* HTML5 introduced several new tags to provide better semantic meaning and structure to web documents. Some of the notable new tags introduced in HTML5 are:
  + 1. `<header>`: Defines a header section for a document or a section.
  + 2. `<footer>`: Defines a footer section for a document or a section.
  + 3. `<nav>`: Defines a navigation section for navigation links.
  + 4. `<article>`: Defines an independent piece of content that could stand alone.
  + 5. `<section>`: Defines a section in a document.
  + 6. `<aside>`: Defines content aside from the content it is placed in (e.g., sidebars).
  + 7. `<main>`: Defines the main content of a document.
  + 8. `<figure>`: Defines self-contained content, such as images, diagrams, photos, code listings, etc.
  + 9. `<figcaption>`: Defines a caption for a `<figure>` element.
  + 10. `<time>`: Represents a specific period in time or a range of time.
  + 11. `<mark>`: Highlights text.
  + 12. `<progress>`: Represents the progress of a task.
  + 13. `<meter>`: Represents a scalar measurement within a known range.
  + 14. `<details>`: Defines additional details that the user can view or hide.
  + 15. `<summary>`: Defines a visible heading for a `<details>` element.
  + 16. `<datalist>`: Specifies a list of pre-defined options for input controls.
  + 17. `<dialog>`: Defines a dialog box or window.
  + 18. `<output>`: Represents the result of a calculation or user action.
  + 19. `<canvas>`: Used to draw graphics, via scripting (usually JavaScript).
  + 20. `<audio>`: Embeds sound content into documents.
  + 21. `<video>`: Embeds video content into documents.
  + 22. `<source>`: Specifies multiple media resources for `<audio>` or `<video>` elements.
  + 23. `<track>`: Specifies text tracks for `<audio>` or `<video>` elements.
  + These tags enhance the structure and semantics of HTML documents, making them more accessible and easier to understand for both humans and machines.
* To embed audio and video in a webpage, you can use HTML5's `<audio>` and `<video>` elements. Here's how you can do it:
  + 1. \*\*Embedding Audio:\*\*
  + ```html
  + <audio controls>
  + <source src="audio\_file.mp3" type="audio/mpeg">
  + Your browser does not support the audio element.
  + </audio>
  + ```
  + Explanation:
  + - The `<audio>` element is used to embed audio content in a webpage.
  + - The `controls` attribute adds basic playback controls (play, pause, volume) to the audio player.
  + - Inside the `<audio>` element, you can include one or more `<source>` elements to specify different audio file formats. The browser will choose the first format it supports.
  + - The text "Your browser does not support the audio element." will be displayed if the browser does not support the `<audio>` element.
  + 2. \*\*Embedding Video:\*\*
  + ```html
  + <video controls width="640" height="360">
  + <source src="video\_file.mp4" type="video/mp4">
  + Your browser does not support the video element.
  + </video>
  + ```
  + Explanation:
  + - The `<video>` element is used to embed video content in a webpage.
  + - The `controls` attribute adds basic playback controls (play, pause, volume) to the video player.
  + - You can specify the dimensions of the video using the `width` and `height` attributes.
  + - Inside the `<video>` element, you can include one or more `<source>` elements to specify different video file formats. The browser will choose the first format it supports.
  + - The text "Your browser does not support the video element." will be displayed if the browser does not support the `<video>` element.
  + Make sure to replace `"audio\_file.mp3"` and `"video\_file.mp4"` with the paths to your audio and video files, respectively. Additionally, provide alternative formats (e.g., OGG for audio, WebM for video) inside `<source>` elements to ensure compatibility with different browsers.
* In HTML5, semantic elements are used to give meaning to the content they enclose, making it more understandable for both the browser and developers. Semantic elements play a crucial role in improving accessibility, search engine optimization (SEO), and the overall structure of the webpage. Here are some common semantic elements introduced in HTML5:
  + 1. `<header>`: Represents introductory content or a group of introductory content. Typically contains navigation links, headings, logos, etc.
  + 2. `<nav>`: Defines a section of navigation links.
  + 3. `<main>`: Represents the main content of the document. Should not include navigational elements.
  + 4. `<section>`: Defines a thematic grouping of content, typically with a heading. It's a generic container for content that doesn't have a more specific semantic element available.
  + 5. `<article>`: Represents a self-contained piece of content, such as a blog post, forum post, newspaper article, etc.
  + 6. `<aside>`: Represents content that is tangentially related to the content around it, such as sidebars, pull quotes, etc.
  + 7. `<footer>`: Represents a footer for the nearest sectioning content or sectioning root element. Typically contains metadata, copyright information, contact details, etc.
  + 8. `<figure>`: Represents any content that is referenced from the main content, such as images, illustrations, diagrams, code snippets, etc.
  + 9. `<figcaption>`: Represents a caption or legend for the content inside a `<figure>` element.
  + 10. `<details>` and `<summary>`: Used to create a disclosure widget (often a collapsible section) with a summary or title.
  + Using these semantic elements appropriately helps to create a clear and meaningful structure for your HTML documents, improving accessibility and making your content more understandable to search engines and other parsers.
* In HTML5, both the `<canvas>` and `<svg>` tags are used for creating graphics, but they have different approaches and purposes:
  + 1. \*\*Canvas (`<canvas>`):\*\*
  + - `<canvas>` is an HTML element that provides a drawing space through JavaScript.
  + - It is used for dynamic, scripted rendering of graphics, such as animations, games, data visualizations, etc.
  + - The content of a canvas is rendered using JavaScript code that accesses a drawing context, usually 2D or WebGL, to draw shapes, images, text, etc.
  + - Content drawn on a canvas is not part of the DOM, making it less accessible to assistive technologies and less SEO-friendly compared to SVG.
  + - Examples of canvas use cases include interactive charts, games, photo editing applications, and animations.
  + Example:
  + ```html
  + <canvas id="myCanvas" width="400" height="200"></canvas>
  + ```
  + 2. \*\*SVG (`<svg>`):\*\*
  + - `<svg>` stands for Scalable Vector Graphics.
  + - It is an XML-based markup language for describing vector graphics, such as shapes, paths, text, gradients, etc.
  + - SVG elements are part of the DOM and can be styled, animated, and interacted with using CSS and JavaScript.
  + - SVG graphics are resolution-independent and can be scaled without losing quality, making them suitable for responsive designs.
  + - SVG is more accessible to assistive technologies and offers better SEO because its content is directly readable by search engines.
  + - Common use cases for SVG include logos, icons, illustrations, diagrams, maps, and scalable graphics for the web.
  + Example:
  + ```html
  + <svg width="400" height="200">
  + <circle cx="100" cy="100" r="50" fill="red" />
  + </svg>
  + ```
  + In summary, use `<canvas>` for dynamic, scripted graphics rendering and `<svg>` for scalable vector graphics that can be styled, animated, and interacted with directly within the DOM. The choice between the two depends on the specific requirements and characteristics of your project.