**Module:1**

1. **Define HTML. What is the purpose of HTML in web development?**

* **HTML** (HyperText Markup Language) is the standard language used to create and structure web pages.
* Its purpose is to define the content and layout of a webpage using tags, like headings, paragraphs, images, and links.

1. **Explain the basic structure of an HTML document. Identify the mandatory tagsand their purposes.**

* <!DOCTYPE html> – Declares the HTML version.
* <html> – Root element of the document.
* <head> – Contains meta info, title, and links (like CSS).
* <title> – Sets the page title (shown in browser tab).
* <body> – Holds the visible content like text, images, links.

1. **What is the difference between block-level elements and inline elements in HTML? Provide examples of each.**

* **Block-level elements** start on a new line and take up full width (e.g., <div>, <p>, <h1>).  
  **Inline elements** stay in line with text and take up only as much space as needed (e.g., <span>, <a>, <strong>).

1. **Discuss the role of semantic HTML. Why is it important for accessibility andSEO? Provide examples of semantic elements.**

* Semantic HTML uses meaningful tags (like <header>, <nav>, <article>, <footer>) to describe content structure.
* Accessibility – Screen readers understand the layout better.
* SEO – Search engines index content more effectively.
* Examples: <main>, <section>, <aside>, <figure>.

**Module:2**

1. **What are HTML forms used for? Describe the purpose of the input, textarea, select, and button elements.**

* HTML forms collect user input (like login info or feedback).
* <input> – Takes single-line input (text, email, etc.).
* <textarea> – Allows multi-line text input.
* <select> – Creates a dropdown menu.
* <button> – Submits the form or triggers actions.

1. **Explain the difference between the GET and POST methods in form submission. When should each be used?**

* **GET** adds form data to the URL—best for simple, non-sensitive data (like searches).  
  **POST** sends data securely in the body—used for sensitive or large data (like login forms).

1. **What is the purpose of the label element in a form, and how does it improve accessibility?**

* The **<label>** element links text to a form input, making forms easier to understand and use.
* It improves **accessibility** by helping screen readers identify inputs, especially for users with disabilities.

**Module:3**

1. **Explain the structure of an HTML table and the purpose of each of the following elements:<table>,<tr>,<td>,<th> and <thead>?**

* <table>: The main container for the table.
* <tr> (Table Row): Defines a row in the table.
* <th> (Table Header): Defines a header cell (bold and centered by default).
* <td> (Table Data): Defines a standard data cell.
* <thead>: Groups the header content (usually contains <tr> with <th> cells).

1. **What is the difference between colspan and rowspan in tables? Provide examples.**

* Colspan:Merges columns (cells across).
* Rowspan: Merges rows (cells down).

1. **Why should tables be used sparingly for layout purposes? What is a better alternative?**

* Tables should be used sparingly for layout because:
* They are not responsive.
* They make code harder to maintain.
* They reduce accessibility for screen readers.
* Better alternative: Use CSS with <div> for layout (e.g., Flexbox or Grid).

**Module:4**

1. **Difference b/w HTML & HTML5?**

* **HTML**: Older version, limited multimedia support, uses external plugins (like Flash).
* **HTML5**: Newer version, supports audio/video, new semantic tags (<header>, <section>), offline storage, and better mobile support.

1. **What are the additional tags used in HTML5?**

* Additional HTML5 Tags (Short):
* Structural: <header>, <footer>, <section>, <article>, <nav>, <aside>
* Media: <audio>, <video>, <source>, <track>
* Form: <datalist>, <output>, <progress>, <meter>
* Interactive: <details>, <summary>, <dialog>
* These improve structure, media handling, and interactivity.

**Module:5**

1. **What is a CSS selector? Provide examples of element, class, and ID selectors.**

* A CSS selector targets HTML elements to apply styles.
* Element selector: p { color: blue; }
* Class selector: .menu { font-size: 18px; }
* ID selector: #header { background: gray; }

1. **Explain the concept of CSS specificity. How do conflicts between multiple stylesget resolved?**

* CSS specificity decides which style wins when multiple rules target the same element.
* Inline styles > IDs > Classes/attributes > Elements
* Conflicts are resolved by:
* Higher specificity wins.
* If specificity ties, the last rule written wins.

1. **What is the difference between internal, external, and inline CSS? Discuss the advantages and disadvantages of each approach.**

* **Internal CSS:**
* In <style> inside <head>.
* Advantage: Easy for single pages.
* Disadvantage: Not reusable across pages.
* **External CSS:**
* In a separate .css file linked via <link>.
* Advantage: Reusable, keeps HTML clean.
* Disadvantage: Extra HTTP request.
* **Inline CSS:**
* In the style attribute of elements.
* Advantage: Quick for small changes.
* Disadvantage: Hard to maintain, not reusable.

**Module:6**

1. **Explain the CSS box model and its components (content, padding, border, margin). How does each affect the size of an element?**

* The CSS box model defines how elements are sized:
* Content: The actual text or image.
* Padding: Space inside the border, around content.
* Border: The edge surrounding padding and content.
* Margin: Space outside the border, separating elements.
* Effect on size:  
  Element size = Content + Padding + Border + Margin.

1. **What is the difference between border-box and content-box box-sizing inCSS? Which is the default?**

* Content-box (default):
* Width/height = content only.
* Padding and border add extra size.
* Border-box:
* Width/height = content + padding + border combined.
* Easier to manage layout.

**Module:7**

1. **What is CSS Flexbox, and how is it useful for layout design? Explain the terms flex-container and flex-item.**

* CSS Flexbox (Flexible Box Layout) is a layout model in CSS that allows you to design a responsive and flexible layout structure without using float or positioning.
* **flex-container**
* The parent element where display: flex is applied. It defines how the child elements (flex-items) are arranged.
* **flex-item**
* The direct child elements of the flex-container. These items can grow, shrink, and align easily using Flexbox properties.

1. **Describe the properties justify-content, align-items, and flexdirection used in Flexbox.**

* **flex-direction**: Sets the direction of items.  
  → row, column, row-reverse, column-reverse
* **justify-content**: Aligns items **horizontally** (main axis).  
  → flex-start, center, flex-end, space-between, etc.
* **align-items**: Aligns items **vertically** (cross axis).  
  → flex-start, center, flex-end, stretch, baseline

**Module:8**

1. **Explain CSS Grid and how it differs from Flexbox. When would you use Grid over Flexbox?**

* **Difference from Flexbox:**
* **Grid**: Works in **both rows and columns** (2D).
* **Flexbox**: Works in **one direction at a time** (1D — row *or* column).
* **When to use Grid:**
* Use **Grid** for **complex layouts** with rows and columns (e.g., full-page layouts).
* Use **Flexbox** for **simple, one-directional layouts** (e.g., nav bars, buttons in a row).

1. **Describe the grid-template-columns, grid-template-rows, and grid-gap properties. Provide examples of how to use them.**

* **grid-template-columns**
* Defines the number and width of columns.
* **grid-template-rows**
* Defines the number and height of rows.
* **grid-gap *(or gap)***
* Sets space between rows and columns.
* **Example:**
* .container {
* display: grid;
* grid-template-columns: 1fr 1fr;
* grid-template-rows: auto auto;
* gap: 10px;
* }

**Module:9**

1. **What are media queries in CSS, and why are they important for responsive design?**

* They make websites responsive, meaning they adapt to different screen sizes (mobile, tablet, desktop).
* **Example:**
* @media (max-width: 768px) {
* body {
* font-size: 16px;
* }
* }

1. **Write a basic media query that adjusts the font size of a webpage for screens smaller than 600px.**

* @media (max-width: 600px) {
* body {
* font-size: 14px;
* }
* }

**Module:10**

1. **Explain the difference between web-safe fonts and custom web fonts. Whymight you use a web-safe font over a custom font?**

* **Web-safe fonts** are pre-installed on most devices (e.g., Arial, Times New Roman), so they display the same across all browsers without needing downloads.
* **Custom web fonts** (e.g., from Google Fonts) are downloaded by the browser and offer more design flexibility.
* **Why use web-safe fonts?**
* **Faster loading**
* **Better compatibility**
* Useful when **performance** or **fallback** is important.

1. **What is the font-family property in CSS? How do you apply a custom GoogleFont to a webpage?**

* **Import in <head>:**
* <link href="https://fonts.googleapis.com/css2?family=Roboto&display=swap" rel="stylesheet">
* body {
* font-family: 'Roboto', sans-serif;
* }

**Module:10**

1. **What is JavaScript? Explain the role of JavaScript in web development.**

* JavaScript is a language that makes websites interactive and dynamic, working with HTML (structure) and CSS (style).

1. **How is JavaScript different from other programming languages like Python orJava?**

* JavaScript mainly runs inside browsers for web interactivity, while Python and Java are general-purpose languages used for apps, AI, backend, etc.

1. **Discuss the use of <script> tag in HTML. How can you link an external JavaScript file to an HTML document?**

* The **<script> tag** is used to add JavaScript in HTML.
* You can link an external file with:
* <script src="file.js"></script>

1. **What are variables in JavaScript? How do you declare a variable using var, let,and const?**

* Variables in JavaScript are used to store data values.
* **var** → old way, function-scoped
* **let** → block-scoped, changeable
* **const** → block-scoped, cannot be reassigned ✅

1. **Explain the different data types in JavaScript. Provide examples for each.**

* **String** → text → "Hello"
* **Number** → numbers → 123, 3.14
* **Boolean** → true/false → true
* **Undefined** → no value → let x;
* **Null** → empty → let y = null;
* **Object** → key-value pairs → {name:"Raj", age:20}
* **Array** → list → [1, 2, 3]

1. **What is the difference between undefined and null in JavaScript?**

* Undefined → variable declared but no value assigned.  
  Null → assigned value meaning "empty" or "nothing".

1. **What are the different types of operators in JavaScript?**

* **Explain with examples.**
* **Arithmetic operators**
* **Assignment operators**
* **Comparison operators**
* **Logical operators**
* **Arithmetic → + , - , \* , /**
* **Assignment → = , += , -=**
* **Comparison → == , === , > , <**
* **Logical → && , || , !**

1. **What is the difference between == and === in JavaScript?**

* == → compares values only (performs type conversion).  
  === → compares values and data types (strict).

1. **What is control flow in JavaScript? Explain how if-else statements work with example.**

* Control flow = order of code execution.  
  if-else → runs one block if condition true, else another.

1. **Describe how switch statements work in JavaScript. When should you use a switch statement instead of if-else?**

* Switch → used to test one value with many cases, better than many if-else.

1. **Explain the different types of loops in JavaScript (for, while, do-while). Provide abasic example of each.**

* **for** → repeats with counter
* **while** → runs while condition true
* **do-while** → runs at least once

1. **What is the difference between a while loop and a do-while loop?**

* while → checks condition first, may run 0 times.  
  do-while → runs once first, then checks condition.

1. **What are functions in JavaScript? Explain the syntax for declaring and calling a function.**

* Functions = reusable code blocks.
* function name(){...}
* name();

1. **What is the difference between a function declaration and a function expression?**

* Declaration → named, hoisted.  
  Expression → in variable, not hoisted.

1. **Discuss the concept of parameters and return values in functions.**

* Parameters = inputs, Return = output of function.

1. **What is an array in JavaScript? How do you declare and initialize an array?**

* Array = collection of values in one variable.

1. **Explain the methods push(), pop(), shift(), and unshift() used in arrays.**

* **push() → add at end**
* **pop() → remove from end**
* **shift() → remove from start**
* **unshift() → add at start**

1. **What is an object in JavaScript? How are objects different from arrays?**

* Object = collection of key-value pairs.  
  Array = ordered list (index-based).

1. **Explain how to access and update object properties using dot notation and bracket notation.**

* Access/Update → dot: obj.key, bracket: obj["key"]

1. **What are JavaScript events? Explain the role of event listeners.**

* Events = actions in browser (click, keypress, load).  
  Event listeners = functions that run when event happens.

1. **What is the DOM (Document Object Model) in JavaScript? How does JavaScript interact with the DOM?**

* DOM = tree-like structure of a webpage (HTML as objects).  
  JavaScript can access, change, add, or delete elements in the DOM.

1. **What is the DOM (Document Object Model) in JavaScript? How does JavaScript interact with the DOM?**

* DOM = representation of webpage as a tree.  
  JavaScript interacts by manipulating elements, styles, and content.

1. **Explain the methods getElementById(), getElementsByClassName(),and querySelector() used to select elements from the DOM.**

* **getElementById() → select by ID**
* **getElementsByClassName() → select all by class (HTMLCollection)**
* **querySelector() → select first match (CSS selector)**

1. **Explain the setTimeout() and setInterval() functions in JavaScript. Howare they used for timing events?**

* setTimeout() → runs code once after delay.  
  setInterval() → runs code repeatedly at intervals.

1. **Provide an example of how to use setTimeout() to delay an action by 2 seconds.**

* setTimeout(()=>{
* console.log("Action after 2s");
* }, 2000);

1. **What is error handling in JavaScript? Explain the try, catch, and finally blocks with an example.**

* **Error handling = managing runtime errors safely.**
* **try** → code to test
* **catch** → handles error
* **finally** → runs always.

1. **Why is error handling important in JavaScript applications?**

* Error handling prevents apps from crashing, ensures smooth user experience, and helps debug issues.