop, smartphone, or tablet.

The server is the device or program in that network that waits for incoming requests and sends back data. This might be an in-house server, a rented server at a data center, or cloud server. At Codecademy, we have servers that store lesson data and our servers are sending this lesson data to your client device.

**404 Status Code**

Let’s take a deeper dive into the client-server model through exploring a part of HTTP that you’ve probably seen before: HTTP status codes.

When a server responds to a client, the server specifies a status code as a part of the response. Status codes indicate whether or not the HTTP request was successfully completed and if there was an error, they contain some information about the type of error that happened. The status code helps the browser know how to handle the data that was sent back with the response.

Review the HTTP statuses below and see if any of them seem familiar.

A white rectangular box with black text

AI-generated content may be incorrect.

**How Do Browsers Work?**

So far we’ve seen how a single request and response are handled between a client and a server. But most of the time, our devices aren’t making a single request. Every time we load a webpage, our device sends a request for each file that makes up that page. So even when we’re just loading one webpage, that page can make multiple requests in order to retrieve different pieces of content, like images.

So how does this process work when we’re making multiple requests simultaneously?

All of the following steps happen in a split second:

When a user types in a URL and presses enter, the server processes the request and sends the HTML file back to the client. HTML files hold the content of a website and they also contain links for any additional assets or code that are needed to display the site properly.

The browser will begin to search for elements in the HTML file and it will start to make additional HTTP requests for any other external resources used by the HTML file. This often includes:

One or more CSS stylesheets. CSS stands for cascading style sheets; CSS creates the style and layout of a web page. The browser will request the CSS stylesheet, and when the server sends it back, the browser analyzes the CSS and starts applying the visual styles to the content of the site.

The request-response cycle also sends website assets, like images and videos, from the server to the browser. If these files are large, there might even be a noticeable delay before they are rendered by the browser.

One or more JavaScript files. JavaScript makes the webpage interactive. This programming language functions as the “behavior” of the web page. A webpage that does not use JavaScript is known as a static webpage.

In most modern browsers, these additional requests are made in parallel. This means that these requests are initiated at the same time, and the browser does not wait to receive one resource before requesting the next resource.

All of the resources are typically displayed as soon as possible. The HTML will be displayed even if some of the other assets have not been received by the browser.

Voila! The user can now interact with the website that they requested to see. This whole process typically happens in about a second or less, depending on the speed of the user’s connection, the size of the website, and even the physical distance between the browser and the server.

**Web 2.0**

Now that we’ve covered some of the basics of how the internet works, let’s check out some trends that are fundamental to the emergence of modern web development and modern JavaScript.

The earliest static websites were composed of text, images, and links, with very little interactivity beyond browsing from one page to another. These websites are called static, which means lacking in movement because they do not change based on user behavior. As internet connection speeds and web technologies progressed, more complex interactions became possible on the web.

A collection of advances in the early 2000s created a cluster of web applications that are often called “Web 2.0”. In comparison to early static websites, Web 2.0 applications are often defined by:

Providing a dynamic user experience by offering content that responds to user input without forcing the page to reload. In the early web, user input would typically take the user to a new page — and they would have to wait for the new page to load. In Web 2.0, websites could just update selected regions of the page, avoiding the interruption caused by reloading.

Emphasizing user-generated content and social sharing. In the early web, content was generally authored by a single source. The rise of blogging, social media, and wikis in web 2.0 meant that users could generate content and share it with their friends.

There were important technical advances that enabled each of these advances in the user interface of the internet. For example:

jQuery was the first JavaScript framework that could fetch data while the web page is running.

The rise of web frameworks that connected to databases, like Spring, Django, and Ruby-on-Rails, enabled user-generated content to effectively be created, stored, and displayed.

**Current Internet Trends**

The rise of internet-connected smartphones has profoundly changed how users interact with the internet. Mobile internet traffic now accounts for more than half of all internet traffic and web development practices have evolved in order to provide a good user experience regardless of device type.

**Responsive Web Design**

The rise of responsive web design has changed how websites are built. Responsive web design was enabled by additions to the CSS language, like media queries and relative units. These additions allow the presentation of websites to adjust based on the size of the window in which they are displayed.

**Mobile Applications and Devices**

The rise of internet-connected mobile applications has changed the way that we think about browsing the internet. Users accessing the internet on smartphones are likely to spend much more time with specific applications, rather than using their phone’s browser.

Though most mobile applications are internet connected, they are not part of the world wide web. The web is built out of links, whereas mobile applications are designed to keep the user’s attention.

If you want to learn about mobile development, web development is a great place to start! While the majority of mobile applications are built in programming languages, like Swift for iOS, it is increasingly common to see developers using JavaScript frameworks to build new apps.

**Languages for Web Development**

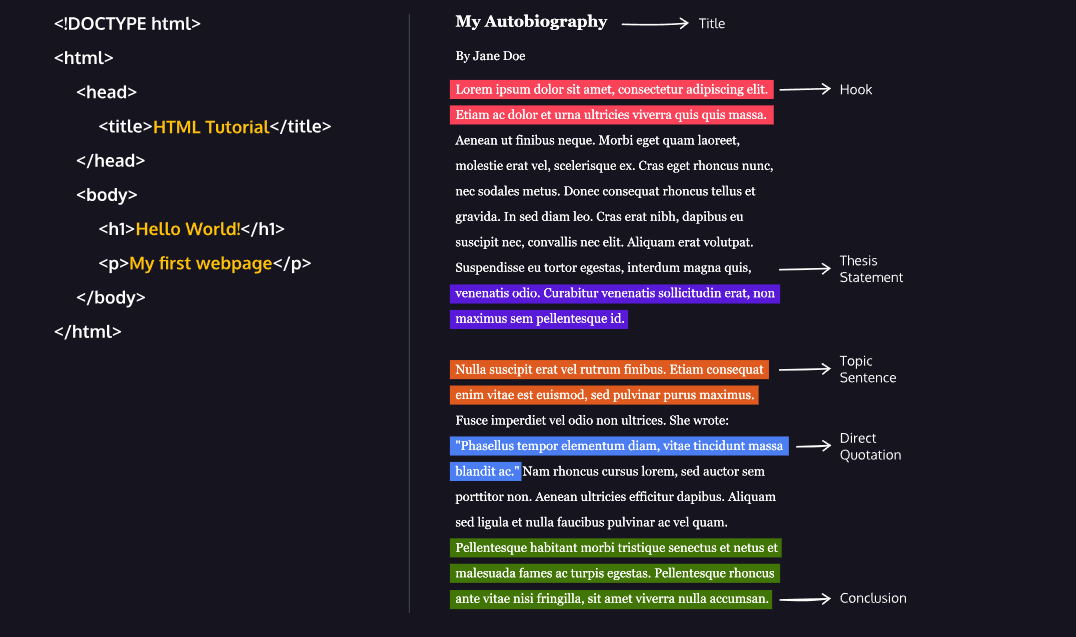
What is HTML?

HTML stands for hypertext markup language. HTML is the skeleton of all web pages. It provides structure to the content on a website, including text, images, buttons, videos, and more.

HTML is a practical place to start learning to code. You can build basic websites after learning just a little HTML, with text, images, and videos. You can always open up your work-in-progress website with your browser and see what you’re building.

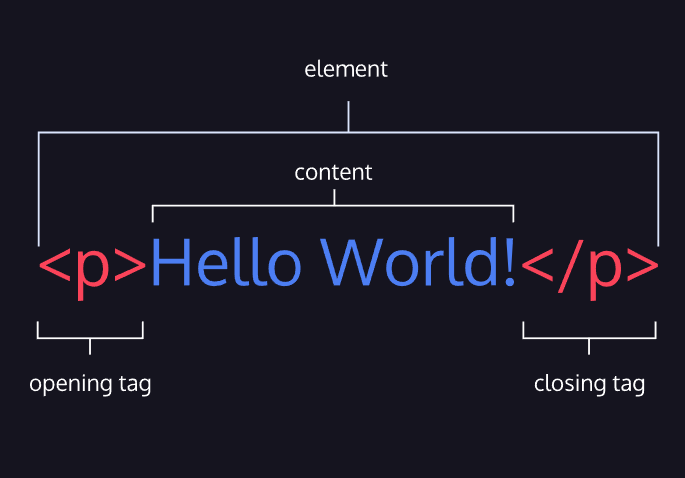
The ML in HTML stands for markup language. Markup refers to a way of annotating text that is distinguishable from the text itself. The same way that a teacher might “markup” a student essay by underlining topic sentences and circling spelling errors, HTML annotates the content within a web page.

A teacher might use a red pen to make sure that their comments are easy to distinguish from the student’s own work. HTML separates content and annotation by using HTML tags, which are denoted by angle brackets (also known as less-than and greater-than signs).

**HTML Elements**

HTML tags are the “markup” for HTML. They are annotations that provide information about the type of content they contain. Let’s take a close look at the syntax for how HTML tags surround content to create an HTML element. The diagram displays an HTML paragraph element. As we can see, the paragraph element is made up of one opening tag (<p>), the content (“Hello World!” text), and a closing tag (</p>).

Let’s quickly review each part of the element pictured:



HTML element — a unit of content in an HTML document formed by HTML tags and the text or media it contains.

Opening Tag — the element name used to start an HTML element. The tag type is surrounded by opening and closing angle brackets.

Content — The information (text or other elements) contained between the opening and closing tags of an HTML element.

Closing tag — the second HTML tag used to end an HTML element. Closing tags have a forward slash (/) inside of them, directly after the left angle bracket.

**Hypertext and the World Wide Web**

The H and T in HTML stands for hypertext. Hypertext is text that is linked to other text. This diagram shows different websites that are connected to each other through links, which are represented by arrows.

What’s so hyper about hypertext? The prefix hyper indicates that the text expands beyond the traditional constraints of the written word. Instead of reading documents from beginning to end, like you would read a book, someone going through hypertext can browse. By clicking on links from one document to another, the user can navigate to whatever page they find the most interesting and carve their own unique path through the web.

Many modern websites provide rich user experiences, with features like embedded videos, animations, and interactivity, so it doesn’t necessarily feel like you are just navigating from one HTML page to the next. But despite all of the advances that have taken place with the growth of the web, at its core the web is still just a collection of hyperlinked documents.

**Adding Hyperlinks**

An attribute in HTML provides additional information about an HTML element. It comes in a name and value pair with the structure name="value". For example, you can specify the width of an HTML element with the attribute width="500".

Links are created in HTML with something called the href attribute, which stands for hyperlink reference. The href attribute allows us to specify the URL, or address on the web, that a link should take users to. See below for an example of the href attribute in action.

When we set the href attribute on an anchor tag (abbreviated to <a>) we can specify both the text that should be rendered for the user (the text within the anchor tag) and the URL that the browser should navigate to.

With this code, we’re assigning the value www.codecademy.com to the href attribute. When a user clicks on the text of this link (Learn to code!), they will be directed to [www.codecademy.com](http://www.codecademy.com/).

<a href="www.codecademy.com">Learn to code!</a>

**What is CSS?**

CSS is the language that provides style to the content of an HTML page. This includes colors, fonts, positioning, layout, and more!

Why do we need a separate language for content and presentation? This is an example of the computer science principle separation of concerns. Large codebases are kept maintainable when each section has clearly differentiated problems that it is trying to solve.

**Applying CSS**

CSS contains selectors that specify the HTML elements to which the style should be applied as well as visual rules that specify how that content should be displayed.

Now it’s time to use the HTML link element to apply the CSS file to her existing website. This is done with an HTML link tag. An HTML link tag is often used to create a connection between an HTML file and the CSS file and tells the browser to apply the CSS styles to the HTML.

**What is JavaScript?**

Any website that provides more than just static information probably utilizes JavaScript in some way. Here are some familiar examples of website features most often built with JavaScript:

popup ads

animated graphics (2D or 3D)

interactive audio and video

maps and other features access the user’s geographic location

and much more!

One of the defining features of JavaScript is its ability to respond to browser events. These events happen in real time and can be triggered by a wide variety of user interactions, including:

the user clicking on a button

the user pressing enter on their keyboard

a video file finishing loading

the user re-sizing their window

the user hovering over text on the webpage

**JavaScript Functions**

Functions allow us to write a chunk of code once that can be reused over and over.

Events allow JavaScript to respond to user behaviors, like the user hovering their mouse over an HTML element or resizing their window.

Events and functions combine to give JavaScript the ability to create interactive experiences. When an event is fired, a function is executed. This pattern is used again and again in web development to create interactive websites.

**What is SQL?**

SQL stands for structured query language. SQL stores information in tables, which is simply a collection of information organized into rows and columns. If you’ve worked with a spreadsheet like Microsoft Excel or Google Sheets, you might be familiar with working with tables. To get the information, you would write a query, or a program that would retrieve data from the table. Here’s an example of a query that would select all of the columns —the \* keyword is a shorthand for “all”— from the page\_views table:

SELECT \* FROM page\_views;

Web developers and software engineers also use SQL to build apps that can save, modify, and access data. There’s even a growing field of data engineering, which is a specialized subset of software engineers who ensure that the applications they are working with accurately and efficiently record all of the required data.