

Q12. What is XSLT? How is it used to transform XML documents into other formats, such as HTML?

XSLT (Extensible Stylesheet Language Transformations):

- A language used to transform XML documents into other formats, like HTML.
- It uses templates to match elements in the input XML document and generates corresponding output based on defined rules.

Example:

XML

```
<xsl:template match="book">
  <h1><xsl:value-of select="title"/></h1>
  <p>Author: <xsl:value-of select="author"/></p>
</xsl:template>
```

Use code [with caution](#).

This XSLT template transforms a `<book>` element in an XML document into an HTML heading and paragraph.

Q13. Explain the difference between block-level and inline elements in HTML.

Block-level Elements:

- Start on a new line and occupy the full width of their container.
- Examples: `<div>`, `<p>`, `<h1>`, ``, ``

Inline Elements:

- Do not start on a new line and only take up the space they need.
- Examples: ``, `<a>`, ``, ``

Q14. How do you create a form in HTML that allows users to submit data to a web server? Explain the roles of different form elements and attributes.

Creating an HTML Form:

- Use the `<form>` tag to define the form.
- Use various input elements (like `<input>`, `<select>`, `<textarea>`) to collect user data.
- Use the `action` attribute to specify the URL where the form data will be sent.
- Use the `method` attribute to specify the HTTP method (usually `GET` or `POST`).

Form Elements:

- **`<input>`:** Creates different types of input fields (text, email, password, checkbox, radio, submit, reset).
- **`<select>`:** Creates a dropdown list.

- **<textarea>**: Creates a multi-line text area.
- **<label>**: Associates a label with an input element for accessibility.

Q15. What are the ways for inserting CSS to your webpage and explain the concept of cascading order?

Ways to Insert CSS:

1. **Inline CSS**: Directly within the `<style>` attribute of an HTML element.
2. **Internal CSS**: Within the `<style>` tag in the `<head>` section of the HTML document.
3. **External CSS**: In a separate `.css` file linked using the `<link>` tag.

Cascading Order:

- Determines the priority of styles when multiple rules apply to the same element.
- Factors: specificity, importance (`!important`), origin, and order in the stylesheet.

Q16. Design a form by explaining HTML elements like text boxes, checkboxes, combo boxes, reset, and email button.

HTML

```
<form action="submit.php" method="post">
  <label for="name">Name:</label>
  <input type="text" id="name" name="name" required><br>
  <label for="email">Email:</label>
  <input
type="email" id="email" name="email" required><br>
  <label for="event">Event
Type:</label>
  <select id="event" name="event">
    <option value="workshop">Workshop</option>
    <option value="conference">Conference</option>
    <option value="webinar">Webinar</option>
  </select><br>

  <label for="accommodation">Accommodation:</label>
  <input type="radio" id="accommodation_yes" name="accommodation"
value="yes">
  <label for="accommodation_yes">Yes</label>
  <input type="radio" id="accommodation_no" name="accommodation"
value="no">
  <label for="accommodation_no">No</label><br>
  <label for="restrictions">Dietary Restrictions:</label>
  <textarea id="restrictions" name="restrictions"></textarea><br>
  <input type="checkbox" id="terms" name="terms" required>
  <label for="terms">Agree to terms and conditions</label><br>
  <input type="submit" value="Submit">

  <input type="reset" value="Reset">
</form>
```

Use code [with caution](#).

Q17. Why is XML used and what are the differences between XML and HTML?

XML (Extensible Markup Language):

- Designed for data storage, exchange, and manipulation.
- Highly customizable with user-defined tags.
- Strict syntax requirements.

HTML (Hypertext Markup Language):

- Designed for creating web pages and presenting content.
- Predefined tags with specific meanings.
- More forgiving of syntax errors.

Q18. Explain the 5 essential elements of an HTML page structure. Write a code example that demonstrates these elements within the structure of an HTML page.

1. **<!DOCTYPE html>:** Declares the document type.
2. **<html></html>:** Root element of the HTML document.
3. **<head></head>:** Contains metadata about the page (title, links, scripts).
4. **<title></title>:** Sets the title of the page.
5. **<body></body>:** Contains the visible content of the page.

HTML

```
<!DOCTYPE html>
<html>
<head>
  <title>My Web Page</title>
</head>
<body>
  <h1>Welcome!</h1>
  <p>This is some text.</p>
</body>
</html>
```

Use code [with caution](#).

Q19. Differentiate between HTML tags and HTML Elements. List and explain the 4 core HTML attributes.

Tags: Instructions that define the structure of an HTML document. **Elements:** Consist of a starting tag, content, and an ending tag.

Core HTML Attributes:

1. **id:** Unique identifier for an element.
2. **class:** Specifies a class name for an element.
3. **style:** Defines inline styles for an element.
4. **title:** Provides a tooltip for an element.

Q20. Write a code snippet to add an image to the web page and a link to some external web page. What is the difference between the image tag's "src" attribute and the anchor tag's "href" attribute?

HTML

```

<a href="https://example.com">Link to External Page</a>
```

Use code [with caution.](#)

src attribute: Specifies the source URL of the image. **href attribute:** Specifies the destination URL of the link.

Q21. What do you understand by cascading styles in CSS? Explain the different cases through appropriate code snippets.

Cascading Styles: Determines the priority of styles when multiple rules apply to the same element.

Specificity: More specific selectors (e.g., `#id > .class > element`) have higher priority.

CSS

```
/* High specificity */
#my-element {
    color: red;
}

/* Medium specificity */
.my-class {
    color: blue;
}

/* Low specificity */
p {
    color: green;
}
```

Use code [with caution.](#)

Importance: Styles declared with `!important` override other rules.

CSS

```
p {
    color: green;
}

p {
    color: blue !important;
}
```

Use code [with caution.](#)

Origin: Styles defined in the current document override external stylesheets.

Order: Styles defined later in the stylesheet override earlier ones.

Q22. I. Why do we use HTML Forms?

II. Create an HTML form for a simple event registration page.

I. Why do we use HTML Forms?

HTML forms allow users to interact with web pages by providing input data, which can be sent to the server for processing.

II. Event Registration Form:

HTML

```
<form action="submit.php" method="post">
  <label for="name">Name:</label>
  <input type="text" id="name" name="name" required><br>
  <label for="email">Email:</label>
  <input
type="email" id="email" name="email" required><br>
  <label for="phone">Phone Number:</label>
  <input type="tel" id="phone" name="phone"><br>

  <label for="event">Event Type:</label>
  <select id="event" name="event">
    <option value="workshop">Workshop</option>
    <option value="conference">Conference</option>
    <option value="webinar">Webinar</option>
  </select><br>

  <label for="accommodation">Accommodation:</label>
  <input type="radio" id="accommodation_yes" name="accommodation"
value="yes">
  <label for="accommodation_yes">Yes</label>
  <input type="radio" id="accommodation_no" name="accommodation"
value="no">
  <label for="accommodation_no">No</label><br>
  <label for
```

Q23. Image Maps

I. What are image maps?

Image maps allow you to define clickable areas within an image, each linking to a different URL. This creates interactive elements within images.

Code Snippet:

HTML

```


<map name="myMap">
  <area shape="rect" coords="0,0,100,100" href="link1.html">
  <area shape="circle" coords="200,200,50" href="link2.html">
```

</map>

Use code [with caution.](#)

This creates an image map with two clickable areas: a rectangular area in the top-left corner and a circular area in the center.

II. Embedding Audio and Video:

Audio:

HTML

```
<audio controls>
  <source src="audio.mp3" type="audio/mpeg">
  Your browser does not support the audio element.
</audio>
```

Use code [with caution.](#)

Controls:

- `autoplay`: Starts playback automatically.
- `loop`: Repeats playback.
- `muted`: Mutes the audio.
- `controls`: Displays playback controls (play/pause, volume, etc.).

YouTube Video:

HTML

```
<iframe width="560" height="315"
src="https://www.youtube.com/embed/video_id" title="YouTube Video"
frameborder="0" allow="accelerometer; autoplay; clipboard-write; encrypted-
media; gyroscope; picture-in-picture" allowfullscreen></iframe>
```

Use code [with caution.](#)

Controls:

- `autoplay`: Starts playback automatically.
- `loop`: Repeats playback.
- `muted`: Mutes the video.
- `controls`: Displays playback controls (play/pause, volume, etc.).

Q24. Why do we use CSS?

CSS (Cascading Style Sheets):

- Defines the presentation of HTML elements.
- Separates content from style for better organization and maintainability.
- Allows for consistent styling across multiple pages.

Syntax:

CSS

```
selector {  
  property: value;  
  property: value;  
}
```

Use code [with caution.](#)

Example:

CSS

```
h1 {  
  color: blue;  
  font-size: 36px;  
}
```

Use code [with caution.](#)

Inserting CSS:

1. **Inline:** Within the `<style>` attribute of an HTML element.

HTML

```
<h1 style="color: red;">Heading</h1>
```

Use code [with caution.](#)

2. **Internal:** Within the `<style>` tag in the `<head>` section.

HTML

```
<head>  
  <style>  
    h1 { color: red; }  
  </style>  
</head>
```

Use code [with caution.](#)

3. **External:** In a separate `.css` file linked using `<link>`.

HTML

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

Use code [with caution.](#)

Q25. Selectors in CSS

Selectors: Target specific elements in an HTML document.

Types:

1. **Element Selectors:** Target elements based on their tag name.

CSS

```
p {  
  color: blue;  
}
```

Use code [with caution.](#)

2. **Class Selectors:** Target elements with a specific class attribute.

CSS

```
.my-class {  
  font-weight: bold;  
}
```

Use code [with caution.](#)

3. **ID Selectors:** Target elements with a unique ID attribute.

CSS

```
#my-element {  
  background-color: yellow;  
}
```

Use code [with caution.](#)

4. **Universal Selector:** Targets all elements.

CSS

```
* {  
  margin: 0;  
  padding: 0;  
}
```

Use code [with caution.](#)

5. **Grouping Selectors:** Target multiple elements with the same styles.

CSS

```
h1, h2, h3 {  
  font-family: Arial, sans-serif;  
}
```