1 What are the new tags added in HTML5?

HTML5 introduced several new tags and attributes to enhance the structure and functionality of web documents. Some notable additions include:

1. <article\>

 - Represents a self-contained piece of content that could be distributed and reused independently, such as a blog post or news article.

2. <section\>

 - Defines a thematic grouping of content, typically with a heading. It is used to divide a document into sections.

3. <nav\>

 - Represents a navigation menu, typically containing links to other pages or parts of the same page.

4. <header\> <footer\>

 - These tags represent the header and footer of a section or page, respectively.

5. <aside\>

 - Represents content that is tangentially related to the content around it, such as a sidebar or pull quote.

6. <figure\><figcaption\>

 - Used to group media content (like images or videos) and their captions.

7. <mark\>

 - Highlights text for reference or notation purposes.

8. <progress\>

 - Represents the completion progress of a task, such as the loading of a web page.

9. <output\>

 - Represents the result of a calculation or user action.

10. <time\>

 - Represents a specific period in time or a range of time, typically used with the "datetime" attribute.

11. <datalist\> <option\>

 - Used to create a predefined list of options for input controls like \<input type="text"\>.

12. <details\> <summary\>

 - Used to create a disclosure widget from which the user can obtain additional information.

13. <main\>

   - Represents the main content of the \<body\> in a document, excluding headers, footers, and sidebars.

14. <canvas\>

   - Provides a space for drawing graphics using JavaScript.

These new tags help to provide more semantic meaning to the structure of web documents and make it easier for developers to create accessible and well-organized content.

2 How to embed audio and video in a webpage?

It seems like there might be a typo in your question, but I'll assume you're asking for more information or clarification on embedding audio and video in a webpage. If you have specific questions or need further explanation on any part of the process, feel free to ask. Here's a bit more detail on the steps:

Embedding Audio:

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

  <h1>Audio Example</h1>

  <audio controls>
    <source src="your-audio-file.mp3" type="audio/mp3">
    Your browser does not support the audio element.
  </audio>
</body>
</html>
```

Explanation:

- The `audio` element is used to embed audio content in the page.
- The `controls` attribute adds audio playback controls (play, pause, volume).
- The `<source>` element inside provides different sources for the audio file, allowing for browser compatibility.

Embedding Video:

```
html

<!DOCTYPE html>

<html lang="en">

<head>

  <meta charset="UTF-8">

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

  <title>Video Example</title>

</head>

<body>

  <h1>Video Example</h1>

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

    <source src="your-video-file.mp4" type="video/mp4">

    Your browser does not support the video element.

  </video>

</body>

</html>
```

Explanation:

- The `video` element is used to embed video content in the page.
- The `width` and `height` attributes set the dimensions of the video player.
- The `controls` attribute adds video playback controls.
- Similar to audio, the `<source>` element provides different sources for the video file.

Remember to replace `"your-audio-file.mp3"` and `"your-video-file.mp4"` with the actual paths to your audio and video files. If you have more specific questions or if there's a particular aspect you'd like more information on, please let me know!

3 Semantic element in HTML5?

HTML5 introduced several semantic elements that provide more meaning and structure to the content of a webpage. Semantic elements help convey the intended meaning of different parts of a document, making it more understandable for both browsers and developers. Here are some of the key semantic elements introduced in HTML5:

1. <article>

   - Represents a self-contained piece of content that could be distributed and reused independently, such as a blog post or news article.

2. <section>

   - Defines a thematic grouping of content, typically with a heading. It is used to divide a document into sections.

3. <nav>

   - Represents a navigation menu, typically containing links to other pages or parts of the same page.

4. <header>

   - Represents the header of a section or page. It can contain headings, logos, and introductory content.

5. <footer>

   - Represents the footer of a section or page. It typically contains metadata, copyright information, and links to related documents.

6. <aside>

   - Represents content that is tangentially related to the content around it, such as a sidebar or pull quote.

7. <figure>

   - Represents any content that is referenced from the main content, such as images, charts, or code snippets.

8. <figcaption>

   - Represents a caption or legend for the content inside a `<figure>` element.

9. <main>

   - Represents the main content of the `<body>` in a document, excluding headers, footers, and sidebars.

10. <mark>

- Highlights text for reference or notation purposes.

11. <time>

   - Represents a specific period in time or a range of time. It is often used with the "datetime" attribute to provide machine-readable date and time information.

12. <progress>

   - Represents the completion progress of a task, such as the loading of a web page.

13. <output>

   - Represents the result of a calculation or user action.

These elements enhance the structure and meaning of web documents, making it easier for both developers and assistive technologies to understand the content and improve accessibility. When building a webpage, it's a good practice to use semantic elements appropriately to convey the intended meaning of different parts of your document.

4 Canvas and SVG tags ?

Both the <canvas> and <svg> elements in HTML5 are used for drawing graphics on a webpage, but they have different approaches and use cases.

<canvas> Element:

The<canvas>element is a drawing surface that allows you to use JavaScript to create dynamic graphics, animations, and interactive content. It provides a pixel-based rendering context that is well-suited for rendering complex graphics, such as games or data visualizations.

Example:

html

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Canvas Example</title>
</head>
<body>
  <h1>Canvas Example</h1>
  <canvas id="myCanvas" width="400" height="200"></canvas>
  <script>
    // JavaScript code to draw on the canvas
    var canvas = document.getElementById("myCanvas");
var ctx = canvas.getContext("2d");
        // Drawing a rectangle
        ctx.fillStyle = "blue";
        ctx.fillRect(50, 50, 100, 50);
    </script>
</body>
</html>
```

In this example, a blue rectangle is drawn on the canvas using JavaScript.

<svg> Element:

The <svg> element, on the other hand, is an XML-based markup language for describing two-dimensional vector graphics. SVG stands for Scalable Vector Graphics. It is resolution-independent and can be scaled without loss of quality. SVG is suitable for static graphics, logos, icons, and illustrations.

Example:

html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>SVG Example</title>
</head>
<body>
    <h1>SVG Example</h1>
    <svg width="400" height="200">
        <!-- Drawing a rectangle -->
        <rect x="50" y="50" width="100" height="50" fill="blue" />
    </svg>
</body>
</html>
```

In this example, a blue rectangle is drawn using SVG markup.

Key Differences:

1. Rendering Model:

   - <canvas> uses a bitmap-based rendering model, where you manipulate pixels directly.

   - <svg> uses a vector-based rendering model, where graphics are described as mathematical shapes and can be scaled without losing quality.

2. Interactivity:

   - <canvas> is suitable for interactive content and animations.

   - <svg> supports interactive elements like hyperlinks, and its elements can be styled and animated using CSS.

3. Accessibility:

   - Content drawn on <canvas> is considered a raster image and may not be as accessible as SVG, which is text-based and can be easily styled and manipulated.

The choice between <canvas> and <svg> depends on your specific use case. If you need dynamic and interactive graphics, <canvas> may be more suitable. If you require resolution-independent graphics and want to work with vector-based graphics, <svg> is a better choice.