VS Code Extension For HTML

---

1. **Live Server**: This extension provides a live preview of your web pages in real-time as you edit them. It's especially useful for front-end development, allowing you to see changes instantly in the browser.
2. **HTML End Tag Labels**: This extension adds labels to closing HTML tags. It's particularly helpful in large HTML files to quickly identify which closing tag corresponds to which opening tag, improving code readability.
3. **Auto Close Tag**: Automatically adds HTML/XML close tags as soon as you type the opening tag. It speeds up your HTML coding by reducing the amount of typing required.
4. **Highlight Matching Tag**: Whenever you select an HTML tag, this extension highlights the matching opening or closing tag. This feature is useful for quickly navigating and understanding the structure of your HTML code.
5. **Auto Rename Tag**: When you rename one HTML/XML tag, this extension automatically renames the matching paired tag. It's a great time-saver and helps prevent errors in tag renaming.
6. **Code Spell Checker**: This extension helps to catch common spelling errors in your code. It supports multiple languages and programming languages, making it an essential tool for maintaining professional and error-free code.
7. **Indent Rainbow**: This extension colorizes the indentation in front of your text, alternating four different colors on each step. It's visually appealing and makes it easier to distinguish different levels of indentation at a glance.
8. **Prettier - Code Formatter**: Prettier is a popular code formatter that supports many languages and integrates with most editors. It formats your code consistently, following a set of rules, which improves readability and reduces the chance of syntax errors.
9. **Axe Accessibility Linter**: This extension is focused on web accessibility, providing automated testing of HTML for accessibility issues. It's a vital tool for ensuring that web content is accessible to all users, including those with disabilities.

VS Code Emmet

- Visual Studio Code (VS Code) Emmet is a set of tools and functionalities integrated into the VS Code editor, designed to improve HTML and CSS workflow efficiency.
- Emmet uses a special syntax that allows you to quickly generate HTML and CSS code with minimal typing. It's particularly useful for web developers and front-end designers.

Common Abbreviations

- `!` or `html:5` : Basic HTML5 template.
- `a` : Creates an `<a href=""></a>` tag.
- `p` : Generates a `<p></p>` paragraph tag.
- `img` : Produces an `<img src="" alt="">` tag.
- `ul>li` : Creates a list, can be combined with `*` for multiple items.
- `table>tr>td` : Generates a table structure.
- `input:checkbox` : Creates a checkbox input.
- `form:post` : Generates a form with the method set to POST.
- `br` : Inserts a line break.

ID and Class Abbreviations

- `#header` : Generates `<div id="header"></div>` .
- `.container` : Produces `<div class="container"></div>` .

Attribute Abbreviations

`[attr=value]` : Any tag can be appended with attributes like `a[href="http://"]` .

Web Page Structure

---

1. **DOCTYPE Declaration**:
   - `<!DOCTYPE html>`: This declaration defines the document type and version of HTML. For HTML5, use `<!DOCTYPE html>`.
2. **HTML Element**:
   - `<html>`: This is the root element of an HTML page.
3. **Head Section**:
   - `<head>`: Contains meta-information about the document, like character encoding, title, linked CSS files, and scripts.
     - `<meta charset="UTF-8">`: Specifies the character encoding for the HTML document.
     - `<title>`: Title of the webpage, displayed in the browser's title bar or tab.
     - `<link rel="stylesheet" href="style.css">`: Link to an external CSS file.
     - `<script src="script.js"></script>`: Link to an external JavaScript file.
4. **Body Section**:
   - `<body>`: Contains the contents of the HTML document. This is where you place the content that will be visible to users.
     - **Headings**: `<h1>` to `<h6>` tags for headings (h1 for the main heading).
     - **Paragraphs**: `<p>` tag for paragraphs.
     - **Links**: `<a href="url">` to create a hyperlink.
     - **Images**: `<img src="image.jpg" alt="description">` for embedding images.
     - **Lists**:
       - Ordered List: `<ol>` with `<li>` items.
       - Unordered List: `<ul>` with `<li>` items.
     - **Tables**: `<table>`, with `<tr>` for rows, and `<td>` for cells.
     - **Forms**: `<form>` for user input.
       - Input Elements: `<input>`, `<textarea>`, `<button>`, etc.
       -
5. **Comments**:
   - `<!-- Comment -->`: Comments are not displayed in the browser, but they can help document your HTML source code.

Here's a simple example of an HTML page:

Meta Tags

Basic Meta Tags

Title Tag: Defines the page title shown in browser tabs and search results.

```
<title>Page Title</title>
```

Description Meta Tag: A brief summary of the page's content.

```
<meta name="description" content="Short description of the page.">
```

SEO Meta Tags

Keywords Meta Tag: Lists keywords relevant to the page's content.

```
<meta name="keywords" content="keyword1, keyword2">
```

Robots Meta Tag: Directives for search engine crawling and indexing.

```
<meta name="robots" content="index, follow">
```

HTTP-Equiv Meta Tags

Content-Type: Specifies the character encoding for the HTML document.

```
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
```

Default-Style: Defines the default stylesheet.

```
<meta http-equiv="default-style" content="style.css">
```

Refresh: Sets an automatic page refresh interval.

```
<meta http-equiv="refresh" content="30">
```

Open Graph Meta Tags (For Social Media)

OG Title: The title for social media sharing.

```
<meta property="og:title" content="Social Media Title">
```

OG Type: The type of content (e.g., website, article).

```
<meta property="og:type" content="article">
```

OG Image: Image URL for social media.

```
<meta property="og:image" content="http://example.com/image.jpg">
```

Twitter Card Meta Tags

Twitter Card: The Twitter card type.

```
<meta name="twitter:card" content="summary">
```

Twitter Title/Description: Title and description for Twitter.

```
<meta name="twitter:title" content="Twitter Title">
```

Mobile Device Meta Tags

Viewport: Controls the page's viewport settings.

```
<meta name="viewport" content="width=device-width, initial-scale=1.0">
```

Security Meta Tags

Content-Security-Policy: Security policy for resource loading.

```
<meta http-equiv="Content-Security-Policy" content="default-src 'self'">
```

Web Application Meta Tags

1. **Application-Name**: The name of the web application.

```
<meta name="application-name" content="My Web App">
```

2. **Theme-Color**: Suggests a browser theme color.

```
<meta name="theme-color" content="#ffffff">
```

Verification Meta Tags

Google Search Console Verification: Verifies ownership for Google Search Console.

```
<meta name="google-site-verification" content="verification_token">
```

Authorship Meta Tags

Author, Publisher: Defines the author or publisher of the page.

```
<meta name="author" content="Author Name">
```

Cache Control Meta Tags

Pragma, Cache-Control, Expires: Controls browser and server caching.

```
<meta http-equiv="cache-control" content="no-cache">
```

Manifest

Structure of `manifest.json`

1. `name`: The full name of the application.
2. `short_name`: A shorter name for the application, used where space is limited.
3. `start_url`: The entry point of the application when launched. It's relative to the location of the manifest file.
4. `display`: Defines the preferred display mode, such as `fullscreen`, `standalone`, `minimal-ui`, or `browser`.
5. `background_color`: Specifies a background color for the splash screen when the app is launched.
6. `description`: A brief description of the application.
7. `lang`: The primary language of the app, specified using a language tag (`en-US`, `fr`, etc.).
8. `dir`: The text direction of the app, like `ltr` (left-to-right) or `rtl` (right-to-left).
9. `orientation`: Specifies the default orientation of the app, such as `portrait` or `landscape`.
10. `theme_color`: Defines the default theme color for the application.
11. `icons`: An array of image objects representing the app icon in different sizes. Each object typically includes `src`, `sizes`, and `type` properties.
12. `scope`: Specifies the set of URLs that the browser considers to be within your app, and navigations to those URLs stay within the app.
13. `related_applications`: An array of objects specifying native apps that are related to the web app, potentially with a `platform` and a `url` or `id` depending on the platform.

```
{
  "name": "Example App",
  "short_name": "App",
  "start_url": "/",
  "display": "standalone",
  "background_color": "#ffffff",
  "description": "An example Progressive Web App",
  "lang": "en-US",
  "dir": "ltr",
  "orientation": "portrait",
  "theme_color": "#000000",
  "icons": [
    {
      "src": "icon/lowres.webp",
      "sizes": "48x48",
      "type": "image/webp"
    },
    {
      "src": "icon/hd_hi.ico",
      "sizes": "72x72 96x96 128x128 256x256",
      "type": "image/x-icon"
    }
  ],
  "scope": "/",
  "related_applications": [
    {
      "platform": "play",
      "url": "https://play.google.com/store/apps/details?id=com.example.app"
    },
```

```
    {
      "platform": "itunes",
      "url": "https://itunes.apple.com/app/example-app/id123456789"
    }
  ]
}
```

Structural Tags

- `<!DOCTYPE html>`: Declares the document type
- `<html>`: The root element of an HTML page.
- `<head>`: Contains meta-information about the document
- `<body>`: Contains the contents of the HTML document.
- `<header>`: Represents introductory content, typically containing one or more heading elements (`<h1>` to `<h6>`) and navigation elements (`<nav>`).
- `<nav>`: Defines a set of navigation links.
- `<main>`: Specifies the main content of the document.
- `<section>`: Defines a section in a document, such as a chapter, header, footer, or any other sections of the document.
- `<article>`: Represents a self-contained composition in a document, page, application, or site.
- `<aside>`: Used for content that is tangentially related to the content around the `<aside>` element.
- `<footer>`: Represents the footer of a document or section

```
<!DOCTYPE html>
<html>
<head>
    <title>Sample Web Page</title>
</head>
<body>
    <header>
        <h1>Welcome to My Website</h1>
        <nav>
            <ul>
                <li><a href="#home">Home</a></li>
                <li><a href="#about">About</a></li>
                <li><a href="#contact">Contact</a></li>
            </ul>
        </nav>
    </header>

    <main>
        <section id="home">
            <h2>Home</h2>
            <p>This is the Home section of the page.</p>
        </section>

        <article id="about">
            <h2>About</h2>
            <p>This section contains information about the website or its creator.</p>
        </article>

        <aside>
            <h2>News</h2>
            <p>Latest news related to the website or topic.</p>
        </aside>
    </main>

    <footer>
```

```
        <p>Contact us: email@example.com</p>
    </footer>
</body>
</html>
```

Text Formatting Tags

- `<h1>` to `<h6>`: These tags are used for creating headings. `<h1>` represents the most important heading, while `<h6>` represents the least important.
- `<p>`: Defines a paragraph.
- `<strong>`: Makes text bold.
- `<em>`: Italicises text, indicating emphasis.
- `<br>`: Inserts a line break.
- `<small>`: Decreases the size of the text.
- `<mark>`: Highlights text.
- `<sub>`: Creates subscript text.
- `<sup>`: Creates superscript text.
- `<ins>`: Represents inserted text.
- `<del>`: Represents deleted text.
- `<code>`: Displays its contents styled in a fashion intended to indicate that the text is a short fragment of computer code.
- `<samp>`: Used to enclose inline text which represents sample (or quoted) output from a computer program.
- `<q>`: Defines a short inline quotation.
- `<blockquote>`: Represents a section that is quoted from another source, and the `cite` attribute can include the URL of the source.
- `<address>`: Indicates that the enclosed HTML provides contact information.
- `<hr>`: Inserts a horizontal rule, which defines a thematic break in an HTML page.

```
<!DOCTYPE html>
<html>
<head>
    <title>Text Formatting Example</title>
</head>
<body>
    <h1>Heading 1</h1>
    <h2>Heading 2</h2>
    <p>This is a paragraph <strong>with bold text</strong> and <em>italic text</em>.</p>
    <p>This is another paragraph with a <br> line break.</p>
    <p>Here's an example of <small>smaller text</small> and <mark>highlighted text</mark>.</p>
    <p>You can also use <sub>subscript</sub> and <sup>superscript</sup>.</p>
    <p>This is a paragraph with <ins>inserted</ins> text and <del>deleted</del> text.</p>
    <p>Here's some <code>computer code</code> and a <samp>sample output</samp> from a computer
program.</p>
    <p>Quotation: <q>Do not go gentle into that good night.</q></p>
    <blockquote cite="https://www.poetryfoundation.org/poems/48979/do-not-go-gentle-into-that-good-
night">
        Do not go gentle into that good night,
        Old age should burn and rave at close of day;
        Rage, rage against the dying of the light.
    </blockquote>
    <p>An address example:</p>
    <address>
        Written by John Doe.<br>
        Visit us at:<br>
```

```
        Example.com<br>
        Box 564, Disneyland<br>
        USA
    </address>
    <hr>
    <p>This is a paragraph after a horizontal rule.</p>
</body>
</html>
```

List Tags

- `<ul>` is used for an unordered list, where list items (`<li>`) are typically displayed with bullets.
- `<ol>` is used for an ordered list, where list items are displayed with numbers or letters.
- `<dl>` creates a description list, which is a list of terms and their corresponding descriptions. Inside the `<dl>`, `<dt>` is used for the term (name), and `<dd>` is used for the term's description.

```
<!DOCTYPE html>
<html>
<head>
    <title>List Tags Example</title>
</head>
<body>
    <h2>Unordered List</h2>
    <ul>
        <li>Apple</li>
        <li>Banana</li>
        <li>Cherry</li>
    </ul>

    <h2>Ordered List</h2>
    <ol>
        <li>First item</li>
        <li>Second item</li>
        <li>Third item</li>
    </ol>

    <h2>Description List</h2>
    <dl>
        <dt>Coffee</dt>
        <dd>Black hot drink</dd>
        <dt>Milk</dt>
        <dd>White cold drink</dd>
    </dl>
</body>
</html>
```

Link Tags

- **External links**: These are links that lead to other websites. The `target="_blank"` attribute opens the link in a new tab or window.
- **Internal links**: These links lead to different sections within the same web page. They use the `#` symbol followed by the `id` of the destination element. In this example, clicking on "Go to Section 1" will navigate the user to the part of the page where `<h2 id="section1">` is defined.
- **Email link**: The `mailto:` protocol in the `href` attribute creates a link that opens the user's default email client with the address specified in the link.
- **Telephone link**: The `tel:` protocol creates a link that, when clicked on a device capable of making phone calls, will start a call to the specified number.

Image and Multimedia Tags

- `<img>`: This tag is used for embedding an image in the HTML page. The `src` attribute specifies the path to the image file, and the `alt` attribute provides alternative text for the image if it cannot be displayed. The `width` attribute specifies the width of the image.
- `<video>`: The `<video>` tag is used to embed video content. It can contain one or more `<source>` elements to specify multiple video sources for different formats. The `controls` attribute adds video controls, like play, pause, and volume.
- `<audio>`: Similar to `<video>`, the `<audio>` tag is used for embedding audio content and can contain multiple `<source>` elements. The `controls` attribute adds audio controls.
- `<iframe>`: This tag is used to embed another HTML document within the current page. In this example, it's used to embed a YouTube video. The `src` attribute specifies the URL of the page to embed.

```
<!DOCTYPE html>
<html>
<head>
    <title>Image and Multimedia Tags Example</title>
</head>
<body>
    <h2>Image Tag</h2>
    <img src="example.jpg" alt="Example Image" width="300">

    <h2>Video Tag</h2>
    <video width="320" height="240" controls>
        <source src="movie.mp4" type="video/mp4">
        <source src="movie.ogg" type="video/ogg">
        Your browser does not support the video tag.
    </video>

    <h2>Audio Tag</h2>
    <audio controls>
        <source src="audio.mp3" type="audio/mpeg">
        <source src="audio.ogg" type="audio/ogg">
        Your browser does not support the audio tag.
    </audio>

    <h2>Embedded YouTube Video</h2>
    <iframe width="560" height="315" src="https://www.youtube.com/embed/dQw4w9WgXcQ"
frameborder="0" allow="accelerometer; autoplay; encrypted-media; gyroscope; picture-in-picture"
allowfullscreen></iframe>
</body>
</html>
```

Form Tags

- `<form>`: This tag is used to create the form. The `action` attribute specifies where the form data should be sent when it's submitted, and the `method` attribute specifies the HTTP method (GET or POST).
- `<label>`: Defines a label for an `<input>` element. The `for` attribute should be the same as the `id` of the input it's labeling.
- `<input>`: This tag is a versatile form element. Depending on the `type` attribute, it can be a text field, password field, radio button, checkbox, etc. In this example, types used include text, email, date, radio, checkbox, and submit.
- `<textarea>`: Allows for multi-line text input. It's useful for longer, free-form text like comments or descriptions.
- `<input type="submit">`: Defines a button for submitting the form.

```
<!DOCTYPE html>
<html>
<head>
    <title>Form Tags Example</title>
</head>
<body>
    <h2>Registration Form</h2>
    <form action="/submit-form" method="post">
        <label for="fname">First Name:</label><br>
        <input type="text" id="fname" name="fname"><br>

        <label for="lname">Last Name:</label><br>
        <input type="text" id="lname" name="lname"><br>

        <label for="email">Email:</label><br>
        <input type="email" id="email" name="email"><br>

        <label for="birthday">Birthday:</label><br>
        <input type="date" id="birthday" name="birthday"><br>

        <p>Gender:</p>
        <input type="radio" id="male" name="gender" value="male">
        <label for="male">Male</label><br>
        <input type="radio" id="female" name="gender" value="female">
        <label for="female">Female</label><br>
        <input type="radio" id="other" name="gender" value="other">
        <label for="other">Other</label><br>

        <label for="bio">Biography:</label><br>
        <textarea id="bio" name="bio" rows="4" cols="50"></textarea><br>

        <input type="checkbox" id="subscribe" name="subscribe" value="newsletter">
        <label for="subscribe">Subscribe to newsletter</label><br>

        <input type="submit" value="Submit">
    </form>
</body>
```

```
</html>
```

Scripting Tags

- The `<script>` tag in the `<head>` section contains JavaScript code directly within it. This script declares a function named `displayDate()` that writes the current date and time to an HTML element with the ID `date`.
- A `<button>` element in the body uses the `onclick` attribute to call the `displayDate()` function when clicked.
- Another `<script>` tag at the end of the body is used to link an external JavaScript file (`myScript.js`). This file would contain JavaScript code that the page can use. The `src` attribute specifies the path to the external script file.
- The `<noscript>` tag provides an alternative content for users who have JavaScript disabled in their browser. This is important for accessibility and ensuring that essential information or functionality is still available without JavaScript.

```
<!DOCTYPE html>
<html>
<head>
    <title>Scripting Tags Example</title>
    <script>
        // JavaScript code can be written directly within this script tag
        function displayDate() {
            document.getElementById("date").innerHTML = Date();
        }
    </script>
</head>
<body>
    <h1>Welcome to My Web Page</h1>
    <button type="button" onclick="displayDate()">Click me to display Date and Time.</button>
    <p id="date"></p>

    <!-- External JavaScript file can be linked here -->
    <script src="myScript.js"></script>

    <!-- The noscript tag is for users who have JavaScript disabled -->
    <noscript>Your browser does not support JavaScript or it is disabled.</noscript>
</body>
</html>
```

Interactive Element Tags

- `<button>`: A clickable button is created. The `onclick` attribute is used here to display an alert dialog box when the button is clicked.
- `<details>` and `<summary>`: These tags create a collapsible section. `<details>` is the container, and `<summary>` provides a clickable heading. Content placed within `<details>` but outside `<summary>` is hidden or shown when the user clicks on the `<summary>`.
- `<dialog>`: Represents a dialog box or other interactive component. JavaScript is used to show and hide the dialog.
- `<progress>`: Displays a progress bar. The `value` attribute specifies the current progress.
- `<input type="range">`: Creates a slider for selecting a numerical value within a specified range.
- `<select>` and `<option>`: Create a dropdown list. `<select>` defines the dropdown, and `<option>` elements represent the options within the dropdown.

```
<!DOCTYPE html>
<html>
<head>
    <title>Interactive Elements Example</title>
    <style>
        dialog {
            border: 1px solid #000;
            border-radius: 5px;
            padding: 10px;
        }
    </style>
</head>
<body>
    <h2>Interactive HTML Elements</h2>

    <h3>Button</h3>
    <button onclick="alert('Hello World!')">Click Me!</button>

    <h3>Details and Summary</h3>
    <details>
        <summary>More Information</summary>
        <p>This is additional information that the user can view on demand.</p>
    </details>

    <h3>Dialog</h3>
    <button onclick="document.getElementById('myDialog').showModal();">Open Dialog</button>
    <dialog id="myDialog">
        <p>This is a dialog window. You can include any content here.</p>
        <button onclick="document.getElementById('myDialog').close();">Close</button>
    </dialog>

    <h3>Progress Bar</h3>
    <label for="file">Downloading progress:</label>
    <progress id="file" max="100" value="70">70%</progress>

    <h3>Range Slider</h3>
    <label for="volume">Volume Control:</label>
    <input type="range" id="volume" name="volume" min="0" max="11">
```

```
    <h3>Select Dropdown</h3>
    <label for="cars">Choose a car:</label>
    <select id="cars">
        <option value="volvo">Volvo</option>
        <option value="saab">Saab</option>
        <option value="mercedes">Mercedes</option>
        <option value="audi">Audi</option>
    </select>
</body>
</html>
```

Special Purpose Tags

- `<iframe>`: This tag embeds an external webpage within the current HTML document. Here, it's used to embed a Google Maps location.
- `<embed>`: Used for embedding external content like a video. In this example, a `.webm` video is embedded.
- `<object>`: A versatile tag for embedding various types of multimedia and applications. Here, it's used to embed a PDF file. If the browser cannot display the PDF, a download link is provided.
- `<svg>`: Stands for Scalable Vector Graphics. It's used to define vector-based graphics in XML format. This example creates an SVG circle.

```
<!DOCTYPE html>
<html>
<head>
    <title>Special Purpose Tags Example</title>
</head>
<body>
    <h2>Special Purpose HTML Elements</h2>

    <h3>Embedding an iFrame</h3>
    <p>An embedded Google Maps location:</p>
    <iframe
        width="600"
        height="450"
        style="border:0"
        loading="lazy"
        allowfullscreen
        src="https://www.google.com/maps/embed/v1/place?q=place_id:ChIJN1t_tDeuEmsRUsoyG83frY4">
    </iframe>

    <h3>Embedding a Video</h3>
    <embed
        type="video/webm"
        src="movie.webm"
        width="300"
        height="200">
    </embed>

    <h3>Object Tag for PDF</h3>
    <object
        data="file.pdf"
        type="application/pdf"
        width="300"
        height="200">
        <a href="file.pdf">Download PDF</a>
    </object>

    <h3>SVG Graphics</h3>
    <svg width="100" height="100">
        <circle cx="50" cy="50" r="40" stroke="green" stroke-width="4" fill="yellow" />
```

```
    </svg>
</body>
</html>
```