**Module (HTML5 ) – 3**

* What are the new tags added in HTML5?
* HTML5 introduced several new tags that were designed to improve the structure, accessibility, and functionality of web pages. Here are some of the most important new tags introduced with HTML5:

1. **<article>**

* Represents a self-contained piece of content that can be independently distributed or reused (e.g., a blog post, news article, or forum post).

1. **<aside>**

* Represents content that is tangentially related to the content around it, like sidebars, pull quotes, or advertising.

1. **<details>**

* Represents a disclosure widget from which the user can obtain additional information or details. It is typically used in conjunction with <summary>.

1. **<summary>**

* Provides a summary or heading for the <details> element. It is visible by default and can be clicked to reveal or hide the content inside <details>.

1. **<fig caption>**

* Provides a caption or description for a <figure> element, which typically contains media such as images, charts, or diagrams.

1. **<figure>**

* Used to group media content (images, videos, charts, etc.) and their caption, often used in conjunction with <fig caption>.

1. **<footer>**

* Represents a footer for a section or page, typically containing information like copyright notices, links to related documents, or contact information.
* How to embed audio and video in a webpage?
* Embedding audio and video in a webpage in HTML5 is straightforward using the <audio> and <video> elements. These elements allow you to include media files directly in your HTML document without needing third-party plugins like Flash.

1. **Embedding Audio**

* To embed audio, you use the <audio> element, which allows you to play audio files like MP3, Ogg, or WAV directly in the browser.
* The **controls** attribute adds play/pause, volume control, and other media controls to the audio player.
* The **<source>** element specifies multiple audio file formats to ensure compatibility across different browsers.
* The fallback message (e.g., "Your browser does not support the audio element") appears if the browser doesn't support the <audio> element.

1. **Embedding Video**

* The <video> element works similarly to the <audio> element, allowing you to embed video files in your webpage.
* **controls**: Displays the video controls (play/pause, volume, etc.).
* The <source> elements provide alternative video formats for different browsers.
* The fallback message appears if the browser doesn't support the <video> tag.

1. **Additional Video and Audio Attributes:**

* **preload**: Specifies if and how the video/audio file should be loaded when the page loads.
* preload="auto": The browser will load the entire video/audio file.
* preload="metadata": The browser will load only the metadata (like duration, dimensions, etc.).
* preload="none": The browser will not load the video/audio until the user presses play.
* **poster** (for videos): Specifies an image to show as a preview before the video plays.

1. **Compatibility Considerations:**

* Not all browsers support every media format, so it's a good idea to provide multiple file formats for both audio and video (e.g., MP3, Ogg, WebM for audio; MP4, Ogg, WebM for video).
* **MP4 (H.264 video + AAC audio)** is the most widely supported video format, while **MP3** is the most widely supported audio format.
* Semantic element in HTML5?
* In HTML5, **semantic elements** refer to elements that carry meaning about the content they contain. These elements help define the structure of a web page, making it more readable for both humans and machines (such as search engines and screen readers). They provide clarity about the type of content they enclose, which improves accessibility, SEO (Search Engine Optimization), and code maintainability.

Why Use Semantic Elements?

1. **Improved Accessibility**:

* Screen readers and other assistive technologies can more easily understand the structure and meaning of the page.

1. **Better SEO**:

* Search engines give preference to well-structured, semantic HTML as it helps them understand the content and context of the page.

1. **Code Maintainability**:

* Code becomes more readable, easier to maintain, and more logically organized.

**Key Semantic Elements in HTML5**

1. **<header>**

* Represents a container for introductory content or a group of navigational links.
* Typically contains elements like the site logo, navigation menus, or a page title.

1. **<footer>**

* Represents a footer for a section or page, often containing copyright information, contact links, or related documents.
* It can also be used for legal or author information related to the content.

1. **<article>**

* Represents a self-contained piece of content that can be independently distributed or reused (e.g., blog posts, news articles, forum posts).
* Each <article> should make sense on its own, even outside the context of the page it appears on.

1. **<section>**

* Represents a thematic grouping of content, typically with its own heading. It is used to divide a page into sections, like chapters, subsections, or distinct parts.
* A <section> element usually contains a heading (e.g., <h1>, <h2>).

1. **<nav>**

* Represents a section of navigation links.
* Typically used for a menu or a set of navigation controls that allow users to navigate through the website.

1. **<main>**

* Represents the main content of the <body> of the document.
* It should be used for content that is directly related to or expands upon the central topic of the document, and it should not include things like headers, footers, or sidebars.
* Canvas and SVG tags?
* In HTML5, both the **<canvas>** and **<svg>** tags are used to create graphics, but they work in different ways. Let's explore each one, their differences, and how they are used.

1. **<canvas> Tag**

* The <canvas> element is used to draw graphics via scripting, typically JavaScript. It's a container that allows you to create 2D graphics and animations directly in the browser. Unlike SVG, <canvas> renders graphics dynamically, meaning the content is created via code rather than being defined in the markup.

**Key Features:**

* **Dynamic Rendering**: Graphics are drawn using JavaScript code.
* **Bitmap Rendering**: It generates raster (pixel-based) graphics, which means each time a drawing is made, it's essentially creating a "bitmap" image.
* **No built-in content**: The canvas initially doesn't contain anything until JavaScript draws on it.

**Key Methods & Properties:**

* Get Context('2d'): Gets the 2D drawing context for the canvas (essentially enabling drawing on it).
* Fill React(x, y, width, height): Draws a filled rectangle.
* Clea rarest(x, y, width, height): Clears part of the canvas.
* begin Path() and arc(x, y, radius, start Angle, end Angle): Draw shapes, like arcs and circles.
* draw Image(): Draw an image onto the canvas.
* fill Text(text, x, y): Draw text on the canvas.

Use Cases:

* Drawing shapes, graphs, and charts dynamically.
* Creating game graphics or interactive animations.
* Generating images on the fly, such as photo editing tools or image effects.

1. **<svg> Tag**

* SVG (Scalable Vector Graphics) is a markup language used to create vector-based graphics. The <svg> element is used to define the graphics directly in the HTML. Unlike <canvas>, which is bitmap-based, SVG uses **vector graphics** (paths, lines, shapes, and text), which are scalable and resolution-independent.

**Key Features:**

* **Vector Graphics**: SVG is vector-based, meaning the images are defined by mathematical equations rather than pixels. As a result, they scale infinitely without losing quality.
* **Static or Animated**: SVG can be both static (unchanging) or animated (using CSS or JavaScript).
* **DOM Accessible**: Each element in an SVG is part of the DOM, meaning you can manipulate it directly using JavaScript or CSS.

**Key Elements:**

* **<circle>**: Draws a circle.
* **<rect>**: Draws a rectangle.
* **<line>**: Draws a line between two points.
* **<path>**: Defines a complex shape using a series of commands.
* **<polygon>**: Draws a multi-pointed shape (e.g., a triangle or a pentagon).
* **<text>**: Adds text to the SVG image.
* **<g>**: Groups multiple elements for easier manipulation.

**Use Cases:**

* Logos, icons, and illustrations that need to scale across different screen sizes.
* Diagrams, infographics, and flowcharts.
* Interactive maps or visualizations.
* Animations (via CSS or JavaScript).