Getting Started with Basics

Welcome to the Course

Hi! Welcome to the course. This course will help you discover web development and learn to build your own interactive websites using advanced front-end and back-end technologies like Node JS, Bootstrap, MongoDB, etc.

You'll also build several projects along the course, which will reinforce and strengthen your skills.

This course is divided into two modules:

1. **Front-end Web Development** – Here you'll start from the pure basics, understanding basic principles of user interface design and user experience.

   You'll construct fresh, appealing, and responsive websites with HTML, CSS, JavaScript, jQuery, and Bootstrap. This part of the course focuses on core concepts, along with laying a strong programming foundation.

2. **Back-end Web Development** – During this part, you'll learn how to build and maintain the technologies that power the administrative side of websites.

   It's a deep-dive that will help you understand a web application's operation with databases to evolve simple, static websites into dynamic, database-driven web applications. You'll go through a wide range of topics like NodeJS, event handling, file uploads, social authentication, deployment, etc.

History of Web

Tim Berners-Lee created HTML in late 1991 but didn't release it officially. He published it later in 1995 as HTML 2.0. The idea behind creating the Web was to create a service that helps everyone communicate, share, and receive information. Then came HTML 4 which served as a major version of HTML. HTML has evolved very much and received various updates since its creation. With each version, the creation of webpages got easier and stylish.

HTML is generally used to design the basic **structure** of the webpages, which will be then improved using other technologies like CSS and JavaScript. **CSS** controls the **looks**, feel, layout and formatting, whereas **JavaScript** is used to control different elements' **behaviour** on the page.

For more information, you can have a look at the sites below:

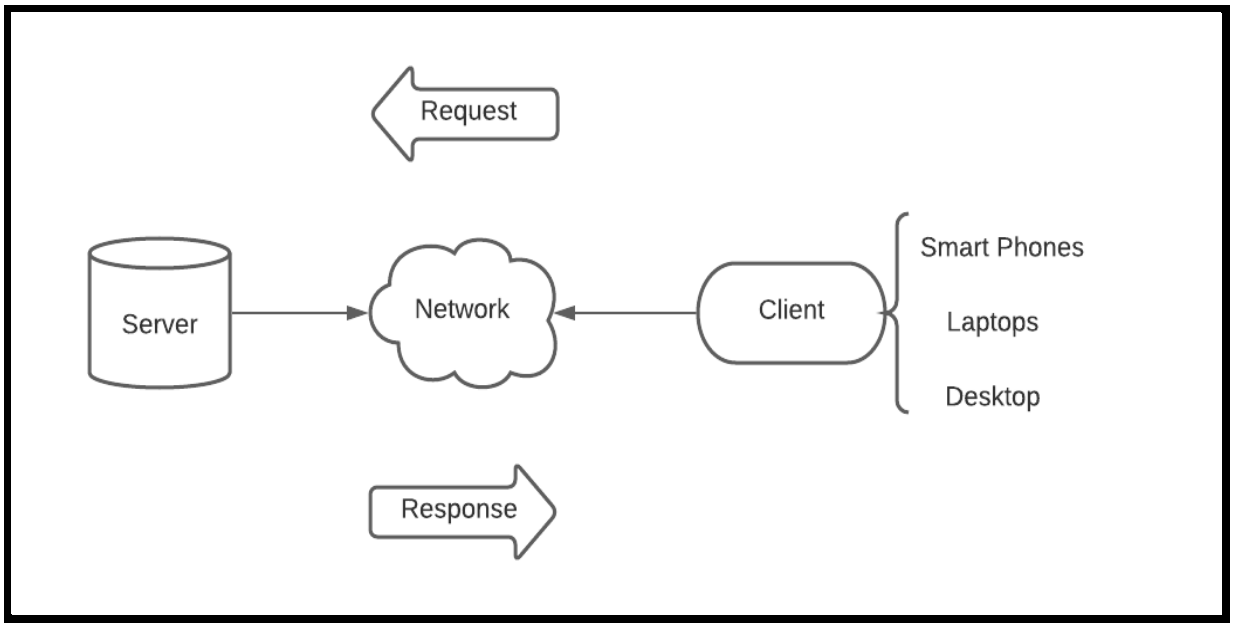
- https://webfoundation.org/about/vision/history-of-the-web/
- https://www.sitesbay.com/css/css-history
- https://auth0.com/blog/a-brief-history-of-javascript/

Client-Server Architecture

Here, we're going to take a look into the Client-Server Architecture, and see how the internet works.

See, like in the outside world, the **Client** means any person or organization, who uses a particular service. Similarly, in the world of computer networks, a **Client** is a host (any computer or a device that is connected to a computer network) that receives some service from certain service providers (**Servers**).

As the word suggests, a **Server** is a remote computer that provides some service/information to its clients. So, basically, a Client requests some information, and the Server responds by serving it to the Client. Servers store and retrieve the information from the databases.



The client can be any device like a mobile phone, laptop, desktop, etc.

Front end and Back end

There are two faces of a website: front-end and back-end. These are the most popular and crucial topics in web development.

Front-end: The front-end is the part of a website with which the users interact directly. It's also called the 'client-side' of a website. It includes everything the users see: text, colors, styles, images, buttons, menus, etc. HTML, CSS, and JavaScript are the tools generally used for frontend development. Front-end developers build the front-end part of a website.
Being a front-end developer, you will have to ensure that your websites look good on all devices for a smooth user experience.

There are many front-end frameworks and libraries which developers use, like React.js, jQuery, Angular.js, etc.

Backend: Back-end is the server-side of a website. A back-end web-developer uses it to store and manage data and make sure that the client-side of the website works without any issues. The website's users or clients cannot see and interact with the back-end part of a website. The backend part of the software is abstracted from the users.

Every line of code you write for the backend will be used on the frontend side anyways. In simple words, we can say that everything that happens in the background can be credited to the backend.
Like frontend, there are many backend libraries that developers use like Express, Rails, Django, etc.

Static Vs Dynamic Websites: A static website is a website that displays the same content to each user, and it's usually written using simple HTML and CSS.
Whereas a dynamic website displays different content to different users depending on user data and preferences and provides user interaction. In addition to HTML, making dynamic websites require advanced technologies, frameworks, and databases to store user data.

Usually, when static websites run on a browser, the content shown is the same for every person accessing the website. An excellent example of a static website would be a simple Blog page.

A dynamic website on the other hand is more functional. Here the users are required to interact with the website as user-interaction plays a big role in dynamic websites.
As the website is dynamic, the content shown on the website will vary according to specific users. The information shown on the page will not be the same like Facebook, where the content is relevant and related to the specific user.

What happens when you visit a website?

Here, we'll try to explain what happens when you try to visit a website.

Every website has its own IP address, using which we can access a particular website. But, as humans, we are **not** good at remembering numbers, we use **domain names**, which is a **user-friendly** way to access a website's IP address. As the user enters the URL of the website, the browser sends a request for the domain name of the website.

The DNS (Domain Name Server), which is like a phonebook of the internet, connects domain names with IP addresses and responds with the web server's IP address. Then the browser sends an HTTP request to the web server's IP (which is provided by DNS). The Server sends over the website's necessary files, which are then correctly rendered by the browser and displayed as a website.