**Module (HTML5) – 3**

**Que.1 What are the new tags added in HTML5?**

**Ans.** HTML5 introduced several new tags that provide more semantic meaning to the structure of a web page. Some of the new tags added in HTML5 are:  
  
1. `<header>`: Represents the introductory content or a group of navigational links at the top of a document or section.  
2. `<nav>`: Defines a section of navigation links.  
3. `<section>`: Represents a standalone section of content within a document.  
4. `<article>`: Represents a self-contained composition that can be independently distributed or syndicated.  
5. `<aside>`: Defines content that is tangentially related to the main content, such as sidebars or pull quotes.  
6. `<footer>`: Represents the footer of a document or section, typically containing information about the author, copyright, or contact details.  
7. `<main>`: Specifies the main content of a document, excluding header, footer, and navigation.  
8. `<figure>` and `<figcaption>`: Used together to represent self-contained content, such as images, diagrams, or illustrations, along with their captions.  
9. `<time>`: Represents a specific time or a range of time.  
10. `<mark>`: Highlights or marks a specific portion of text for reference or emphasis.  
11. `<progress>`: Represents the progress of a task or process.  
12. `<meter>`: Represents a scalar measurement within a known range, such as disk usage or voting percentages.  
13. `<datalist>`: Provides a list of predefined options for an `<input>` element.  
14. `<output>`: Displays the result of a calculation or user action.  
  
These are just a few examples of the new tags introduced in HTML5. They help improve the structure and semantics of web pages, making them more accessible and easier to understand for both humans and machines.

**Que.2 How to embed audio and video in a webpage?**

**Ans.** To embed audio and video in a webpage, you can use HTML5's `<audio>` and `<video>` elements. Here's how you can do it:  
  
Embedding Audio:  
1. Prepare your audio file in a supported format like MP3, WAV, or OGG.  
2. Use the `<audio>` element in your HTML code and specify the source file using the `src` attribute.  
  
Example:

**html**

**<audio src="audio\_file.mp3" controls></audio>**

In the above code, the `controls` attribute adds a default set of playback controls to the audio player.  
  
Embedding Video:  
1. Prepare your video file in a supported format like MP4, WebM, or OGG.  
2. Use the `<video>` element in your HTML code and specify the source file using the `src` attribute.  
  
Example:

**html**

**<video src="video\_file.mp4" controls></video>**

Similar to the audio element, the `controls` attribute adds default playback controls to the video player.  
  
You can further customize the audio and video elements by adding additional attributes and options. For example, you can specify the width and height of the video, enable autoplay, loop the playback, and more.  
  
Note: It's important to provide alternative formats for audio and video files to ensure compatibility across different browsers. You can do this by including multiple `<source>` elements within the audio or video element, each with a different file format.  
  
Example:

**html**

**<video controls>**

**<source src="video\_file.mp4" type="video/mp4">**

**<source src="video\_file.webm" type="video/webm">**

**<source src="video\_file.ogg" type="video/ogg">**

**Your browser does not support the video tag.**

**</video>**

By following these steps, you can embed audio and video files into your webpage using HTML5.

**Que.3 Semantic element in HTML5?**

**Ans.** In HTML5, semantic elements are tags that provide meaning and structure to the content within a web page. They help search engines, screen readers, and other technologies understand the purpose and organization of the content. Here are some commonly used semantic elements in HTML5:

1. `<header>`: Represents the introductory content or a container for a group of introductory content at the top of a page or section.

2. `<nav>`: Defines a section of navigation links.

3. `<main>`: Represents the main content of a document. It should be unique to the document and not repeated across multiple pages.

4. `<article>`: Represents a self-contained composition within a document, such as a blog post, news article, or forum post.

5. `<section>`: Defines a standalone section of content within a document, such as chapters, tabs, or different parts of a page.

6. `<aside>`: Represents content that is tangentially related to the main content, such as sidebars, pull quotes, or advertising.

7. `<footer>`: Represents the footer of a document or a section. It typically contains information about the author, copyright, or links to related documents.

These semantic elements help improve the accessibility, maintainability, and search engine optimization of a web page by providing clear structure and meaning to the content. It is recommended to use these elements appropriately based on the purpose and organization of your content.

**Que.4 Canvas and SVG tags**

**Ans**. Canvas and SVG are two different approaches for creating graphics and visual elements on a web page.

Canvas:

The `<canvas>` tag is an HTML element that provides a drawing surface on which you can dynamically render graphics using JavaScript. It allows you to draw shapes, images, and animations programmatically. The canvas element is a bitmap-based drawing technology, meaning that it works with pixels. You can manipulate individual pixels or draw complex shapes by using JavaScript to control the canvas context.

SVG:

SVG stands for Scalable Vector Graphics. It is an XML-based markup language for describing two-dimensional vector graphics. Unlike the `<canvas>` tag, SVG is resolution-independent and uses mathematical equations to define shapes, lines, and curves. SVG elements are treated as DOM (Document Object Model) nodes, which means they can be styled, animated, and interacted with using JavaScript.

When to use Canvas:

Canvas is suitable for scenarios where you need to create dynamic and interactive graphics, such as games, data visualizations, or complex animations. It provides low-level pixel manipulation and is well-suited for rendering real-time graphics.

When to use SVG:

SVG is ideal for scenarios where you need to create scalable and resolution-independent graphics, such as icons, logos, or illustrations. It is particularly useful when you want to manipulate and animate individual elements within the graphic, as SVG elements are part of the DOM and can be easily targeted and modified using JavaScript.

In summary, the `<canvas>` tag is used for bitmap-based graphics and allows for pixel-level manipulation, while SVG is used for scalable vector graphics and provides a more structured and interactive approach to creating graphics on the web. The choice between Canvas and SVG depends on the specific requirements and nature of the graphics you want to create.