

**Describe how HTML fits into the broader ecosystem of a website. Contrast the fundamental role of HTML with the primary roles of CSS and JavaScript. (5 points)**

HTML, CSS, and JavaScript are the three foundational pillars of nearly every website, working together within the broader web ecosystem to deliver a complete user experience. HTML (HyperText Markup Language) is the structural backbone of a webpage; its fundamental role is to define the content and organization of the page. It basically dictates the hierarchy of elements. It defines what is a heading, what is a paragraph, where an image should go, and how a list is structured. In contrast, CSS (Cascading Style Sheets) focuses on the presentation and aesthetics of that structure. Its primary role is to manage the visual styling, including the colors, fonts, layout, spacing, and responsive design, transforming raw HTML content into a visually appealing interface. Finally, JavaScript brings interactivity and dynamic behavior to the page. Its primary role is to execute logic, handle user input (like button clicks), manipulate the HTML and CSS (known as the DOM), and communicate with servers, allowing for complex functionalities such as form validation, carousels, or real-time updates. Overall, HTML provides the content, CSS makes it look good, and JavaScript makes it work.

**Explain the difference between HTML structure and HTML semantics. Why is writing semantic HTML considered a best practice? Provide one example of a semantic HTML element and one example of a non-semantic element. (10 points)**

The difference between HTML structure and HTML semantics lies in their focus. HTML structure refers to the arrangement and nesting of elements simply to define the *physical layout* or hierarchy of content on the page, often achieved using generic container elements like `<div>` and `<span>` for grouping and styling purposes. In contrast, HTML semantics involves using elements that convey the meaning or role of the content they contain, rather than just how the content appears. Writing semantic HTML is considered a best practice because it significantly improves accessibility and Search Engine Optimization (SEO). For accessibility, screen readers and other assistive technologies rely on semantic tags (like `<nav>` or `<main>`) to understand the page's structure and navigate content for users with disabilities. For SEO, search engine crawlers better understand the importance and context of the content, leading to improved rankings. An example of a semantic HTML element is `<article>`, which clearly defines an independent, self-contained piece of content (like a blog post). An example of a non-semantic element is `<div>`, which has no intrinsic meaning regarding the content it holds—it's merely a division or container.

**What is the "three-tier model" (also known as three-tier architecture) in web systems? Briefly describe the function and responsibility of each of the three tiers. (15 points)**

The "three-tier model" is a well-established software architecture pattern for designing and implementing distributed web systems, separating the application into three logical and often

physical computing tiers. This separation allows for better maintainability, scalability, and security.

First is the Presentation Tier (Client Tier). This is the top-most tier that users directly interact with, often running in the user's web browser. Its main responsibility is to display the user interface, translate tasks and results into a format the user can understand (HTML, CSS, JavaScript), and collect input from the user. In other words, it is the visual layer of the application, so for example, the HTML page displayed on a user's laptop.

Second is the application tier. This middle tier serves as the controller and processing engine. It contains the business logic, the rules, and procedures that govern the application's core functions. It accepts requests from the Presentation Tier, processes them according to the business rules (e.g., calculating a price, validating a login), and often communicates with the Data Tier. An example of this would be a server-side application built with Node.js, Python, or Java that processes a user's form submission and determines which database records to access.

Third is the Data Tier. This is the bottom-most tier, which stores, retrieves, and manages the application's data. Its responsibility is to provide data persistence and integrity, ensuring data is stored correctly and can be accessed efficiently by the Application Tier. It's often implemented using a Relational Database Management System (MySQL) or a NoSQL database (DynamoDB).

### **Explain what is meant by a Uniform Interface in a REST API. (5 points)**

The Uniform Interface is one of the six key architectural constraints that define REST (Representational State Transfer) and is arguably the most crucial principle for achieving system simplicity and scalability. It dictates that the interface between clients and servers must be standardized and constrained, promoting the overall decoupling of the architecture. This means clients should interact with all resources in the same manner, regardless of the underlying server implementation or the specific resource being accessed. The Uniform Interface constraint is achieved through the following sub-principles:

**Identification of Resources:** Every resource is uniquely identified via a URI (Uniform Resource Identifier).

**Manipulation of Resources Through Representations:** Clients manipulate resources by exchanging representations (like JSON or XML) of the resource's current state. For example, a client updates a user's address by sending a JSON representation of the new data.

**Self-Descriptive Messages:** Every message exchanged must contain enough information for the recipient (client or server) to understand how to process the message without relying on previous session state. This typically involves using standard HTTP methods (like **GET**, **POST**,

`PUT`, `DELETE`) to convey the *intent* of the request, and media types (like `application/json`) to describe the *format* of the data.

Hypermedia as the Engine of Application State (HATEOAS): The server returns hypermedia (links) within its responses, which guide the client on what actions are possible next, rather than the client needing to hardcode the available interactions.

In essence, the Uniform Interface makes a REST API predictable and discoverable; a client doesn't need to know the specific details of the server to interact with it, only the standard rules of HTTP, leading to better visibility, simplicity, and decoupling between the components.

**Explain how your browser chooses which CSS rule to apply to a tag in the case where there are multiple rules that could apply. (15 points)**

When a browser encounters a situation where multiple CSS rules could potentially apply to a single HTML element (tag), it uses a set of precise rules to determine which declaration takes precedence; this process is called cascading and specificity. The specificity of a CSS selector is a calculated weight assigned to it, which determines how strongly a declaration will override others. The browser calculates this weight based on the number and type of selectors used in the rule.

Inline Styles: Styles applied directly in the HTML tag (`<p style="...">`) have the highest specificity and always win (unless overridden by `!important`).

IDs: Selectors using an ID attribute (e.g., `#my-id`) have the next highest weight.

Classes, Attributes, and Pseudo-classes: Selectors using classes (e.g., `.my-class`), attribute selectors (e.g., `[type="text"]`), or pseudo-classes (e.g., `:hover`) are less specific than IDs.

Elements and Pseudo-elements: Selectors targeting an HTML element (e.g., `p`, `h1`) or pseudo-elements (e.g., `::before`) have the lowest specificity.

The browser counts the number of each type of selector, and the one with the highest count (starting from IDs, then classes, then elements) wins. For example, a rule defined as `\#main-nav li a` (1 ID, 1 element, 1 element) is more specific than a rule defined as `.nav-link` (1 class). If two rules have the same specificity score, the last rule defined in the stylesheet (the one that appears lower in the source code) is the one that is applied, following the cascade order. This systematic process ensures a predictable and deterministic way for the browser to render the final visual style.