**Module (HTML5) – 3**

* What are the new tags added in HTML5?
  + HTML5 introduced several new tags to provide better semantic structure and improve the organization of web documents. Some of the notable new tags introduced in HTML5 include:
    - <header>: Defines a header for a document or section.
    - <footer>: Defines a footer for a document or section.
    - <nav>: Defines a set of navigation links.
    - <article>: Defines an independent, self-contained piece of content.
    - <section>: Defines a section in a document.
    - <aside>: Defines content aside from the content (like a sidebar).
    - <main>: Defines the main content of a document.
    - <figure>: Defines self-contained content, like images, diagrams, photos, code snippets, etc.
    - <figcaption>: Defines a caption for a <figure> element.
    - <details>: Defines additional details that the user can view or hide.
    - <summary>: Defines a summary or a heading for the <details> element.
    - <time>: Defines a specific period in time or a timestamp.
    - <mark>: Defines text highlighted for reference or notation purposes.
    - <meter>: Represents a scalar measurement within a known range (e.g., a gauge).
    - <progress>: Represents the progress of a task.
    - <datalist>: Provides a list of predefined options for <input> elements.
    - <output>: Represents the result of a calculation or user action.
  + These new tags help developers create more meaningful and accessible web documents, improving both the structure and semantics of HTML.
* How to embed audio and video in a webpage?
  + To embed audio and video in a webpage using HTML5, you can use the <audio> and <video> elements respectively. Here's how you can do it:
  + Embedding Audio:
    - <audio controls>
    - <source src="audio\_file.mp3" type="audio/mpeg">
    - Your browser does not support the audio element.
    - </audio>
  + The <audio> element is used to embed audio content in a webpage.
  + The controls attribute adds playback controls (play, pause, volume, etc.) to the audio player.
  + Inside the <audio> element, you can provide one or more <source> elements, each with a different audio file format. The browser will choose the first compatible format.
  + The type attribute specifies the MIME type of the audio file.
  + The text "Your browser does not support the audio element." is displayed if the browser does not support the <audio> element.
  + Embedding Video:
    - <video controls width="640" height="360">
    - <source src="video\_file.mp4" type="video/mp4">
    - Your browser does not support the video element.
    - </video>
  + The <video> element is used to embed video content in a webpage.
  + The controls attribute adds playback controls (play, pause, volume, etc.) to the video player.
  + The width and height attributes specify the dimensions of the video player.
  + Inside the <video> element, you can provide one or more <source> elements, each with a different video file format. The browser will choose the first compatible format.
  + The type attribute specifies the MIME type of the video file.
  + The text "Your browser does not support the video element." is displayed if the browser does not support the <video> element.
* Semantic element in HTML5?
  + Semantic elements in HTML5 are tags that convey meaning about the content they contain. These elements help describe the structure of a webpage more accurately, making it easier for both humans and search engines to understand the content. Some of the most commonly used semantic elements in HTML5 include:
    - <header>: Represents introductory content or a group of introductory content. Typically contains headings, logos, navigation menus, etc.
    - <footer>: Represents a footer for a section or the whole document. Typically contains information such as copyright notices, author information, contact details, etc.
    - <nav>: Represents a section of navigation links. Typically contains links to other pages or sections within the same page.
    - <article>: Represents a self-contained piece of content that could be independently distributed or reused. For example, a blog post, a news article, a forum post, etc.
    - <section>: Represents a generic section of content within a document. It's often used to group related content together.
    - <aside>: Represents content that is tangentially related to the content around it. It's typically used for sidebars, pull quotes, related links, etc.
    - <main>: Represents the main content of a document. It should not include content that is repeated across multiple pages, such as navigation menus or footers.
    - <figure>: Represents self-contained content, such as images, illustrations, diagrams, code snippets, etc. It's often accompanied by a <figcaption> element to provide a caption for the content.
    - <figcaption>: Represents a caption or legend for a <figure> element.
    - <time>: Represents a specific point in time or a duration. It can be used to mark up dates, times, or time intervals.
    - <mark>: Represents text highlighted for reference or notation purposes.
    - <meter>: Represents a scalar measurement within a known range, such as disk usage, completion progress, etc.
    - <progress>: Represents the completion progress of a task or process.
  + These semantic elements help improve accessibility, SEO, and maintainability of web pages by providing clearer structure and meaning to the content.
* Canvas and SVG tags
  + In HTML5, both the <canvas> and <svg> elements are used for graphics rendering, but they serve different purposes and have different approaches to creating and manipulating graphics.
  + <canvas> Element:
  + The <canvas> element is used to draw graphics, animations, and interactive content dynamically using JavaScript.
  + It provides a bitmap-based rendering context, which means that the graphics drawn on the canvas are essentially a raster image.
  + The canvas is suitable for complex, dynamic rendering such as interactive games, data visualizations, and complex animations.
  + Graphics drawn on the canvas are not part of the DOM, so they can't be styled or manipulated directly using CSS. All rendering and manipulation are done programmatically through JavaScript.
  + Example usage:
    - <canvas id="myCanvas" width="400" height="200"></canvas>
    - <script>
    - var canvas = document.getElementById('myCanvas');
    - var context = canvas.getContext('2d');
    - // Drawing operations using JavaScript
    - </script>
  + <svg> Element:
  + The <svg> element is used to create vector graphics, allowing for scalable, resolution-independent graphics.
  + SVG stands for Scalable Vector Graphics, and it defines graphics in XML format.
  + SVG graphics are part of the DOM, which means they can be styled, animated, and manipulated using CSS and JavaScript.
  + SVG is suitable for static or semi-static graphics such as icons, logos, illustrations, and simple animations.
  + Graphics in SVG can be created using various shapes, paths, text, and other graphical elements defined in the SVG specification.
  + Example usage:
    - <svg width="400" height="200">
    - <circle cx="100" cy="100" r="50" fill="red"/>
    - <rect x="200" y="50" width="100" height="100" fill="blue"/>
    - </svg>
  + Comparison:
  + Use <canvas> for dynamic and complex graphics that require extensive scripting and interactivity.
  + Use <svg> for static or semi-static graphics, especially when scalability and accessibility are important.
  + <canvas> is pixel-based, while <svg> is vector-based.
  + <canvas> is suitable for complex animations, whereas <svg> is better for simple animations or static graphics.
  + Choose between <canvas> and <svg> based on your specific requirements for graphics rendering, interactivity, and scalability.