

WEEK 4 - PART 2

WEB DEV 101 -

FRONT-END, BACK-END & CLIENT-SERVER MODEL



LINNAEUS
UNIVERSITY
SUMMER 2024

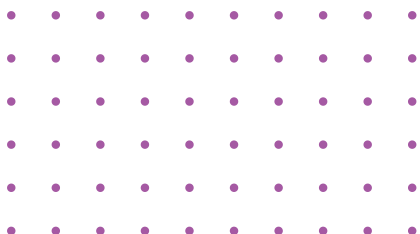
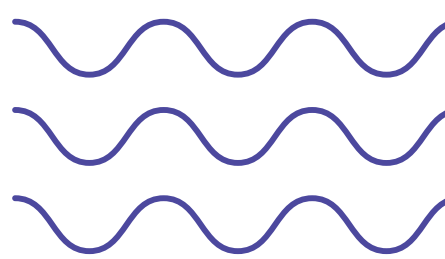


TABLE OF CONTENTS

3	what is web development?	
5	front-end & back-end	
	how it works	5
7	client-server vs front-end/back-end	
	client-server	7
	front-end in detail	8
	back-end in detail	9
	putting it all together	10
	HTTP, HTTPS & communication	11
	front-end vs back-end development	12
	server-side rendering vs front-end rendering	13
	json	15
17	the big 3 of client side programming	

html	18
css	19
javascript	20



WHAT IS WEB DEVELOPMENT?

Web development is the process of building and maintaining websites and web applications. The web operates through a series of interactions between your browser (the front-end) and a server (the back-end).

Front-End development - focuses on what users see and interact with in their web browsers. Front-end development involves creating the visual

layout and user interface of a website or web application. For example, front-end developers design the look and feel of a website, including its colors, fonts, and layout, and add interactive elements like buttons and forms.

Back-End development focuses on the server-side of web development, where the behind-the-scenes processes occur. Back-end development involves managing databases, server code, and application integration. It ensures that data is correctly processed and stored, and that it flows seamlessly to and from the front-end.




A website is a collection of web pages that users can access through a web browser. It's designed to provide information, services, or interactive experiences to users. It provides the visual and interactive elements that users interact with, such as text, images, forms, and buttons. Websites display content and features that users engage with, such as reading articles, watching videos, or filling out forms.

Imagine a website as a shopfront. The shopfront (website) displays various products (content) that you can browse and interact with, like looking at items in a window or asking for help.

When users interact with a website, data flows back and forth between their web browser and the server.

A server is a powerful computer or software system that stores, manages, and processes data. It acts as the central hub that provides resources, services, or data to other computers (clients) over a network, such as the internet.





The server stores various types of data, like user profiles, articles, images, and other content. When a web browser makes a request, the server processes this request, performs necessary operations, and prepares the data to send back.

Think of a server as a library. When you want a specific book (data), you ask the librarian (server). The librarian then retrieves the book from the shelves (database) and hands it to you.

Server code is stored on the server's hard drive, much like how files are saved on your own computer.

Servers can be of two main types - physical servers or virtual servers.

Physical servers are real, tangible computer housed in a data center, which is a large facility full of computers.

Virtual servers are virtual machines that operate within another physical computer. It's akin to having a mini-computer inside a larger one, and it is often hosted in the cloud—a vast network of powerful computers accessible over the internet.

In essence, server code is saved on these servers—whether physical or virtual—that are specifically designed to manage and process requests from websites and applications.



FRONT-END & BACK-END

HOW IT WORKS

Interaction - You interact with the website (e.g., clicking a button).

Request - Website sends a request to the server (e.g., showing a message).

Processing - The server processes the request and fetches the needed data.

Response - Server sends the data back to the website.

Display - Website updates and displays the data for you to see.

1

User Interaction with the Browser (front-end)

You, as the user, interact with a website through your web browser.

This interaction involves:

- Navigating: You enter a URL or click on links to visit different web pages.
- Submitting Forms: You might fill out forms, such as login credentials or search queries.
- Clicking Buttons: Actions like clicking "Submit," "Add to Cart," or "Like" trigger events.

Example: You visit an online store, choose a product, and click the "Add to Cart" button.

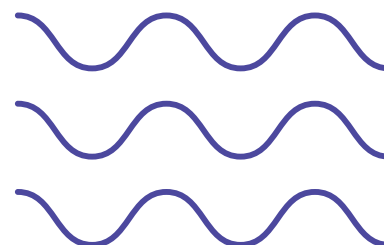
2

Sending Data to the Server (back-end)

When you perform an action on the front-end, your browser sends a request to the server. This request might include:

- Form Data: Information you've entered into a form (like your username and password).
- Requests for Information: Such as asking for product details or searching for items.

Example: Clicking "Add to Cart" sends a request to the server with details about the product you want to add.



3

Server Processing (back-end)

The server receives your request and performs various tasks:

- **Data Retrieval:** It may look up information in a database.
- For example, if you're logging in, the server checks if your username and password match what's stored in the database.
- **Data Manipulation:** The server may update or process data based on your request. For instance, adding a product to your cart requires updating your cart's contents in the database.
- **Business Logic:** The server executes any required logic or calculations, like computing the total price of items in your cart.

Example: The server checks the database to confirm your login credentials and updates your shopping cart with the selected product.

4

Sending Data Back to the Browser

After processing your request, the server sends the relevant data back to your browser. This data includes:

- **Updated Content:** Such as a confirmation of your successful login or an updated cart summary.
- **New Web Pages:** The server might send a new page or update the existing one with the requested information.

Example: After adding an item to your cart, the server sends back an updated cart summary showing the new product and total price.

5

Updating the front-end

The browser receives the server's response and updates the user interface accordingly:

- **Display Changes:** The web page or application reflects the changes, such as showing your updated cart or redirecting you to a new page.
- **Interactive Feedback:** Any feedback from the server, like success messages or error alerts, is shown to you.

Example: Your browser now displays a notification that the product has been successfully added to your cart, and the cart icon is updated with the new item.

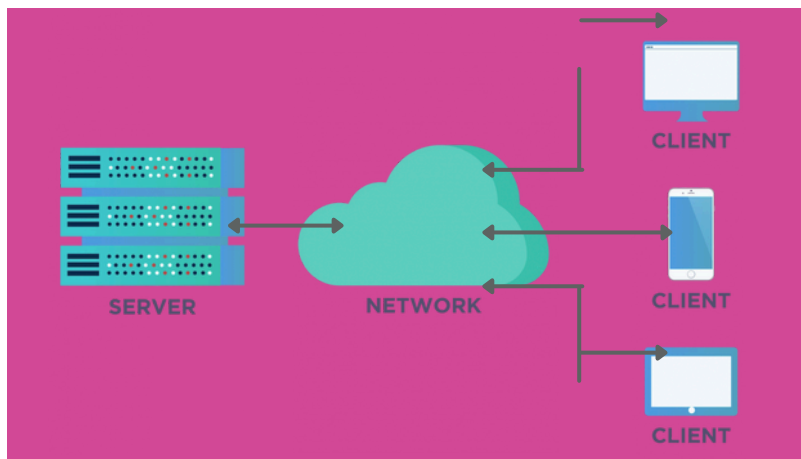


CLIENT-SERVER VS FRONT-END / BACK-END

CLIENT-SERVER

The client-server model is a way of organizing how data and services are provided across a network, like the internet. The model defines how different parts of a system interact over a network.

In computing, a "client" refers to any device or software that is responsible for sending requests for services or resources from a server. It is the component that interacts with the user, often through a web browser or app on a device. For instance, when you enter your username and password and click "Log In," your browser (the client) sends this information to the server. The client is responsible displaying the results to the user.



The server is a powerful computer that stores and processes data. It receives requests from clients, handles them by processing or retrieving data (such as querying a database), and sends back the appropriate information or responses. For example, when the server receives your login credentials, it checks them against its stored data and sends back a confirmation message or an error, depending on the result.

FRONT-END IN DETAIL

The terms front-end and back-end refer to different aspects of web development and the architecture of web applications.

When discussing a web application's architecture, we are talking about the structured design and organization of various components and how they work together to deliver functionality to users. This architecture essentially serves as the blueprint for how a web application is built and operates.

The front-end is the part of a web application or website that users interact with directly.

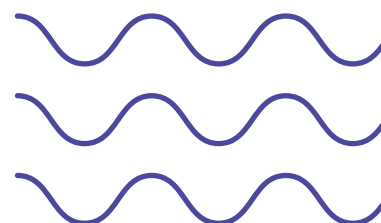
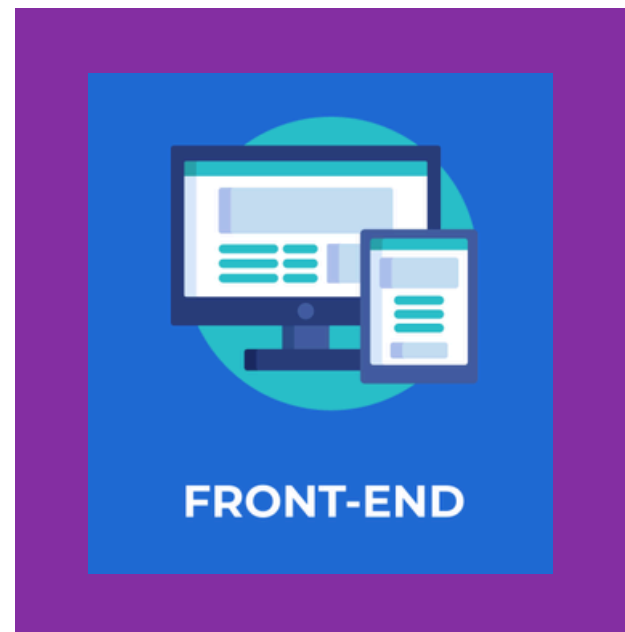
It encompasses everything visible in the browser, including layout, design, and user interface elements.

Technologies used in the front-end include HTML, CSS, and JavaScript. The front-end's primary role is to provide the user interface and handle user interactions.

In a web application, the front-end sends requests to the back-end.

The front-end, which is responsible for the user interface and user interactions, **collects data and triggers requests based on user actions. These requests are sent to the back-end, where they are processed.**

The back-end performs the necessary operations, such as retrieving or updating data, and **sends the results back to the front-end to be displayed to the user.**

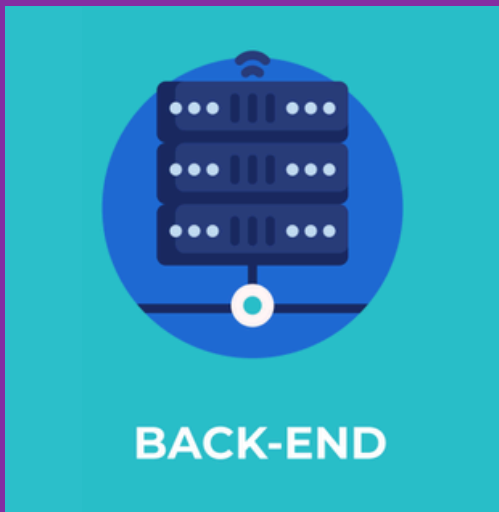


BACK-END IN DETAIL

The back-end operates on the server and is responsible for processing requests, managing data, and running the application's logic - referring to the general rules and procedures used to perform various operations within a web application.

It encompasses the **decision-making processes and algorithms that handle how different pieces of data are manipulated, displayed, or used.**

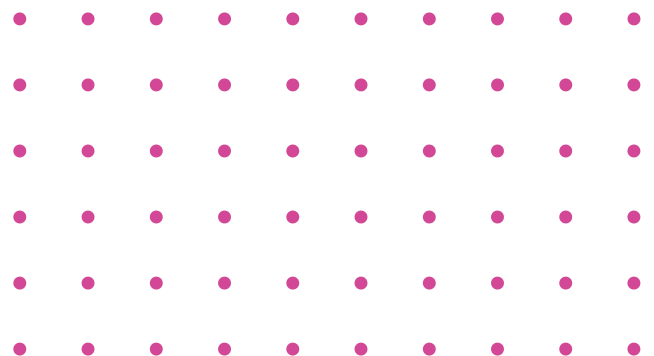
The back-end also handles the business logic - the rules and processes that enforce the policies and procedures of the business. Business logic dictates how data is created, stored, and manipulated according to the specific needs and requirements of the business or application.



For example, **in an e-commerce application, business logic might handle rules related to pricing, inventory management, order processing, and customer discounts.**

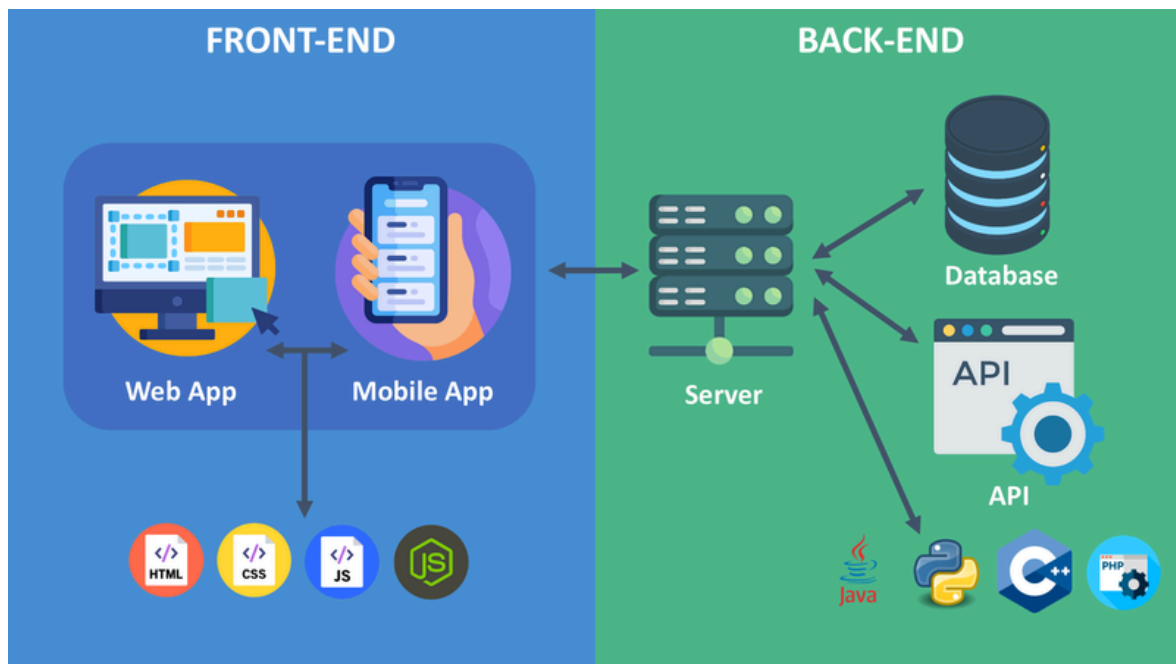
This data processing, and database interactions that are not visible to users but are essential for the application's functionality.

Server-side logic, on the other hand, refers to the **processes and computations that occur on the server in response to client requests.**



PUTTING IT ALL TOGETHER

When you visit a website or take an action, your browser (the client) sends a request to the server. The server processes the request and sends a response back to the browser. This response contains the data or results needed by the front-end to update the user interface.



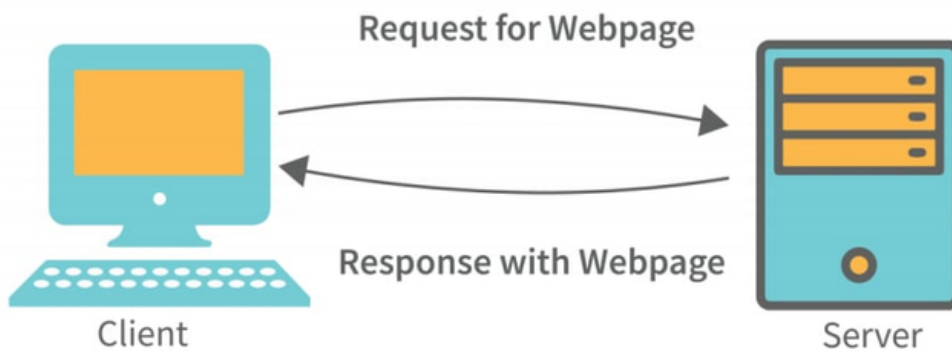
While web browsers are common examples of clients, there are many other types of **devices and applications that also function as clients, such as smartwatches, and smart devices** (such as a Smart TV or a Smart Home Hub).

The client is (which is your web browser in this case), sends requests to the server—a computer that stores the website’s code and data. The server processes these requests and sends back the results, which the client then displays.

The front-end is the part of the system you interact with directly - the **user-facing part of the application on the client side**, including its visual elements. In contrast, the **back-end encompasses the server-side components**. It handles the work behind the scenes, such as processing data and handling requests, and providing information to the front-end.

HTTP, HTTPS & COMMUNICATION

When sending a request from the server, it processes the request and sends a response back to the browser over the internet, utilizing network protocols that ensure secure and reliable communication. These protocols manage the traffic, directing data between the server and browser while verifying that the information is transmitted correctly and efficiently.



The interaction between the front-end and back-end relies on HTTP (HyperText Transfer Protocol) and HTTPS (HTTP Secure) protocols - the fundamental protocols used for communication on the web - for exchanging requests and responses over the web.

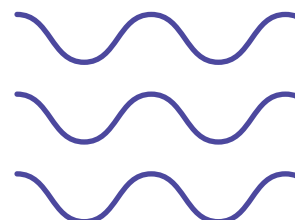
HTTP is the standard protocol for transmitting data between a web browser and a server, used to request and deliver web pages and other resources.

HTTPS, a secure version of HTTP, uses encryption (SSL/TLS) to protect the data being transmitted, ensuring privacy and security, especially for sensitive information like passwords or payment details.

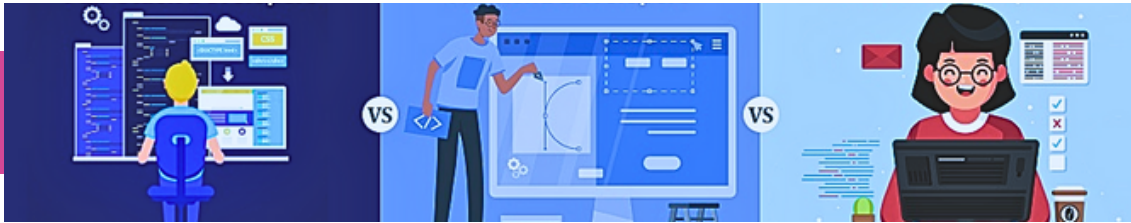
Today, HTTPS (Hypertext Transfer Protocol Secure) is far more common and widely used compared to HTTP - according to recent statistics, over 90% of websites visited on major browsers are using HTTPS.

When you visit a website or perform an action, such as submitting a form (the front-end), your browser (the client) sends an HTTP/HTTPS request to the server (the back-end).

The server processes this request and **sends back an HTTP/HTTPS response containing the requested data or results.** This process involves a continuous exchange of messages between the client and server, where the client makes requests and the server provides responses.



FRONT-END VS BACK-END DEVELOPMENT



Programmers who work with front-end development focus on creating the parts of a website or web application that users interact with and see, such as the layout, visual design, and interactive elements.

This includes creating responsive layouts, and interactive features, and ensuring a smooth, user-friendly experience. They optimize performance, ensure compatibility across browsers, and **use technologies like HTML, CSS, JavaScript, React, and Angular.**

Back-end developers handle servers, and databases, and write application logic that operates behind the scenes. They ensure that the functionality and data management **supporting the front-end** run smoothly. This involves managing databases, writing server-side code, and implementing both application logic and business logic.

On the back-end, programmers handle data processing, security measures, and server maintenance to ensure smooth functionality. **Technologies often used include languages like Python, Java, Ruby, and tools for managing servers and databases.**

As a **full-stack developer, you handle both front-end and back-end development.** This means designing user interfaces, creating interactive features, and managing server-side logic and databases.

You integrate all parts of the web application, and oversee deployment and maintenance. You work with a wide range of technologies from both front-end and back-end.



SERVER-SIDE RENDERING VS FRONT-END RENDERING

Imagine you're setting up a new bulletin board. First, you hang up the board and put up some **basic outlines or placeholders**.

As you receive updates from people—like event **flyers, photos, and notes**—you **attach these to the board in the right spots based on what you need**.

Similarly, the browser sets up the basic structure of a web page and then fills in the text, images, and other content based on the user's requests.

The input from users sends requests to the server, and after processing, the server sends back the requested content to the browser (front-end). This is what makes the page complete and ready for viewing.

The data that the front-end of a web application gets from the back-end includes things like text, images, or other types of content that the page needs. Once the browser receives this information, it **processes that data to fit into the web page**.

The browser updates and displays the new content—the data from the server—such as text, images, and other elements, by filling in the placeholders in the existing layout. This ensures that the web page remains interactive and visually coherent as it dynamically integrates the new data.




This dynamic process of filling in placeholders and updating content highlights how the front-end and back-end of a web application work together to deliver a seamless user experience.

Depending on how the content is delivered to the browser, this interaction can occur in different ways.

However, the method by which content is rendered and displayed on the page can vary, depending on whether the rendering is done on the server or the client.





Server-side rendering means that the web page is created on the server before it is sent to your web browser.

The server prepares the entire web page before sending it to your browser, and you get a fully formed page ready to be displayed immediately.

When using server-side rendering (SSR), the server generates the complete HTML for a web page and sends it to the browser, including embedding all the text, images, and other content directly into the HTML structure. The front-end receives a complete page, with all the content already in place.

Once the page is loaded, JavaScript and other front-end resources may enhance the page's interactivity and visual quality. For example, client-side scripts can still add dynamic features such as animations and handle user input.

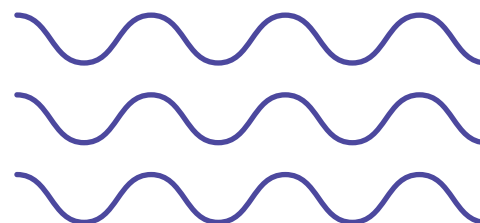
Front-end rendering, also known as client-side rendering (CSR), means that the **web page is built in your browser** after the initial HTML is received from the server. CSR relies on the browser to do most of the work, building the page after receiving a basic HTML shell and JavaScript from the server.

This allows for more dynamic and interactive user experiences but often comes with the trade-off of slower initial load times. **The server sends a minimal HTML layout-** just a basic structure without the full content.

The **browser then builds the web page using the HTML, CSS, and JavaScript it receives from the server** - these files contain the code needed to dynamically build the content of the page.

Once the JavaScript runs, it **dynamically generates the content and inserts it into the HTML structure** to build and populate the page.

As the browser builds and styles the page on-the-fly after executing these scripts, the full content may take longer to appear compared to server-side rendering (SSR). However, front-end rendering enables highly interactive web pages where parts of the page can update dynamically without needing a full page reload.



JSON

Imagine a universal language that makes sure both sides of a conversation can understand each other, no matter what system or software they are using. That's a data interchange format.

A data interchange format is a standardized method for formatting data so that it can be easily transmitted and **understood between different systems**.


One of the most **commonly used ones is JSON** (JavaScript Object Notation), especially within web development.

JSON is a lightweight data-interchange format that is **easy for humans to read and write, and easy for machines to generate and parse**.

Parse means that systems (servers, browsers, databases, and applications) can analyze and interpret JSON to receive data or convert it into a format that other systems can understand and use. JSON is also very flexible, supporting various types of data, including text, numbers, lists, and nested information.

JSON

```
{
  "name": "Ellen Nu",
  "age": 25,
  "isStudent": true,
  "courses": [
    "Math",
    "Programming",
    "Graphic Design"
  ],
  "address": {
    "street": "123 Main St",
    "city": "Anytown",
    "zip": "12345"
  }
}
```



JSON is commonly used to transmit data between a server and a web application as part of client-server communication and plays a crucial role in how dynamic content is loaded and displayed on a web page when using front-end rendering.

When you visit a web page, the browser gets a basic HTML file that might not have all the data - such as the text content asked for.

After the initial HTML is loaded, **the JavaScript running in the browser then asks the server for extra data, usually in JSON format.**

The server sends this data back in JSON format.

The **front-end of the web page takes this JSON data and integrates it into the appropriate elements in the HTML structure.** For instance, if the JSON contains a user's name and bio, the front-end will insert this information into designated spots on the web page, like a user profile section.

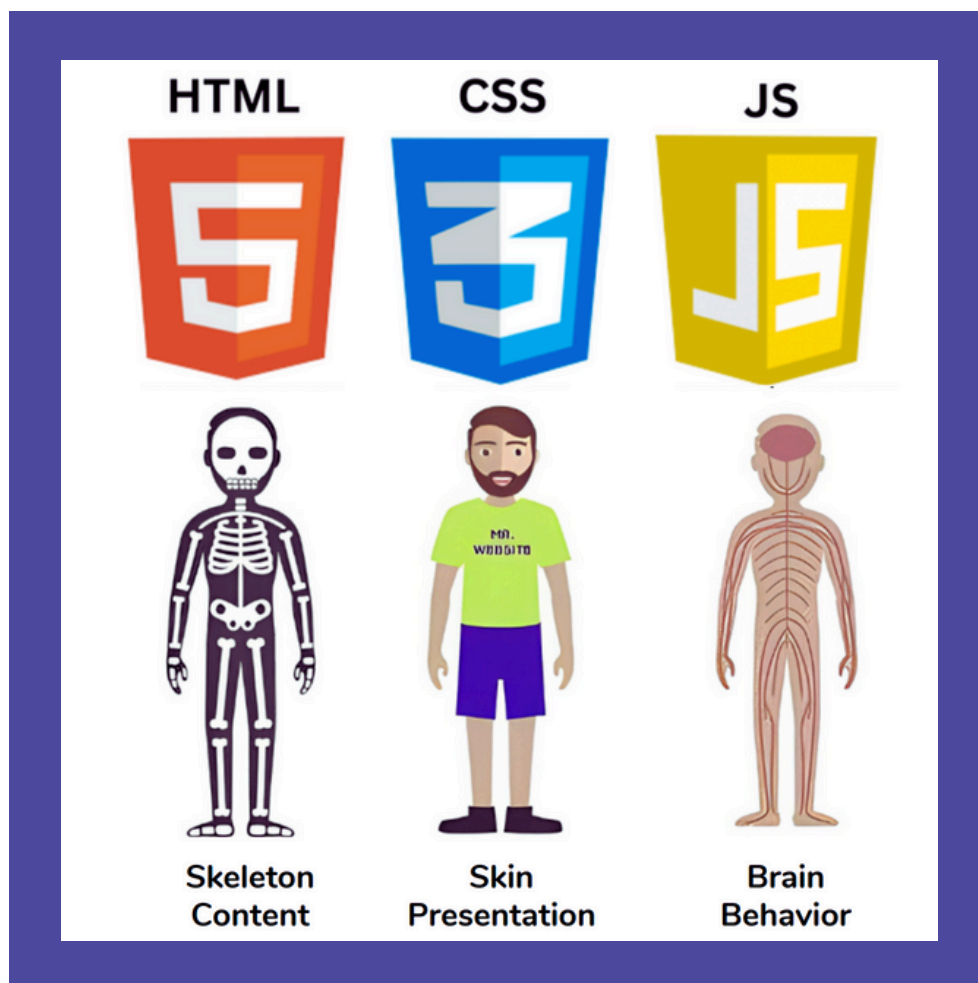
This means that the web page processes JSON data to display content in a way that matches the design of the site, **allowing the content to be rendered and styled according to the page's design but still display user-specific content based on the data received.**

THE BIG 3 OF CLIENT SIDE PROGRAMMING

Three fundamental technologies play a crucial role in this field: **HTML, CSS, and JavaScript.**

Each of these technologies serves a distinct purpose and collectively contributes to the interactive and visually appealing web experiences we encounter today.

HTML structures the content, CSS styles the appearance, and JavaScript adds interactivity, collectively contributing to the dynamic and visually appealing web experiences users encounter.





HTML (HYPERTEXT MARKUP LANGUAGE)

HTML (Hypertext Markup Language) is the backbone of web content - the skeleton. It provides the structure for webpages.

It **defines the hierarchy and layout of content** on a webpage, specifying its organization and structure by **using HTML tags** to arrange the content, making it easy for browsers to understand how to display the information. Using HTML for layout helps in organizing content visually. For example, using <div> tags to create containers for different sections, or using tags to create lists.

HTML uses elements and tags to **create the basic building blocks of a webpage**. Elements in HTML include **headings, paragraphs, links, images, lists, and more**.

HTML also dictates how these elements are nested and ordered. For instance, the <header> tag typically contains navigation at the top of the page, while the <main> tag holds the primary content. Other tags like <footer>, <article>, and <section> define different parts of a webpage.

```
1 <!DOCTYPE html>
2 <html>
3
4   <head>
5     <title>My First Webpage</title>
6   </head>
7
8   <body>
9     <h1>My First Webpage</h1>
10    <p>This is a paragraph.</p>
11  </body>
12
```

HTML, similar to Markdown, has specific rules for formatting that affect how content is displayed. While the written HTML code may appear plain, **it renders differently in a web browser, where the content is styled and organized according to the markup instructions.**

CSS (CASCAADING STYLE SHEETS)



CSS (Cascading Style Sheets) is the design language used to style and layout web pages, enhancing the visual appearance of HTML content. It acts as the "skin" that decorates the "skeleton" provided by HTML.

```
28 .screen-reader-text:focus {
29   background-color: #f1f1f1;
30   border-radius: 3px;
31   box-shadow: 0 0 2px 2px rgba(0, 0, 0, 0.6);
32   clip: auto !important;
33   color: #21759b;
34   display: block;
35   font-size: 14px;
36   font-size: 0.875rem;
37   font-weight: bold;
38   height: auto;
39   left: 5px;
40   line-height: normal;
41   padding: 15px 23px 14px;
```

```
3 p {
4   color: #333;
5   font-size: 13px;
6   line-height: 1.8;
7   font-family: Arial,Helvetica,sans-serif;
8 }
9
10 You: 7 days ago • add more content
11 .navbar, h1, h2, h3, h4, h5 {
12   font-family: 'Roboto Condensed', Helvetica, sans-serif;
13 }
14
15 .block-title {
16   font: bold 16px/16px "Roboto Condensed",sans-serif;
17   font-size: 20px;
18 }
19
20 .titles {
21   font: bold 19px/19px "Roboto Condensed",sans-serif;
```

CSS defines the look and formatting of a webpage, specifying how HTML elements should be styled and arranged.

By using CSS rules, you can control various aspects of a webpage's appearance, such as colors, fonts, spacing, and positioning, making it visually appealing and easy to navigate. For example, CSS can be used to set the background color of a page, adjust the font size of headings, or arrange elements in a grid layout.

CSS utilizes selectors and properties to apply styles to HTML elements. Selectors **target specific elements** or groups of elements, while **properties define the styles.** For instance, a selector like h1 might be used to apply styles to all <h1> headings, specifying properties such as color and font-size.



JAVASCRIPT

JavaScript is a dynamic programming language used to add interactivity and functionality to web pages, bringing life to the static structure provided by HTML and styled by CSS. It acts as the "brains" of a webpage, enabling complex behaviors and interactions.

JavaScript defines how elements on a webpage behave and respond to user actions, **adding interactivity and functionality** to the site.

It allows you to create interactive features such as **form validation, animations, and dynamic content updates**. For example, JavaScript can be used to validate user input on a form, create a carousel that rotates images, or load new content without refreshing the page.

JavaScript utilizes **functions and events to manage behavior and interactions**. **Functions are blocks of code** that perform specific tasks, while **events are triggers for executing these functions**.

An event is triggered when a specific action or condition occurs, such as a user clicking a button—when a **predetermined action happens, the function designed to respond to that event runs**. For example, if a user clicks a button, the “click event” is triggered, and the JavaScript code written to handle that event (e.g., showing a message or revealing additional content on the page) is executed.

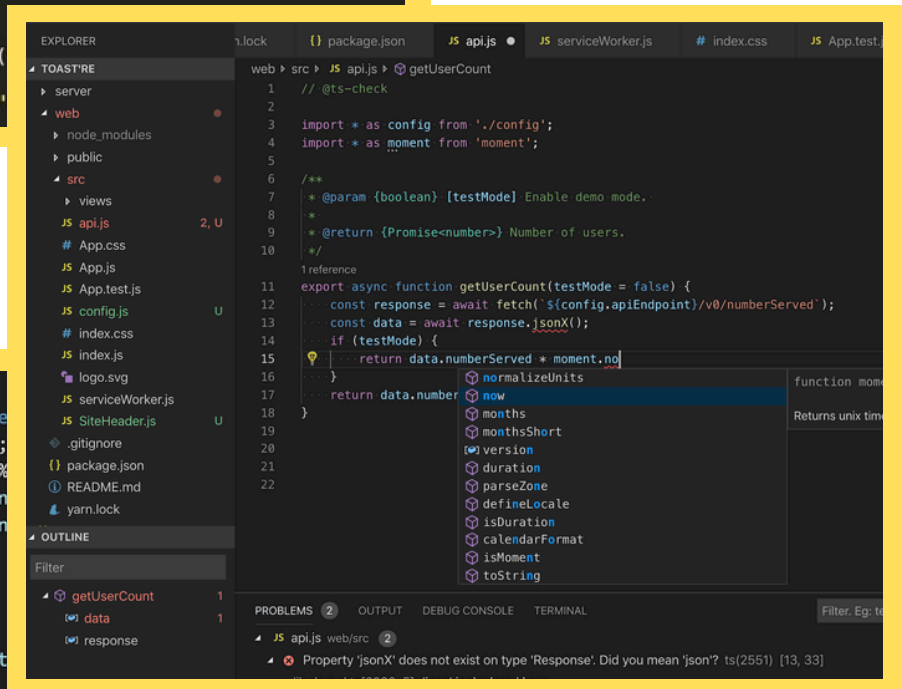
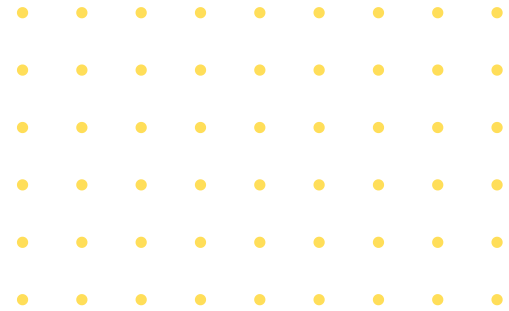
JavaScript can change the content, structure, and styles of elements dynamically. For example, you can use JavaScript to update the text of a paragraph or change the color of a button based on user actions.



```

1 var button = document.getElementById("enter");
2 var input = document.getElementById("userinput");
3 var ul = document.querySelector("ul");
4
5 function inputLength() {
6     return input.value.length;
7 }
8
9 function createListElement() {
10    var li = document.createElement("li");
11    li.appendChild(document.createTextNode(input.value));
12    ul.appendChild(li);
13    input.value = "";
14 }
15
16 function addListAfterClick() {
17     if (inputLength() > 0) {
18         createListElement();
19     }
20 }
21
22 function addListAfterKeypress(event) {
23     if (inputLength() > 0 && event.keyCode === 13) {
24         createListElement();
25     }
26 }
27
28 button.addEventListener("click", addListAfterClick);
29
30 input.addEventListener("keypress", addListAfterKeypress);

```



```

14 function isPrime(num) {
15     if (num <= 1) return false;
16     if (num <= 3) return true;
17     if (num % 2 === 0 || num % 3 === 0) return false;
18     for (let i = 5; i * i <= num; i += 6) {
19         if (num % i === 0 || num % (i + 2) === 0) return false;
20     }
21     return true;
22 }
23
24 function sumOfPrimesInRange(start, end) {
25     let sum = 0;
26     for (let i = start; i <= end; i++) {
27         if (isPrime(i)) sum += i;
28     }
29     return sum;
30 }
31
32 function generateFibonacciSequence(count) {
33     const sequence = [];
34     for (let i = 0; i < count; i++) {
35         sequence.push(fibonacci(i));
36     }
37     return sequence;
38 }
39
40 const num = 10;
41 const primeSum = sumOfPrimesInRange(1, num);
42 const fibSequence = generateFibonacciSequence(num);
43 console.log("Sum of primes:", primeSum);
44 console.log("Fibonacci sequence:", fibSequence);
45
46 // https://es.wikipedia.org/wiki/JavaScript

```

