1.What does HTML stand for and what is its purpose?

HTML stands for **HyperText Markup Language**. Its primary purpose is to structure content on the web, allowing for the creation of web pages and applications.

2.Describe the basic structure of an HTML document.

An HTML Document is mainly divided into two parts:

- **HEAD**: This contains the information about the HTML document including the Title of the page, version of HTML, Meta Data, etc.
- **BODY**: This contains everything you want to display on the Web Page.

3.What do DOCTYPE and html lang attributes do?

- **`<!DOCTYPE html>`**: This declaration defines the document type and version of HTML, helping the browser render the page correctly.
- **`<html lang="en">`**: The `lang` attribute specifies the language of the document, improving accessibility and SEO.

4.What is the difference between head and body tags?

- **`<head>`**: Contains metadata, links to stylesheets, scripts, and the title of the document. It doesn't display content directly on the page.
- **`<body>`**: Contains the actual content that displays on the web page, such as text, images, and other elements.

5.Can you explain the purpose of meta tags in HTML?

Meta tags provide metadata about the HTML document, such as character set, author, description, and keywords. They are essential for SEO and for controlling the behavior of browsers and search engines.

6.How do you link a CSS file to an HTML document?

You can link a CSS file using the `<link>` tag within the `<head>` section:

```
<link rel="stylesheet" href="styles.css">
```

7.How do you link a JavaScript file to an HTML document?

You can link a JavaScript file using the `<script>` tag, usually placed before the closing `</body>` tag:

```
<script src="script.js"></script>
```

8.How do you add a comment in HTML and why would you use them?

You can add comments using the following syntax:

Comments are used for documentation, notes, and explaining sections of code without affecting the rendered output.

9.How do you serve your page in multiple languages?

You can serve multiple languages by using the `lang` attribute in the `<html>` tag and providing localized content. Additionally, you might use the `<meta>` tag for `charset` and implement language-specific URLs.

10.What are data-* attributes and when should they be used?

Data attributes are custom attributes that start with `data-`, used to store extra information on standard HTML elements without using additional properties. They are useful for storing data that can be accessed via JavaScript.

```
<div data-user-id="12345"></div>
```

11.What is the difference between b and strong tags?

- **`<b>`**: Makes text bold but does not convey any additional importance.
- **`<strong>`**: Also makes text bold but indicates that the text is of strong importance or seriousness.

12.When would you use em over i, and vice versa?

- **`<em>`**: Emphasizes text, usually implying stress or importance, and is semantically meaningful.
- **`<i>`**: Renders text in italics but does not imply any special importance. Use `<em>` for emphasis and `<i>` for stylistic purposes.

13.What is the purpose of small, s, and mark tags?

- **`<small>`**: Renders text in a smaller size, often for fine print or disclaimers.
- **`<s>`**: Strikes through text to indicate that it is no longer relevant or accurate.
- **`<mark>`**: Highlights text, typically for emphasis or to indicate something important.

14.What are semantic HTML tags and why are they important?

Semantic HTML tags convey meaning about the content they enclose, like `<header>`, `<footer>`, `<article>`, and `<section>`. They improve accessibility, SEO, and the overall structure of the document.

15.How do you create a paragraph or a line break in HTML?

- To create a paragraph:

```
<p>This is a paragraph.</p>
```

- To create a line break:

16.How do you create a hyperlink in HTML?

You can create a hyperlink using the `<a>` tag:

```
<a href="https://www.example.com">Visit Example</a>
```

17.What is the difference between relative and absolute URLs?

- **Relative URLs**: Link to a resource relative to the current document, e.g., `about.html`.
- **Absolute URLs**: Provide the full path, including the protocol, domain, and path, e.g., `https://www.example.com/about.html`.

18.How can you open a link in a new tab?

You can open a link in a new tab by adding the `target` attribute:

```
<a href="https://www.example.com" target="_blank">Open in a new tab</a>
```

19.How do you create an anchor to jump to a specific part of the page?

You can create an anchor link using the `id` attribute:

```
<a href="#section1">Go to Section 1</a>

<h2 id="section1">Section 1</h2>
```

20.How do you link to a downloadable file in HTML?

You can link to a downloadable file using the `<a>` tag with the `download` attribute:

```
<a href="path/to/file.pdf" download>Download PDF</a>
```

21.How do you embed images in an HTML page?

You can embed images using the `<img>` tag:

```
<img src="path/to/image.jpg" alt="Description of image">
```

22.What is the importance of the alt attribute for images?

The **`alt` attribute** provides alternative text for images, which is important for accessibility. It helps screen readers describe the image to visually impaired users and is displayed if the image fails to load.

23.What image formats are supported by web browsers?

Commonly supported image formats include:

- **JPEG** (`.jpg`, `.jpeg`)
- **PNG** (`.png`)
- **GIF** (`.gif`)
- **SVG** (`.svg`)
- **WebP** (`.webp`)

24.How do you create image maps in HTML?

You can create image maps using the `<map>` element along with the `<area>` tags:

```
<img src="image.jpg" usemap="#image-map">

<map name="image-map">
    <area shape="rect" coords="34,44,270,350" href="link1.html" alt="Link
1">
    <area shape="circle" coords="337,300,44" href="link2.html" alt="Link
2">
</map>
```

25.What is the difference between svg and canvas elements?

- **`<svg>`**: Used for scalable vector graphics; defines graphics in XML format, which means it can be manipulated with CSS and JavaScript.
- **`<canvas>`**: Used for drawing graphics on the fly via JavaScript; it's pixel-based and does not retain any drawing state after rendering.

26.What are the different types of lists available in HTML?

HTML provides three main types of lists:

1. **Ordered List** (`<ol>`)
2. **Unordered List** (`<ul>`)
3. **Description List** (`<dl>`)

27.How do you create ordered, unordered, and description lists in HTML?

- **Ordered List**:

```
<ol>
    <li>First item</li>
    <li>Second item</li>
</ol>
```

- **Unordered List**:

```
<ul>
    <li>First item</li>
    <li>Second item</li>
</ul>
```

- **Description List**:

```
<dl>
    <dt>Term 1</dt>
    <dd>Description for term 1</dd>
</dl>
```

28.Can lists be nested in HTML? If so, how?

Yes, lists can be nested by placing one list inside another:

```
<ul>
    <li>First item
        <ul>
            <li>Sub-item</li>
        </ul>
    </li>
</ul>
```

29.What attributes can you use with lists to modify their appearance or behavior?

Common attributes include:

- **`type`**: Specifies the type of marker (for `<ol>` and `<ul>`, e.g., `type="A"` for uppercase letters).
- **`start`**: Defines the starting number for ordered lists.
- **`reversed`**: Indicates that the list should be displayed in reverse order (for `<ol>`).

30.What are HTML forms and how do you create one?

HTML forms are used to collect user input. You can create one using the `<form>` tag:

```
<form action="submit.php" method="post">
    <input type="text" name="username">
    <input type="submit" value="Submit">
</form>
```

31.Describe the different form input types in HTML5.

Common HTML5 form input types include:

- **`text`**: Single-line text input
- **`password`**: Password input (masked)
- **`email`**: Email input (validates format)
- **`number`**: Numeric input
- **`date`**: Date input (date picker)
- **`checkbox`**: Checkbox input
- **`radio`**: Radio button input
- **`file`**: File upload input
- **`submit`**: Submit button
- **`reset`**: Reset button

32.How do you make form inputs required?

You can make an input required by adding the `required` attribute:

```
<input type="text" name="username" required>
```

33.What is the purpose of the label element in forms?

The `<label>` element improves accessibility by providing a clickable label for associated form elements, enhancing user experience.

```
<label for="username">Username:</label>
<input type="text" id="username" name="username">
```

34.How do you group form inputs and why would you do this?

You can group form inputs using the `<fieldset>` element, which is useful for thematically grouping related controls:

```
<fieldset>
    <legend>Personal Information</legend>
    <input type="text" name="name">
    <input type="email" name="email">
</fieldset>
```

35.What is new in HTML5 compared to previous versions?

HTML5 introduced several new features, including:

- New semantic elements (`<article>`, `<section>`, `<nav>`, `<header>`, `<footer>`)
- Native support for audio and video (`<audio>`, `<video>`)
- New form input types and attributes (e.g., `date`, `email`, `required`)
- Improved parsing rules and better error handling.

36.How do you create a section on a webpage using HTML5 semantic elements?

You can create a section using semantic elements like `<section>`, `<article>`, `<header>`, or `<footer>`:

```
<section>
    <header>
        <h1>Section Title</h1>
    </header>
    <p>Content of the section.</p>
</section>
```

37.What is the role of the article element in HTML5?

The `<article>` element represents a self-contained piece of content that can be independently distributed or reused. This can include blog posts, news articles, or user comments. It enhances the semantic structure of the document.

38.Can you explain the use of the nav and aside elements in HTML5?

- **`<nav>`**: This element is used to define a section of navigation links. It helps users find their way around the site and improves accessibility.

```
<nav>
    <ul>
        <li><a href="#home">Home</a></li>
        <li><a href="#about">About</a></li>
    </ul>
</nav>
```

- **`<aside>`**: This element represents content that is tangentially related to the main content. It can be used for sidebars, pull quotes, or related links.

```
<aside>
    <h2>Related Links</h2>
    <ul>
        <li><a href="#link1">Link 1</a></li>
    </ul>
</aside>
```

39.How do you use the figure and figcaption elements?

The `<figure>` element is used to encapsulate media such as images, diagrams, or illustrations, along with an optional `<figcaption>` for a caption.

```
<figure>
    <img src="image.jpg" alt="Description of image">
    <figcaption>Caption for the image</figcaption>
</figure>
```

40.How do you create a table in HTML?

You can create a table using the `<table>` element along with `<tr>` for table rows, `<th>` for header cells, and `<td>` for standard cells.

```
<table>
    <tr>
        <th>Header 1</th>
        <th>Header 2</th>
    </tr>
    <tr>
        <td>Data 1</td>
        <td>Data 2</td>
    </tr>
</table>
```

41.What are thead, tbody, and tfoot in a table?

- **`<thead>`**: Groups the header content in a table.
- **`<tbody>`**: Groups the body content in a table.
- **`<tfoot>`**: Groups the footer content in a table, often used for summary information.

```
<table>
    <thead>
        <tr><th>Header</th></tr>
    </thead>
    <tbody>
        <tr><td>Data</td></tr>
    </tbody>
    <tfoot>
        <tr><td>Footer</td></tr>
    </tfoot>
</table>
```

42.What is a colspan and rowspan?

- **`colspan`**: Attribute used in `<td>` or `<th>` to specify the number of columns a cell should span.

```
<td colspan="2">This cell spans two columns</td>
```

- **`rowspan`**: Attribute used in `<td>` or `<th>` to specify the number of rows a cell should span.

```
<td rowspan="2">This cell spans two rows</td>
```

43.How do you make a table accessible?

To make a table accessible:

- Use `<th>` for header cells.
- Include `<caption>` to provide a summary of the table's purpose.
- Use `scope` attributes to define the relationship between headers and data.

```
<table>
    <caption>Monthly Sales Data</caption>
    <thead>
        <tr>
            <th scope="col">Month</th>
            <th scope="col">Sales</th>
        </tr>
    </thead>
</table>
```

44.How can tables be made responsive?

To make tables responsive:

- Use CSS to enable horizontal scrolling.
- Consider using CSS Grid or Flexbox for layout.
- Use media queries to adjust the table layout on smaller screens.

```
.table-responsive {
    overflow-x: auto;
}
```

45.How do you add audio and video to an HTML document?

You can add audio using the `<audio>` element and video using the `<video>` element:

```
<audio src="audio.mp3" controls></audio>
<video src="video.mp4" controls></video>
```

46.What are the attributes of the video and audio elements?

Common attributes for both `<video>` and `<audio>` include:

- **`src`**: Specifies the source file.
- **`controls`**: Displays playback controls.
- **`autoplay`**: Starts playback automatically.
- **`loop`**: Loops the media indefinitely.
- **`muted`**: Mutes the media by default.

47.How do you provide subtitles or captions for video content in HTML?

You can provide subtitles or captions using the `<track>` element within the `<video>` tag:

```
<video src="video.mp4" controls>
    <track src="subtitles_en.vtt" kind="subtitles" srclang="en"
label="English">
</video>
```

48.What's the difference between embedding and linking media?

- **Embedding**: Involves directly including the media file within the HTML (e.g., using `<video>`, `<audio>`, `<iframe>`).
- **Linking**: Involves providing a hyperlink to the media, allowing users to navigate to it or download it instead of displaying it directly.

49.What is a viewport and how can you set it?

The **viewport** is the visible area of a web page on a device. You can set it using the `<meta>` tag in the `<head>` section:

```
<meta name="viewport" content="width=device-width, initial-scale=1.0">
```

50.Can you describe the use of media queries in HTML?

Media queries are CSS techniques used to apply styles based on the viewport size or device characteristics. They enable responsive design.

```
@media (max-width: 600px) {
    body {
```

```
        background-color: lightblue;
    }
}
```

51.How do you create responsive images with different resolutions for different devices?

You can use the `<picture>` element or the `srcset` attribute to provide different image sources for various resolutions:

```
<picture>
    <source srcset="image-small.jpg" media="(max-width: 600px)">
    <source srcset="image-large.jpg">
    <img src="image-default.jpg" alt="Description">
</picture>
```

52.What is responsive web design?

Responsive web design is an approach to web development that ensures web pages render well on a variety of devices and screen sizes. It uses flexible layouts, images, and CSS media queries to adapt the design to different viewports.

53.How do flexbox and grids help in creating responsive layouts?

Flexbox and **CSS Grid** are layout models that facilitate the creation of responsive designs.

- **Flexbox**: Ideal for one-dimensional layouts (either rows or columns), allowing items to adjust their size and order to fit the available space.

```
.container {
    display: flex;
    flex-wrap: wrap;
}
```

- **Grid**: Best for two-dimensional layouts, enabling complex layouts with rows and columns.

```
.container {
    display: grid;
    grid-template-columns: repeat(auto-fill, minmax(200px, 1fr));
}
```

54.What is accessibility and why is it important in web development?

Accessibility refers to the design of products, devices, services, or environments for people with disabilities. It’s crucial in web development to ensure all users, regardless of their abilities, can access and interact with web content.

55.How do you make a website accessible?

To make a website accessible:

- Use semantic HTML elements.
- Provide alternative text for images (`alt` attribute).
- Ensure keyboard navigability.
- Use ARIA roles where necessary.
- Maintain high contrast between text and background.
- Implement responsive design for all devices.

56.What are ARIA roles and how do you use them?

ARIA (Accessible Rich Internet Applications) roles enhance accessibility by providing additional semantic information to assistive technologies. You use ARIA roles by adding `role` attributes to HTML elements.

```
<div role="navigation">...</div>
```

57.Explain how to use the tabindex attribute.

The **`tabindex`** attribute controls the tab order of elements. It can take values:

- `0`: The element is focusable and follows the tab order.
- `-1`: The element is focusable programmatically but not via tabbing.
- **Positive integers**: Custom tab order, but not recommended for accessibility.

```
<button tabindex="0">Accessible Button</button>
```

58.How do you ensure your images are accessible?

To ensure images are accessible:

- Use the `alt` attribute to provide descriptive text.
- Use `aria-hidden="true"` for decorative images that don't convey information.
- Include captions for meaningful images when necessary.

59.How do you make a navigation bar in HTML?

You can create a simple navigation bar using the `<nav>` element and list items:

```
<nav>
    <ul>
        <li><a href="#home">Home</a></li>
        <li><a href="#about">About</a></li>
    </ul>
</nav>
```

60.What's the significance of breadcrumb navigation?

Breadcrumb navigation provides a secondary navigation scheme that helps users understand their current location within the website's hierarchy. It enhances user experience and can improve SEO.

```
<nav aria-label="Breadcrumb">
```

```
    <ol>
        <li><a href="#">Home</a></li>
        <li><a href="#">Section</a></li>
        <li aria-current="page">Current Page</li>
    </ol>
</nav>
```

61.How do you create a dropdown menu in HTML?

You can create a dropdown menu using nested lists:

```
<nav>
    <ul>
        <li><a href="#home">Home</a></li>
        <li>
            <a href="#services">Services</a>
            <ul class="dropdown">
                <li><a href="#web-design">Web Design</a></li>
                <li><a href="#seo">SEO</a></li>
            </ul>
        </li>
    </ul>
</nav>
```

62.Explain the use of the target attribute in a link.

The **`target`** attribute specifies where to open the linked document. Common values include:

- **`_blank`**: Opens the link in a new tab or window.
- **`_self`**: Opens the link in the same frame (default).
- **`_parent`**: Opens the link in the parent frame.
- **`_top`**: Opens the link in the full body of the window.

```
<a href="https://www.example.com" target="_blank">Open in new tab</a>
```

63.How do you create a slidedown menu?

A slidedown menu typically uses JavaScript for interactivity alongside CSS for styling. Here's a simple example:

```
<button onclick="toggleDropdown()">Menu</button>
<div id="dropdown" style="display:none;">
    <a href="#">Link 1</a>
    <a href="#">Link 2</a>
</div>

<script>
function toggleDropdown() {
    var dropdown = document.getElementById("dropdown");
    dropdown.style.display = dropdown.style.display === "none" ? "block" :
"none";
}
</script>
```

64.What are Web Components and how are they used?

Web Components are a set of web platform APIs that allow you to create reusable custom elements. They consist of three main technologies:

- **Custom Elements**: Define new HTML tags.
- **Shadow DOM**: Encapsulates styles and markup.
- **HTML Templates**: Define reusable markup.

65.What is Shadow DOM and how do you use it?

Shadow DOM allows developers to encapsulate their HTML and CSS, preventing style leakage. You create it using the `attachShadow` method:

```
class MyElement extends HTMLElement {
    constructor() {
        super();
        const shadow = this.attachShadow({ mode: 'open' });
        shadow.innerHTML = `<style>p { color: red; }</style><p>Shadow
DOM!</p>`;
    }
}
customElements.define('my-element', MyElement);
```

66.How do you create a custom HTML element?

To create a custom HTML element, use the Custom Elements API:

```
class MyButton extends HTMLElement {
    constructor() {
        super();
        this.innerHTML = '<button>Click me!</button>';
    }
}
customElements.define('my-button', MyButton);
```

67.Explain HTML templates and their use cases.

HTML templates allow you to define reusable markup that is not rendered when the page loads. You can use the `<template>` element to define the template:

```
<template id="my-template">
    <p>This will not be displayed immediately.</p>
</template>
```

You can then clone and insert it into the DOM using JavaScript.

68.How do you use server-sent events?

Server-Sent Events (SSE) allow a server to push real-time updates to the browser. You use the `EventSource` API:

```
const eventSource = new EventSource('/events');
eventSource.onmessage = function(event) {
    console.log(event.data);
```

```
};
```

69.How do you optimize HTML for search engines?

To optimize HTML for search engines:

- Use semantic HTML.
- Implement proper heading structure (`<h1>`, `<h2>`, etc.).
- Use descriptive `title` and `meta` tags.
- Include alt text for images and proper links.

70.What is semantic HTML and how does it relate to SEO?

Semantic HTML refers to using HTML elements that clearly describe their meaning in a human- and machine-readable way. Semantic elements improve SEO by providing better context for search engines.

71.Explain the significance of heading tags for SEO.

Heading tags (`<h1>` to `<h6>`) structure content hierarchically. The `<h1>` tag is critical for SEO as it usually defines the main topic of a page, helping search engines understand its content.

72.How do structured data and schemas enhance SEO?

Structured data and **schemas** provide search engines with explicit clues about the meaning of a page's content, helping to improve visibility in search results and enabling rich snippets.

73.What are the best practices for using HTML with SEO?

- Use descriptive, keyword-rich title tags.
- Utilize header tags appropriately.
- Include alt text for images.
- Ensure mobile-friendliness and fast load times.
- Use internal and external links wisely.

74.What is the Geolocation API and how is it used?

The **Geolocation API** allows web applications to access the user's geographical location. You can use it as follows:

```
navigator.geolocation.getCurrentPosition((position) => {
    console.log(position.coords.latitude, position.coords.longitude);
});
```

75.How do you utilize local storage and session storage in HTML?

Local Storage and **Session Storage** provide storage capabilities in the browser. You can use them as follows:

```
// Local Storage
```

```
localStorage.setItem('key', 'value');
const value = localStorage.getItem('key');

// Session Storage
sessionStorage.setItem('key', 'value');
const sessionValue = sessionStorage.getItem('key');
```

76.Can you describe the use of the Drag and Drop API?

The **Drag and Drop API** enables users to drag and drop elements within a webpage. You can implement it using event handlers:

```
element.addEventListener('dragstart', (event) => {
    event.dataTransfer.setData('text/plain', event.target.id);
});

targetElement.addEventListener('drop', (event) => {
    const data = event.dataTransfer.getData('text/plain');
    event.target.appendChild(document.getElementById(data));
});
```

77.What is the Fullscreen API and why would you use it?

The **Fullscreen API** allows web applications to request full-screen rendering of an element, providing an immersive user experience. It is often used for videos, games, and interactive content.

```
document.documentElement.requestFullscreen();
```

78.How do you handle character encoding in HTML?

Character encoding is handled using the `<meta>` tag within the `<head>` section. The most common encoding is UTF-8.

```
<meta charset="UTF-8">
```

79.What is the lang attribute and its importance in HTML?

The **`lang` attribute** specifies the language of the content, helping search engines and assistive technologies. It enhances accessibility and SEO.

```
<html lang="en">
```

80.How do you accommodate left-to-right and right-to-left language support in HTML?

You can use the **`dir` attribute** to specify text direction. Use `dir="ltr"` for left-to-right and `dir="rtl"` for right-to-left languages.

```
<p dir="rtl">   </p>
```

81.How do you validate HTML?

You can validate HTML using:

- Online validators (e.g., W3C Markup Validation Service).
- Browser developer tools for checking console errors.

82.What are the benefits of using an HTML preprocessor like Pug (Jade)?

HTML preprocessors like Pug offer:

- Simplified syntax, reducing code verbosity.
- Features like mixins, variables, and loops for dynamic content.
- Easier maintenance and better organization of HTML code.

83.How does a templating engine work with HTML?

A **templating engine** allows you to create HTML templates that can be dynamically populated with data. It processes template files and renders them into standard HTML.

```
h1= title
p Hello, #{name}!
```

84.What are browser developer tools, and how do you use them with HTML?

Browser Developer Tools are built-in tools for inspecting and debugging web pages. You can:

- View HTML structure and modify it live.
- Inspect CSS styles.
- Debug JavaScript code.
- Analyze performance and network requests.

85.What are some common bad practices in HTML?

Common bad practices include:

- Not using semantic elements.
- Overusing inline styles.
- Failing to provide alt text for images.
- Neglecting to use headings correctly.

86.How can you ensure that your HTML code follows best practices?

To ensure best practices:

- Use semantic HTML elements.
- Validate your HTML with tools.
- Follow W3C guidelines and coding standards.

- Keep your code clean and well-organized.

87.What are the benefits of minifying HTML documents?

Minifying HTML reduces file size by removing whitespace, comments, and unnecessary characters, which can improve loading times and reduce bandwidth usage.

88.How do you optimize the loading time of an HTML page?

To optimize loading time:

- Minimize HTML, CSS, and JavaScript files.
- Use asynchronous loading for scripts.
- Implement lazy loading for images and videos.
- Optimize images for web use.

89.What are some popular CSS frameworks that can be integrated with HTML?

Popular CSS frameworks include:

- **Bootstrap**
- **Foundation**
- **Tailwind CSS**
- **Bulma**

90.How do frameworks like Bootstrap simplify HTML development?

Frameworks like Bootstrap provide pre-designed components, grid systems, and utility classes that streamline responsive design and reduce the amount of custom CSS needed.

91.Can you name some JavaScript libraries that enhance HTML interactivity?

Popular JavaScript libraries include:

- **jQuery**
- **React**
- **Vue.js**
- **D3.js** (for data visualization)

92.What are data visualizations in HTML and how can they be implemented?

Data visualizations present data in graphical form, making complex information easier to understand. They can be implemented using libraries like D3.js, Chart.js, or directly using SVG elements.

93.Can you explain how progressive enhancement is applied in HTML?

Progressive enhancement is a strategy that ensures basic functionality is accessible to all users while providing enhanced features for those with better browsers or connections. Start with a simple HTML structure and layer on CSS and JavaScript.

94.How are HTML, CSS, and JavaScript interconnected in web development?

HTML provides the structure, CSS styles the appearance, and JavaScript adds interactivity. Together, they create a cohesive user experience.

95.Discuss the importance of documentation in HTML.

Documentation is essential for:

- Understanding the purpose and usage of HTML elements.
- Ensuring consistency across projects.
- Facilitating collaboration among developers.

96.What updates were introduced in HTML 5.1 and 5.2?

HTML 5.1 introduced updates like:

- Improved support for ARIA.
- Changes to the `<canvas>` element.
- New features for forms.

HTML 5.2 included:

- Updates to the `<dialog>` element.
- Enhanced security features.
- Improvements to custom elements.

97.What future updates do you see coming for HTML?

Future updates may focus on:

- Better support for web components.
- Enhanced accessibility features.
- Integration with emerging technologies like AR/VR.

98.How does HTML continue to evolve with web standards?

HTML evolves through regular updates defined by the W3C and WHATWG, ensuring it stays relevant with modern web practices and user needs.

99.What is the Living Standard and how does HTML adhere to it?

The **Living Standard** is a continuously updated specification for HTML maintained by WHATWG. It reflects the current state of HTML development and includes ongoing changes based on web browser implementations and community feedback.