# Web Development Fundamentals

## 1. Difference between Frontend, Backend, and Full-Stack Development

Frontend development deals with the visual and interactive part of a website that users directly interact with. It includes everything a user experiences in their browser such as layouts, colors, fonts, buttons, and navigation. Frontend developers use HTML for structure, CSS for styling, and JavaScript for interactivity. For example, when you visit Amazon, the product listings, search bar, and add-to-cart button are all part of the frontend.  
  
Backend development focuses on what happens behind the scenes — the logic, databases, and server communication. It handles user requests, data storage, and business rules using languages like Python, Java, Node.js, or PHP. For example, when you log in to your account, the backend checks your credentials and retrieves your data.  
  
Full-stack developers work on both frontend and backend parts of the web application. They understand how data flows from the database to the user interface. For instance, a full-stack developer building a food delivery app can design the user interface, manage orders in the database, and handle payments on the backend.

## 2. Client-Server Model Diagram

Here’s a simple representation of how the client-server model works in web architecture:  
  
Client (Browser) → Internet → Web Server → Application Logic → Database → Back to Client  
  
Example flow:  
1. The client sends a request for a webpage (e.g., www.example.com).  
2. The server processes the request.  
3. The database provides the required data.  
4. The server sends the response back to the client.

## 3. How a Browser Requests and Displays a Web Page

When you type a website URL in a browser, the browser sends a request to the web server using HTTP or HTTPS. The server then finds the requested page and sends it back to the browser. The browser receives the HTML file, then downloads related resources like CSS, JavaScript, and images. It renders the page step by step: first structure (HTML), then styling (CSS), and finally interactivity (JavaScript). This process is known as rendering or page loading.

## 4. Tools Required to Set Up a Web Development Environment

1. \*\*Code Editor (e.g., VS Code)\*\* – Used to write and edit code efficiently.  
2. \*\*Web Browser (e.g., Chrome, Firefox)\*\* – For testing and previewing web pages.  
3. \*\*Node.js & npm\*\* – To run JavaScript outside the browser and manage packages.  
4. \*\*Version Control (Git)\*\* – To manage code versions and collaborate with others.  
5. \*\*Live Server Extension\*\* – For real-time browser reloading during development.  
6. \*\*Database (e.g., MySQL, MongoDB)\*\* – To store and manage application data.

## 5. What is a Web Server?

A web server is software or hardware that delivers web content to users over the internet. It stores website files (HTML, CSS, images, etc.) and serves them when requested by clients. Popular web servers include:  
- \*\*Apache HTTP Server\*\* – Open-source and widely used.  
- \*\*Nginx\*\* – Known for speed and handling high traffic.  
- \*\*Microsoft IIS\*\* – Integrated with Windows systems.  
- \*\*LiteSpeed\*\* – Efficient and lightweight alternative.

## 6. Roles in a Web Development Project

- \*\*Frontend Developer:\*\* Designs and implements the visual layout and user interactions. They ensure the website looks appealing and works on all devices.  
- \*\*Backend Developer:\*\* Manages the server-side logic, APIs, and database interactions. They make sure data flows correctly between the client and server.  
- \*\*Database Administrator (DBA):\*\* Handles the design, security, and maintenance of databases. They ensure data integrity, backups, and performance optimization.

## 7. Installing and Configuring VS Code

To set up VS Code for HTML, CSS, and JavaScript development:  
1. Download and install Visual Studio Code from https://code.visualstudio.com.  
2. Install extensions like ‘Live Server’, ‘Prettier’, and ‘JavaScript (ES6) snippets’.  
3. Create a project folder and open it in VS Code.  
4. Create HTML, CSS, and JS files for your project.  
  
After setting up, take a screenshot of your workspace showing all three files open and Live Server running.

## 8. Difference between Static and Dynamic Websites

A static website displays fixed content that doesn’t change unless the developer manually updates it. These sites are fast and simple, suitable for portfolios or company profiles. Example: A personal portfolio website.  
  
A dynamic website generates content based on user interaction or data from a database. They use server-side scripting languages like PHP or Node.js. Example: Facebook or YouTube, where content changes for each user.

## 9. Common Web Browsers and Rendering Engines

Five popular web browsers are:  
1. \*\*Google Chrome\*\* – Uses Blink rendering engine.  
2. \*\*Mozilla Firefox\*\* – Uses Gecko engine.  
3. \*\*Microsoft Edge\*\* – Uses Blink (formerly EdgeHTML).  
4. \*\*Apple Safari\*\* – Uses WebKit engine.  
5. \*\*Opera\*\* – Also uses Blink engine.  
  
Rendering engines are responsible for converting HTML, CSS, and JavaScript into visual content. Each engine may interpret and optimize elements differently, which is why websites may look slightly different across browsers.

## 10. Basic Web Architecture Flow

The basic web architecture involves four main components:  
  
1. \*\*Client (Browser):\*\* Sends requests and displays the results.  
2. \*\*Server:\*\* Processes requests and executes logic.  
3. \*\*Database:\*\* Stores and retrieves application data.  
4. \*\*API:\*\* Acts as a bridge for data exchange between server and client.  
  
Flow: Client → API Request → Server → Database → Response → Client  
  
This structure ensures modularity and smooth communication between different parts of a web application.