1) What are the new tags added in HTML5?

>>HTML5 introduced several new elements and attributes to enhance the capabilities of web development. Some of the new tags added in HTML5 include:

1. `<header>`: This tag is used to define the header section of a web page or a section within a page. It typically contains elements like the site logo, site title, navigation menus, and other introductory content.

2. `<nav>`: The `<nav>` element is used to define a section of a web page that contains navigation links, menus, or other navigation-related content.

3. `<article>`: The `<article>` tag is used to define a self-contained piece of content that can be distributed and reused independently, such as blog posts, news articles, or forum posts.

4. `<section>`: The `<section>` element is used to create thematic groupings of content within a document, allowing you to structure your content more semantically.

5. `<aside>`: The `<aside>` tag is used for content that is tangentially related to the content around it. It is often used for sidebars, pull quotes, and other content that is not the main focus of the page.

6. `<main>`: The `<main>` element is used to define the main content of a web page. There should be only one `<main>` element per page, and it should contain the primary content.

7. `<figure>` and `<figcaption>`: The `<figure>` element is used to embed multimedia content, images, diagrams, code examples, etc., while the `<figcaption>` element is used to provide a caption or description for the content within the `<figure>`.

8. `<footer>`: The `<footer>` tag is used to define the footer section of a web page or a section within a page. It often contains copyright information, contact details, and other closing content.

9. `<time>`: The `<time>` element is used to mark up dates and times in a machine-readable format, which can be beneficial for search engines and accessibility.

10. `<progress>`: The `<progress>` element allows you to create a progress bar to indicate the completion status of a task or process.

11. `<meter>`: The `<meter>` element is used to represent a measurement within a given range, such as a disk usage gauge or a rating scale.

12. `<details>` and `<summary>`: The `<details>` element creates an interactive disclosure widget that can be used to show or hide additional content. The `<summary>` element is used to provide a label or summary for the disclosure widget.

These are just a few of the new elements introduced in HTML5. HTML5 also brought various new attributes and APIs for multimedia, form handling, and interactive web applications, making it a more versatile language for web development.

2) How to embed audio and video in a webpage?

>>

You can embed audio and video in a webpage using the HTML5 `<audio>` and `<video>` elements. Here's how to do it:

1. Embedding Audio:

To embed audio, you can use the `<audio>` element. You can specify the audio source file using the `src` attribute. Here's an example:

```html

<audio controls>

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

Your browser does not support the audio element.

</audio>

```

In the above code:

- The `controls` attribute adds audio controls like play, pause, and volume.

- The `<source>` element is used to specify the audio file and its type. You can provide multiple `<source>` elements with different formats for better browser compatibility.

- The text within the `<audio>` element is a fallback message that is displayed if the browser doesn't support the `<audio>` element.

2. Embedding Video:

To embed video, you can use the `<video>` element. It works similarly to the `<audio>` element. Here's an example:

```html

<video controls width="640" height="360">

<source src="video.mp4" type="video/mp4">

Your browser does not support the video element.

</video>

```

In the video example:

- The `controls` attribute adds video controls, such as play, pause, and volume.

- The `width` and `height` attributes define the dimensions of the video.

- The `<source>` element specifies the video file and its type. Again, you can provide multiple sources for different formats.

3) Semantic element in HTML5?

>>In HTML5, semantic elements are HTML elements that carry meaning about the structure and content of a web page. They provide a clear and meaningful way to describe the different parts of a webpage, making it more understandable for both browsers and developers. Using semantic elements can improve accessibility and search engine optimization (SEO) while helping with the overall structure of your web page.

Some of the common semantic elements in HTML5 include:

1. `<header>`: Represents the header section of a webpage or a specific section within a page. It typically contains elements like the site title, logo, and navigation menus.

2. `<nav>`: Defines a section containing navigation links, menus, or other navigation-related content.

3. `<main>`: Specifies the main content of a webpage. There should be only one `<main>` element per page.

4. `<article>`: Represents a self-contained piece of content that can be distributed and reused independently, such as blog posts or news articles.

5. `<section>`: Groups related content together and defines a thematic section within a webpage.

6. `<aside>`: Contains content that is tangentially related to the content around it. It is often used for sidebars or content that is not the main focus of the page.

7. `<footer>`: Represents the footer section of a webpage or a specific section within a page. It often contains copyright information, contact details, and closing content.

8. `<figure>` and `<figcaption>`: `<figure>` is used to embed multimedia content, images, diagrams, etc., and `<figcaption>` provides a caption or description for the content within the `<figure>`.

9. `<time>`: Marks up dates and times in a machine-readable format, which can be helpful for search engines and assistive technologies.

10. `<mark>`: Highlights or marks text within a document, often for highlighting search results or other important content.

11. `<meter>`: Represents a measurement within a known range, such as a disk usage gauge or a rating scale.

12. `<progress>`: Creates a progress bar to indicate the completion status of a task or process.

13. `<details>` and `<summary>`: `<details>` defines an interactive disclosure widget for showing or hiding additional content, while `<summary>` provides a label or summary for the disclosure widget.

Using semantic elements makes it easier for web developers to create well-structured and accessible web pages. It also helps search engines better understand the content and hierarchy of your pages, which can have a positive impact on SEO.

4)Canvas and SVG tags

>>Canvas and SVG are two HTML elements that are used for creating graphics and visual elements on web pages. Each has its own approach and use cases:

1. \*\*Canvas\*\*:

The `<canvas>` element is a low-level, immediate-mode graphics API in HTML. It provides a drawing surface that you can use to render 2D and, with WebGL, 3D graphics. Canvas is typically used for drawing dynamic graphics, animations, and interactive content.

To use the `<canvas>` element, you need to use JavaScript to draw on it. You can use the Canvas 2D API for 2D graphics or WebGL for 3D graphics. Here's a simple example of how to create and use a canvas element:

```html

<canvas id="myCanvas" width="400" height="200"></canvas>

<script>

const canvas = document.getElementById('myCanvas');

const context = canvas.getContext('2d');

// Now you can use the Canvas 2D API to draw on the canvas.

context.fillStyle = 'red';

context.fillRect(50, 50, 100, 100);

</script>

```

Canvas is great for real-time animations and interactive games.

2. \*\*SVG (Scalable Vector Graphics)\*\*:

SVG is an XML-based vector graphics format. It allows you to define and manipulate 2D vector graphics in a declarative and accessible way. SVG graphics are resolution-independent, meaning they can be scaled without a loss of quality, which makes them ideal for logos, icons, charts, and other scalable graphic elements.

To embed an SVG in an HTML document, you use the `<svg>` element, and you define shapes and paths using various SVG elements and attributes. Here's a simple example of an SVG drawing:

```html

<svg width="200" height="100">

<rect width="100" height="50" fill="blue" />

<circle cx="50" cy="25" r="20" fill="red" />

</svg>

```

SVG graphics can also be animated and manipulated with CSS or JavaScript.

In summary, Canvas is used for immediate-mode graphics where you draw directly on a pixel-based grid using JavaScript, making it suitable for animations and games. SVG, on the other hand, is a declarative vector graphics format that can be defined within the HTML document, and it is great for static or scalable graphics like icons, logos, and charts. The choice between Canvas and SVG depends on the specific requirements of your project.