HTML and CSS

Please make sure that you cover the following topics that are fundamental to the understanding of HTML and CSS3 HTML (Hypertext Markup Language):

- **Understanding HTML Structure:** HTML is a markup language used to structure content on the web. The basic structure of an HTML document includes tags like <!DOCTYPE html>, <html>, <head>, <body>, etc.
- **Tags and Elements:** HTML is built on tags and elements. Elements include an opening tag, content, and a closing tag. Example: <p>This is a paragraph.</p>. Some tags are self-closing and don't require a closing tag, like the <img /> or <br /> tag.
- **Common HTML Elements:** Familiarize yourself with frequently used elements such as <h1> to <h6> (headings), <p> (paragraph), <a> (anchor for links), <img> (image), <div> and <span> (general containers), <ul> and <li> (unordered list and list item), <table> (for tables) and form-related elements like <form>, <input>, <label>, <button>, etc.
- **Attributes:** HTML elements can have attributes, which are used to provide additional information about the element. Common attributes include class, id, src, href, alt, style, etc.

CSS3 (Cascading Style Sheets):

- **Purpose of CSS:** While HTML structures the content, CSS is used to style and layout web pages. For example, colors, fonts, and layout properties are defined in CSS.
- **CSS Selectors:** Selectors are used to select and manipulate HTML elements. The most common selectors are element type, .class, #id, and attribute selectors.
- **CSS Properties:** CSS properties are what you use to style elements. Properties include color, background-color, font-family, font-size, border, margin, padding, display, width, height, etc.
- **Box Model:** Understanding the CSS box model is crucial. It's the basis for layout on the Web, and consists of: the content box, padding box, border box, and margin box.
- **Positioning and Layouts:** Understanding different CSS positioning properties like static, relative, absolute, fixed, and sticky are important. In

addition, CSS offers several methods for layout including normal flow, flexbox, and grid.

- **Media Queries:** CSS3 introduced media queries, which allow you to apply different styles depending on the characteristics of the device/display (like width and height). This is a core concept for responsive web design.
- **CSS Transitions and Animations:** CSS3 also introduced the ability to create transitions and animations with pure CSS, no JavaScript required.

HTML and DOM: When a web page is loaded, the browser creates a Document Object Model of the page, which is an object-oriented representation of the HTML document that JavaScript can interact with. Every part of an HTML document (HTML tags, attributes, text, etc.) gets represented as an object in the DOM, and these objects are arranged in a hierarchical tree-like structure. For instance, the <body> tag of an HTML document would be a node in the DOM, and any elements within the <body> tag would be its child nodes. This allows developers to manipulate HTML elements (tags) and their attributes/properties, change the CSS, or even change the entire structure of the HTML document dynamically using JavaScript. So, the HTML is the static markup of your webpage, and the DOM is a dynamic, programmable version of your HTML page in the browser's memory.

1. **DOM is an Object-Oriented Representation of the Webpage:**

When you load a webpage in a browser, it reads the HTML (which is the blueprint of the webpage) and creates an object-oriented representation of that page in its memory. This is the Document Object Model (DOM). Think of it as a "map" or "family tree" that the browser creates based on the HTML. Each part of the HTML (tags, attributes, text) is represented as an "object" in this model.

2. **HTML to DOM:**

Let's break down how the HTML is converted into the DOM: First, the browser fetches the HTML document (the webpage's blueprint). The browser reads the HTML from top to bottom. For each piece of HTML (each tag, like <p>, <div>, <img>, etc.), it creates a corresponding "object" in its memory. Each of these objects holds information about a part of the webpage. For example, an object for an <img> tag would store the image's source URL, its alt text, its width and height, and so on. The objects are arranged in a tree structure that mirrors the structure of the HTML. This is the DOM.

3. **JavaScript Manipulates the DOM:**

Here's how JavaScript can be used to manipulate (change) the DOM: JavaScript can select objects in the DOM using methods like **document.getElementById, document.getElementsByClassName, document.querySelector, and others.** These methods allow you to select elements by their tag name, class, id, or other attributes. Once an object is selected, you can manipulate it. For example, you could change an image's source URL to display a different image, or change the text inside a paragraph. JavaScript can also add new objects to the DOM (like adding a new paragraph at the end of a section) or remove objects (like removing an image). Any changes JavaScript makes to the DOM are saved in the browser's memory.

4. **Browser Re-renders the DOM:**

After the DOM is updated, here's how the browser re-renders (redraws) it: The browser notices that changes have been made to the DOM in its memory. It looks at the changed parts of the DOM and compares them to what is currently displayed on the screen. Then it updates only those parts of the screen that correspond to the changed parts of the DOM. This is more efficient than redrawing the whole screen. The updated screen now reflects the changes JavaScript made to the DOM.

Javascript is a very simple scripting language which simply updates the objects in the DOM. Then browser's renderer gets a signal to update the DOM. The browsers are so advanced today that they don't update the entire DOM, they simply update the portion of the screen that is related to that specific DOM object. This technique (in architecture) is called KVO - Key Value Observation - Generally done in all event driven software.

Imagine that DOM is a collection of objects. Each object is a key - that key has a value. The browser keeps on observing the values of these objects (in the DOM) and as soon as an object's value changes, the browser instructs the renderer to redraw that portion of the page.