Introduction to Web Developement

What is a full stack?

A 'full stack' is a set of technologies required to be used to run an application. There are 3 major parts to a full-stack:

- **Front End** - HTML, CSS, JavaScript, usually full stack abbreviations indicate the JavaScript frameworks used, but not necessary to include.
- **Back End** - Python, Ruby, C#, Java, PHP, usually specifying the programming language use and/or the framework for the language.
- **Database** - SQL, MongoDB, CouchDB, Redis, specifying the database used for the stack, but not required.

Full stack is referring to the series of technologies used in a 'stack', but the stack name may not directly indicate all technologies. That said, it's good to know that a full-stack developer, is one that can build a full web application. They don't need a database person, or a back-end person, or a front-end person to get their project done.

The Big Three

Often times, when referring to the core languages of the internet, we talk about 'the big three'. These three technologies are present in just about every application you encounter on the web. Let's recap:

- **HTML** represents the **content** and the structure. We can think of this as the skeleton of a webpage. Elements are first placed into the Document Object Model, or the DOM, so that data can be represented on the browser.
- **CSS** represents the **style** and positioning of our HTML elements. We can think of this as the skin and clothes, the visual side of our website. Things like color, font, sizing, and positioning are all controlled in part by CSS.
- **JavaScript** is the action. We can think of it as the **behavior** of our website. You can build beautiful static websites with HTML & CSS, but they're not actually functional until we add logic. JS allows us to interact with our HTML & CSS by dynamically manipulating the DOM.

Why HTML?

Hyper **T**ext **M**arkup **L**anguage, or **HTML**, is one of the three core technologies that power almost every single website on the internet. HTML allows web designers and developers to define the structure of their websites. Using HTML, developers decide where certain elements like text or images should appear on a web page.

You will find the HTML section, please go through the HTML Tutorial, and finish it before the starting date.

💻**Click on following link to start learning! Explore through HTML Tutorial and practice yourself!**

What is CSS?

CSS is what's called a **C**ascading **S**tyle **S**heet language, and is used to stylize elements written in a markup language such as HTML. It separates the content from the visual representation of the site.

CSS advantages

The difference between a website which implements CSS and one that doesn't is massive and surely noticeable.

You might have seen a website that fails to load completely and has a white background with most of the text being blue and black. This means that the CSS part of the site didn't load correctly or it doesn't exist altogether. That's how a site with only HTML looks like, and I think you'd agree that that's not very appealing. Before using CSS, all of the stylizing had to be included into the HTML markup. This means you had to separately describe all the background, font colors, alignments, etc.

CSS lets you stylize everything on a different file, thus creating the style there and later on integrating the CSS file on top of the HTML markup. This makes the actual HTML markup much cleaner and easier to maintain.

In short, with CSS you don't need to repeatedly describe how individual elements look. This saves time, shortens the code and makes it not as prone to errors. CSS lets you have multiple styles on one HTML page, therefore making the customization possibilities almost endless. Nowadays, this is becoming more a necessity than a commodity.

📹Let us have a look in this video: What is CSS? Why Use CSS? What Does CSS Do? Advantages and Disadvantages of CSS ? - YouTube

💻**Click on here to start learning:** CSS Tutorial Feel free to practice and explore!

Try it Yourself »

Clean Code

"Any fool can write code that a computer can understand. Good programmers write code that humans can understand."

- Martin Fowler

Writing clean, understandable, and maintainable code is a skill that is crucial for every developer to master. The most popular definition of *clean code is code that is easy to understand and easy to change*.

- **Use Comments Sparingly**: to explain what a line of code is doing and why a piece of code is written the way it is.
- **Naming Convention:** One glance at any clean coding handbook and you'll read about the importance of using a naming convention to help you identify variables at-a-glance.

Tips for better Naming Convention:

- Do not use single-character variables.
- Do not abbreviate variable names.
- Always use nouns or noun pairs.
- Always make sure they're easy to search and pronounce.

📜Please check this link to know: Why Clean Code?