**Module (HTML5) – 3**

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

**Ans**.

New tags introduced by HTML5 improve the semantic structure of web content. Here are a some of the significant new tags along with their justifications and illustrations:

1. "header": Describes a document's or a section's header. Usually, it has the site's logo, navigation menu, or an introduction.

**Example:**

html

<header>

<h1>My Website</h1>

<nav>

<a href="#">Home</a>

<a href="#">About</a>

<a href="#">Contact</a>

</nav>

</header>

2. \<nav>: Represents a section containing navigation links. It is typically placed within the header or footer of a document.

Example:

html

<nav>

<a href="#">Home</a>

<a href="#">About</a>

<a href="#">Contact</a>

</nav>

3. \<section>: Defines a standalone section within a document. It represents a logical grouping of content, such as chapters, tabbed content, or articles.

Example:

html

<section>

<h2>About Us</h2>

<p>Lorem ipsum dolor sit amet, consectetur adipiscing elit.</p>

</section>

4. \<article>: Represents a self-contained composition that can be independently distributed or reused, such as a blog post, news article, or forum post.

Example:

html

<article>

<h2>Blog Post Title</h2>

<p>Lorem ipsum dolor sit amet, consectetur adipiscing elit.</p>

</article>

5. \<aside>: Defines a section of content that is tangentially related to the main content. It is often used for sidebars, pull quotes, or advertising sections.

Example:

html

<aside>

<h3>Advertisement</h3>

<p>Check out our new product!</p>

</aside>

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

Example:

html

<footer>

<p>&copy; 2023 My Website. All rights reserved.</p>

</footer>

These are just a few examples of the new tags introduced in HTML5. They help provide clearer semantics to the structure of web documents and improve accessibility and SEO.

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

**Ans.**

To embed audio and video in a webpage using HTML5, you can use the `<audio>` and `<video>` elements. Here are simple examples of how to do this along with basic HTML layouts:

### 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="example.mp3" type="audio/mpeg">

Your browser does not support the audio element.

</audio>

</body>

</html>

```

In this example, we use the `<audio>` element to embed an audio file (`example.mp3`) and add the `controls` attribute to provide playback controls (play, pause, volume) to the user. The `<source>` element inside `<audio>` specifies the source file and its type. The text inside `<audio>` is displayed if the browser doesn't support the audio element.

### 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 controls width="640" height="360">

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

Your browser does not support the video element.

</video>

</body>

</html>

```

In this example, we use the `<video>` element to embed a video file (`example.mp4`). The `controls` attribute adds video playback controls, and you can specify the `width` and `height` attributes to set the video dimensions. Inside `<video>`, the `<source>` element specifies the video source and its type. Like with audio, the text inside `<video>` is shown if the browser can't play the video.

These examples provide a basic structure for embedding audio and video in a webpage. You can replace `"example.mp3"` and `"example.mp4"` with the actual paths to your audio and video files. Additionally, you can further customize the appearance and functionality of the audio and video elements using CSS and JavaScript as needed for your specific design and requirements.

1. **Semantic element in HTML5?**

**Ans**

In HTML5, semantic elements are tags that offer meaningful structure and define the material contained within them. They improve a web page's accessibility, search engine optimisation (SEO), and general structure. Browsers, screen readers, and other technologies use semantic components to comprehend the purpose and hierarchy of distinct portions of a webpage.

Here are some examples of semantic elements in HTML5:

1. `<header>`: Represents the introductory or navigational section of a webpage.

Html

<header>

<h1>Welcome to My Website</h1>

<nav>

<ul>

<li><a href="#">Home</a></li>

<li><a href="#">About</a></li>

<li><a href="#">Services</a></li>

<li><a href="#">Contact</a></li>

</ul>

</nav>

</header>

1. `<nav>`: Defines a section containing navigation links.

Html

<nav>

<ul>

<li><a href="#">Home</a></li>

<li><a href="#">Products</a></li>

<li><a href="#">Services</a></li>

<li><a href="#">Contact</a></li>

</ul>

</nav>

1. `<main>`: Represents the main content area of a webpage.

Html

<main>

<h1>About Us</h1>

<p>Lorem ipsum dolor sit amet, consectetur adipiscing elit. ...</p>

</main>

1. `<article>`: Represents a self-contained composition that could be independently distributed or syndicated.

Html

<article>

<h2>Blog Post Title</h2>

<p>Lorem ipsum dolor sit amet, consectetur adipiscing elit. ...</p>

</article>

1. `<section>`: Defines a thematic grouping of content within a document.

Html

<section>

<h2>Our Team</h2>

<div class="team-member">

<h3>John Doe</h3>

<p>Position: Web Developer</p>

</div>

<div class="team-member">

<h3>Jane Smith</h3>

<p>Position: Graphic Designer</p>

</div>

</section>

These are just a few examples of semantic elements in HTML5. Using these elements appropriately can improve the structure and accessibility of your web pages, making them easier to understand for both humans and machines.

1. **Canvas and SVG tags**

**Ans.**

Certainly! Canvas and SVG are two different ways to create graphics and visual elements in HTML5. Here's a brief overview of each with simple examples:

1. Canvas:

- Concept: The `<canvas>` element is used for drawing graphics, animations, and other visual elements dynamically using JavaScript. It provides a low-level 2D drawing context that allows you to draw shapes, lines, text, and images.

- Example:

```html

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

```

```javascript

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

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

// Draw a rectangle

ctx.fillStyle = 'blue';

ctx.fillRect(10, 10, 150, 80);

// Draw text

ctx.font = '20px Arial';

ctx.fillStyle = 'white';

ctx.fillText('Hello, Canvas!', 20, 50);

```

In this example, we create a canvas element and use JavaScript to draw a blue rectangle and add text to it.

2. SVG (Scalable Vector Graphics):

- Concept: SVG is an XML-based language for describing two-dimensional vector graphics. It's a markup language, so you can create and manipulate SVG graphics directly in HTML using XML tags. SVG graphics are resolution-independent and can be scaled without loss of quality.

- Example:

```html

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

<rect x="10" y="10" width="150" height="80" fill="blue" />

<text x="20" y="50" font-size="20" fill="white">Hello, SVG!</text>

</svg>

```

In this example, we use SVG markup to create a blue rectangle and add text to it. SVG elements are self-contained and can be styled with attributes like `fill` and `stroke`.

Summary:

- Use `<canvas>` for dynamic, pixel-based graphics and animations. It requires JavaScript for rendering and interaction.

- Use SVG for static or dynamic, resolution-independent vector graphics. You can create and manipulate SVG directly using XML-like markup within your HTML.

Both Canvas and SVG have their strengths, and the choice between them depends on the specific requirements of your project. Canvas is great for complex animations and games, while SVG is more suitable for scalable, resolution-independent graphics like icons and charts.