MODULE (HTML5)- 3

WHAT ARE THE NEW TAGS ADDED IN HTML5 ?

HTML5 introduced several new elements and attributes to enhance the markup language. Some of the notable new tags added in HTML5 include:

1. <header>: Used to define introductory content at the beginning of a section or a page.
2. <footer>: Used to define footer content at the end of a section or a page.
3. <nav>: Used to define navigation links.
4. <article>: Used to define a self-contained piece of content, such as a blog post or a news article.
5. <section>: Used to group related content together within a document.
6. <aside>: Used to define content that is tangentially related to the content around it, such as sidebars or pull quotes.
7. <figure>: Used to encapsulate media content, such as images, videos, diagrams, etc., along with their captions using the <figcaption> tag.
8. <main>: Used to define the main content of a document, excluding any header, footer, or sidebar content.
9. <time>: Used to represent dates and times in a machine-readable format.
10. <details> and <summary>: Used to create interactive disclosure widgets, allowing users to show or hide additional information.

These are just a few of the new elements introduced in HTML5. There are several other elements and attributes that were added or updated to improve the structure and semantics of web documents.

HOW TO EMBED AUDIO AND VIDEO IN WEBPAGE ?

To embed audio and video in a webpage, HTML5 provides the `<audio>` and `<video>` elements. Here's how you can use them:

1. Embedding Audio:

```
<audio controls>
        <source src="audio_file.mp3" type="audio/mpeg">
</audio>
```

In this example:

- The <audio> element is used to embed audio content.

- The controls attribute adds playback controls (like play, pause, volume control) to the audio player.
- Inside the <audio> element, you specify one or more <source> elements. Each <source> element contains the src attribute, which specifies the URL of the audio file, and the type attribute, which specifies the MIME type of the audio file. This allows the browser to choose the best-supported format.
- The text "Your browser does not support the audio element." is displayed if the browser does not support the <audio> element or any of the specified audio formats.

2. Embedding video:

```
<video controls width="480" height="270">
        <source src="video_file.mp4" type="video/mp4">
</video>
```

In this example:

- The <video> element is used to embed video content.
- The controls attribute adds playback controls to the video player.
- The width and height attributes specify the dimensions of the video player.
- Inside the <video> element, you specify one or more <source> elements similar to the audio example.
- The text "Your browser does not support the video element." is displayed if the browser does not support the <video> element or any of the specified video formats.

Make sure to replace "audio_file.mp3" and "video_file.mp4" with the paths to your audio and video files, respectively, and adjust the file formats and attributes as needed based on browser support and your specific requirements.

SEMANTIC ELEMENT IN HTML5 ?

Semantic elements in HTML5 are tags that provide meaning to the content they enclose, making it more understandable for both browsers and developers. These elements help structure a web page in a way that is meaningful and descriptive, aiding accessibility and search engine optimization. Some of the commonly used semantic elements in HTML5 include:

1. <header>: Represents introductory content typically containing headings, logos, navigation menus, etc.
2. <footer>: Represents footer content usually containing copyright information, contact details, etc.
3. <nav>: Represents a section with navigation links.

4. <article>: Represents a self-contained piece of content, such as a blog post, news article, etc.
5. <section>: Represents a thematic grouping of content within a document, such as chapters, sections, etc.
6. <aside>: Represents content that is tangentially related to the content around it, such as sidebars, pull quotes, etc.
7. <main>: Represents the main content of the document, excluding header, footer, and sidebar content.
8. <figure>: Represents self-contained content, such as images, videos, diagrams, etc., along with their captions using the <figcaption> tag.
9. <figcaption>: Represents a caption or legend for a <figure> element.

Using semantic elements not only improves the structure and readability of your HTML code but also enhances accessibility and search engine optimization efforts by providing clear context and meaning to the content.

CANVAS AND SVG TAGS.

The <canvas> and <svg> elements are both used to draw graphics on a web page, but they have different underlying technologies and use cases.

Canvas (<canvas>):

- <canvas> is a HTML5 element that provides a bitmap-based drawing surface via JavaScript.
- It allows you to draw 2D graphics dynamically using JavaScript.
- Canvas graphics are immediate mode, meaning that once something is drawn on the canvas, it becomes part of the bitmap and cannot be individually manipulated.
- It's ideal for applications requiring dynamic, interactive graphics such as games, data visualization, and image editing tools.

Example:

```
<canvas id="myCanvas" width="200" height="100"></canvas>
```

Javascript:

```
var canvas = document.getElementById('myCanvas');
var ctx = canvas.getContext('2d');
ctx.fillStyle = 'red';
ctx.fillRect(10, 10, 50, 50);
```

Scalable Vector Graphics (SVG):

- SVG is a markup language for describing two-dimensional graphics in XML format.
- It allows you to create graphics that can be scaled to any size without losing quality.
- SVG graphics are retained mode, meaning that each element within the SVG document is a distinct object that can be manipulated independently.
- It's great for static graphics, logos, icons, charts, and illustrations.

Example

```
<svg width="100" height="100">

    <circle cx="50" cy="50" r="40" stroke="black" stroke-width="3" fill="red" />

</svg>
```

In summary, <canvas> is suitable for dynamic, bitmap-based graphics created through JavaScript, while SVG is better for static, scalable vector graphics defined in XML format. The choice between them depends on the specific requirements of your project, such as interactivity, scalability, and complexity of the graphics.


HTML5