Module-5 Mernstack – HTML5

Theory Assignment:

Question 1: Difference b/w HTML & HTML5?

Here are the key differences between HTML and HTML5:

1. Doctype Declaration:

- **HTML**: Uses a long and complex doctype declaration (`<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN" "http://www.w3.org/TR/html4/strict.dtd">`).
- **HTML5**: Simplifies the doctype declaration to `<!DOCTYPE html>`.

2. New Elements:

- **HTML**: Limited to older elements like `<div>`, `<span>`, etc.
- **HTML5**: Introduces new semantic elements like `<header>`, `<footer>`, `<article>`, `<section>`, and `<nav>`, improving content structure.

3. Multimedia Support:

- **HTML**: Requires plugins (like Flash) for multimedia content (audio and video).
- **HTML5**: Natively supports multimedia with `<audio>` and `<video>` elements.

4. Graphics Support:

- **HTML**: Limited graphic capabilities, requiring external plugins.
- **HTML5**: Includes `<canvas>` and SVG for drawing graphics directly in the browser.

5. Form Elements and Attributes:

- **HTML**: Basic form elements with limited attributes.
- **HTML5**: Adds new form elements (`<date>`, `<color>`, `<range>`, etc.) and attributes (e.g., `required`, `placeholder`, `autofocus`).

6. APIs:

- **HTML**: No built-in APIs.
- **HTML5**: Provides several APIs like Geolocation API, Web Storage API, Web Workers, and WebSockets.

7. Cross-Browser Compatibility:

- **HTML**: Less consistent across different browsers.
- **HTML5**: Designed to improve cross-browser compatibility and performance.

8. Mobile-Friendly:

- **HTML**: Not optimized for mobile devices.
- **HTML5**: Focuses on mobile-friendly features, ensuring better performance on smartphones and tablets.

9. Deprecated Elements:

- **HTML**: Contains elements like `<font>`, `<center>`, and `<big>` which are now considered obsolete.
- **HTML5**: Deprecates many of these elements and encourages the use of CSS for styling.

10. Storage:

- **HTML**: Relies on cookies for storage.
- **HTML5**: Introduces Web Storage (localStorage and sessionStorage) for more efficient client-side storage.

These differences make HTML5 a more powerful and flexible language for modern web development.

Question 2: What are the additional tags used in HTML5?

HTML5 introduces several new tags that enhance the structure, semantics, and functionality of web pages. Here's a list of the key additional tags used in HTML5:

1. Semantic Elements:

- **`<header>`**: Represents introductory content or navigational links.
- **`<nav>`**: Represents a section of navigation links.
- **`<section>`**: Defines a section in a document, used for thematic grouping of content.
- **`<article>`**: Represents self-contained content that can be independently distributed (e.g., blog posts).
- **`<aside>`**: Represents content tangentially related to the content around it, often used for sidebars.
- **`<footer>`**: Represents footer content, often containing information about the author or related documents.
- **`<main>`**: Represents the main content of the document, unique and central to the document.
- **`<figure>`**: Represents self-contained content, often with a caption (`<figcaption>`).
- **`<figcaption>`**: Provides a caption for the `<figure>` element.

2. Multimedia Elements:

- **`<audio>`**: Used for embedding audio content.
- **`<video>`**: Used for embedding video content.

- **<source>**: Defines multiple media resources for `<audio>` and `<video>`.
- **<track>**: Provides text tracks (subtitles, captions) for `<video>` and `<audio>`.

3. Graphics and Interactive Elements:

- **<canvas>**: Used for drawing graphics on the fly via scripting (usually JavaScript).
- **<svg>**: Supports Scalable Vector Graphics.
- **<details>**: Used as a disclosure widget that shows or hides additional content.
- **<summary>**: Provides a summary, legend, or caption for the `<details>` element.

4. Form Elements:

- **<datalist>**: Provides an autocomplete feature for `<input>` elements.
- **<keygen>**: Generates a key-pair and a certificate request (now deprecated in favor of Web Crypto API).
- **<output>**: Represents the result of a calculation or user action.
- **<progress>**: Represents the completion progress of a task.
- **<meter>**: Represents a scalar measurement within a known range, or a fractional value.

5. Other Elements:

- **<mark>**: Highlights text for reference or importance.
- **<time>**: Represents a specific point in time or duration.
- **<wbr>**: Suggests a word break opportunity.
- **<template>**: Defines a template for client-side content that is not rendered immediately.
- **<dialog>**: Represents a dialog box or interactive component, such as a popup.

These additional tags provide more meaning to the web content and improve the user experience by enhancing accessibility, multimedia handling, and form controls.

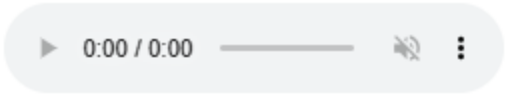
Lab Assignment:

Task:

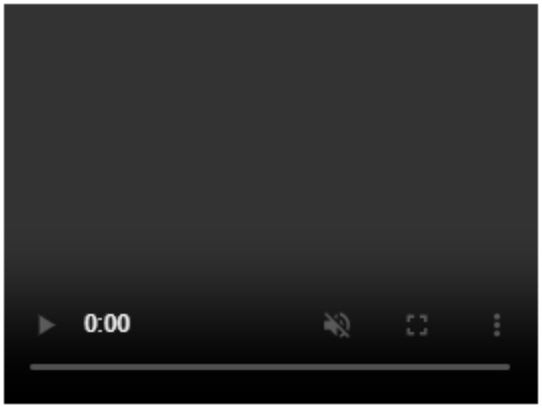
- Create a audio video tag
- Also applied properties like muted loop autoplay
- Create some shape using canvas tag in html
- Create some shape using svg tag in html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>HTML5 Audio, Video, Canvas, and SVG Example</title>
</head>
<body>

    <!-- Audio Tag with properties -->
    <h2>Audio Example</h2>
    <audio controls autoplay loop muted>
        <source src="audiofile.mp3" type="audio/mpeg">
        Your browser does not support the audio element.
    </audio>

    <!-- Video Tag with properties -->
    <h2>Video Example</h2>
    <video width="320" height="240" controls autoplay loop muted>
        <source src="videofile.mp4" type="video/mp4">
        Your browser does not support the video tag.
    </video>

    <!-- Canvas Tag to create shapes -->
    <h2>Canvas Example</h2>
    <canvas id="myCanvas" width="200" height="200" style="border:1px solid #000000;"></canvas>
    <script>
        const canvas = document.getElementById('myCanvas');
        const ctx = canvas.getContext('2d');

        // Draw a rectangle
        ctx.fillStyle = 'blue';
        ctx.fillRect(10, 10, 150, 100);

        // Draw a circle
        ctx.beginPath();
        ctx.arc(100, 150, 50, 0, Math.PI * 2, true);
        ctx.fillStyle = 'green';
        ctx.fill();
    </script>

    <!-- SVG Tag to create shapes -->
    <h2>SVG Example</h2>
    <svg width="200" height="200" xmlns="http://www.w3.org/2000/svg">
        <!-- Draw a rectangle -->
        <rect x="10" y="10" width="150" height="100" fill="blue" />

        <!-- Draw a circle -->
        <circle cx="100" cy="150" r="50" fill="green" />
    </svg>

</body>
</html>
```

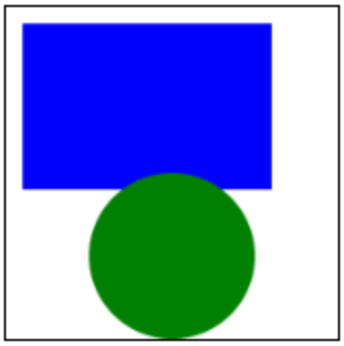
Audio Example



Video Example



Canvas Example



SVG Example