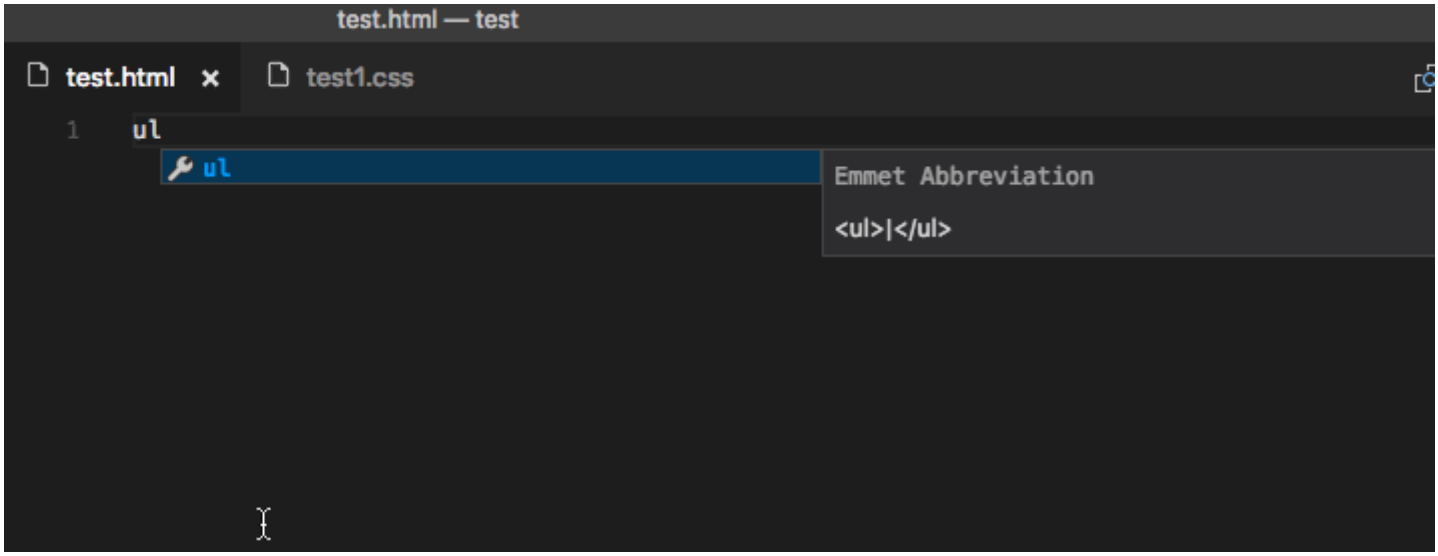
(https://vscode.dev/github/microsoft/vscode-docs/blob/main/docs/editor/emmet.md)

Emmet in Visual Studio Code

Support for Emmet (https://emmet.io/) snippets and expansion is built right into Visual Studio Code, **no extension required**. Emmet 2.0 (https://code.visualstudio.com/blogs/2017/08/07/emmet-2.0) has support for the majority of the Emmet Actions (https://docs.emmet.io/actions/) including expanding Emmet abbreviations and snippets (https://docs.emmet.io/cheat-sheet/).

How to expand Emmet abbreviations and snippets

Emmet abbreviation and snippet expansions are enabled by default in `html`, `haml`, `pug`, `slim`, `jsx`, `xml`, `xsl`, `css`, `scss`, `sass`, `less` and `stylus` files, as well as any language that inherits from any of the above like `handlebars` and `php`.



When you start typing an Emmet abbreviation, you will see the abbreviation displayed in the suggestion list. If you have the suggestion documentation fly-out open, you will see a preview of the expansion as you type. If you are in a stylesheet file, the expanded abbreviation shows up in the suggestion list sorted among the other CSS suggestions.

Using Tab for Emmet expansions

If you want to use the `Tab` key for expanding the Emmet abbreviations, add the following setting:

```
"emmet.triggerExpansionOnTab": true
```

This setting allows using the `Tab` key for indentation when text is not an Emmet abbreviation.

Emmet when quickSuggestions are disabled

If you have disabled the `editor.quickSuggestions` setting (/docs/getstarted/settings), you won't see suggestions as you type. You can still trigger suggestions manually by pressing `Ctrl+Space` and see the preview.

Disable Emmet in suggestions

If you don't want to see Emmet abbreviations in suggestions at all, then use the following setting:

```
"emmet.showExpandedAbbreviation": "never"
```

You can still use the command **Emmet: Expand Abbreviation** to expand your abbreviations. You can also bind any keyboard shortcut to the command id `editor.emmet.action.expandAbbreviation` as well.

Emmet suggestion ordering

To ensure Emmet suggestions are always on top in the suggestion list, add the following settings:

```
"emmet.showSuggestionsAsSnippets": true,
"editor.snippetSuggestions": "top"
```

Emmet abbreviations in other file types

To enable the Emmet abbreviation expansion in file types where it is not available by default, use the `emmet.includeLanguages` setting. Make sure to use language identifiers (/docs/languages/identifiers) for both sides of the mapping, with the right side being the language identifier of an Emmet supported language (see the list above).

For example:

```
"emmet.includeLanguages": {
    "javascript": "javascriptreact",
    "razor": "html",
    "plaintext": "pug"
}
```

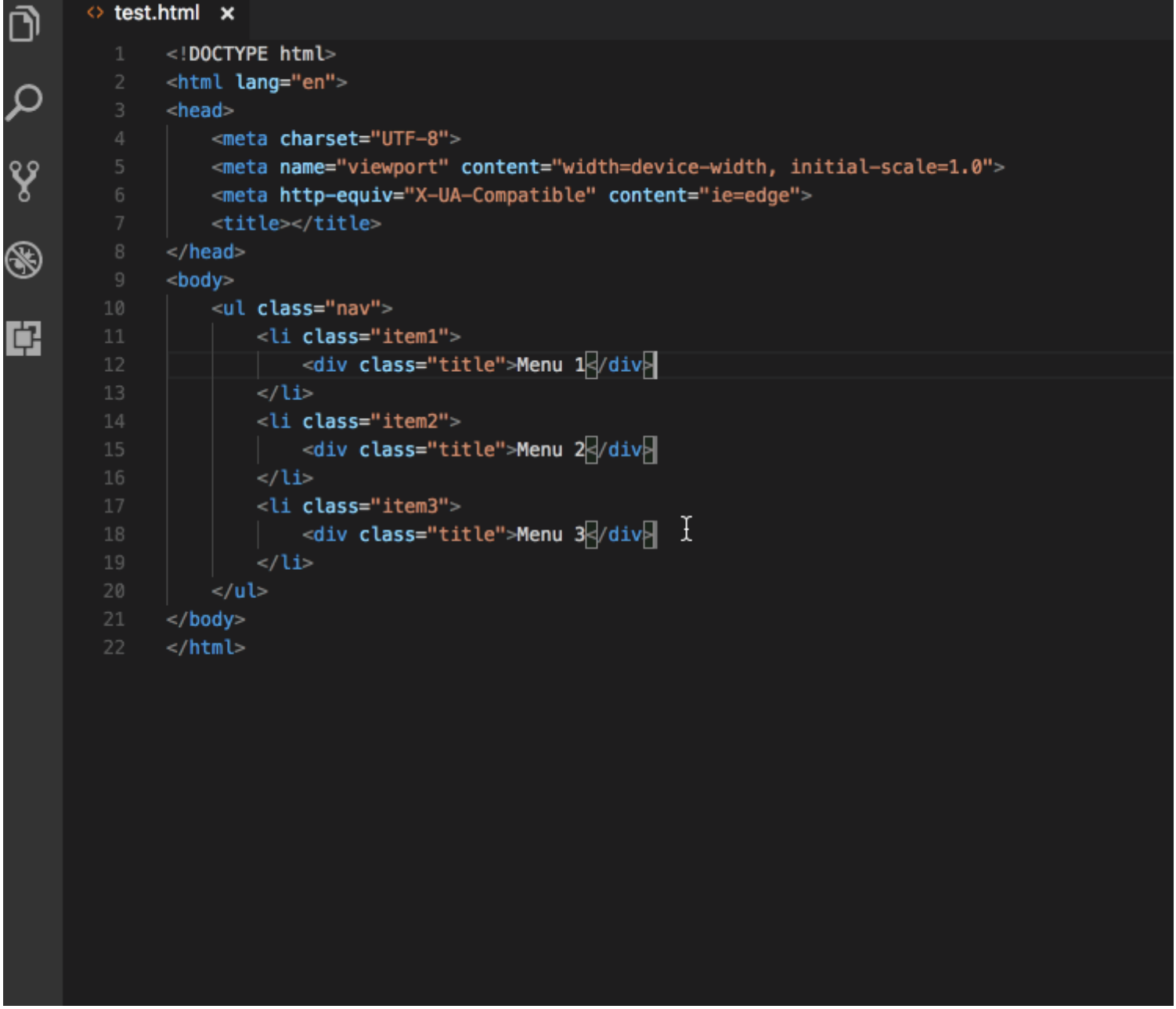
Emmet has no knowledge of these new languages, and so there might be Emmet suggestions showing up in non HTML/CSS contexts. To avoid this, you can use the following setting.

```
"emmet.showExpandedAbbreviation": "inMarkupAndStylesheetFilesOnly"
```

> **Note:** If you used `emmet.syntaxProfiles` previously to map new file types, from VS Code 1.15 onwards you should use the setting `emmet.includeLanguages` instead. `emmet.syntaxProfiles` is meant for customizing the final output (https://docs.emmet.io/customization/syntax-profiles) only.

Emmet with multi-cursors

You can use most of the Emmet actions with multi-cursors as well:



Using filters

Filters are special post-processors that modify the expanded abbreviation before it is output to the editor. There are 2 ways to use filters; either globally through the `emmet.syntaxProfiles` setting or directly in the current abbreviation.

Below is an example of the first approach using the `emmet.syntaxProfiles` setting to apply the `bem` filter for all the abbreviations in HTML files:

```
"emmet.syntaxProfiles": {
    "html": {
        "filters": "bem"
    }
}
```

To provide a filter for just the current abbreviation, append the filter to your abbreviation. For example, `div#page|c` will apply the `comment` filter to the `div#page` abbreviation.

BEM filter (bem)

If you use the Block Element Modifier (http://getbem.com/) (BEM) way of writing HTML, then `bem` filters are very handy for you to use. To learn more about how to use `bem` filters, read BEM filter in Emmet (https://docs.emmet.io/filters/bem/).

You can customize this filter by using the `bem.elementSeparator` and `bem.modifierSeparator` preferences as documented in Emmet Preferences (https://docs.emmet.io/customization/preferences/).

Comment filter (c)

This filter adds comments around important tags. By default, "important tags" are those tags with id and/or class attribute.

For example `div>div#page>p.title+p|c` will be expanded to:

```
<div>
    <div id="page">
        <p class="title"></p>
        <!-- /.title -->
        <p></p>
    </div>
    <!-- /#page -->
</div>
```

You can customize this filter by using the `filter.commentTrigger`, `filter.commentAfter` and `filter.commentBefore` preferences as documented in Emmet Preferences (https://docs.emmet.io/customization/preferences/).

The format for the `filter.commentAfter` preference is different in VS Code Emmet 2.0.

For example, instead of:

```
"emmet.preferences": {
    "filter.commentAfter": "\n<!-- /<%= attr('id', '#') %><%= attr('class', '.') %> -->"
}
```

in VS Code, you would use a simpler:

```
"emmet.preferences": {
    "filter.commentAfter": "\n<!-- /[#ID][.CLASS] -->"
}
```

Trim filter (t)

This filter is applicable only when providing abbreviations for the **Emmet: Wrap with Abbreviation** command. It removes line markers (https://docs.emmet.io/actions/wrap-with-abbreviation/#removing-list-markers) from wrapped lines.

Using custom Emmet snippets

Custom Emmet snippets need to be defined in a json file named `snippets.json`. The `emmet.extensionsPath` setting should have the path to the directory containing this file.

Below is an example for the contents of this `snippets.json` file.

```
{
  "html": {
    "snippets": {
      "ull": "ul>li[id=${1} class=${2}]*2{ Will work with html, pug, haml and slim }",
      "oll": "<ol><li id=${1} class=${2}> Will only work in html </ol>",
      "ran": "{ Wrap plain text in curly braces }"
    }
  },
  "css": {
    "snippets": {
      "cb": "color: black",
      "bsd": "border: 1px solid ${1:red}",
      "ls": "list-style: ${1}"
    }
  }
}
```

Authoring of Custom Snippets in Emmet 2.0 via the `snippets.json` file differs from the old way of doing the same in a few ways:

Topic	Old Emmet	Emmet 2.0
Snippets vs Abbreviations	Supports both in 2 separate properties called `snippets` and `abbreviations`	The 2 have been combined into a single property called snippets. See default HTML snippets (https://github.com/emmetio/snippets/blob/master/html.json) and CSS snippets (https://github.com/emmetio/snippets/blob/master/css.json)
CSS snippet names	Can contain `:`	Do not use `:` when defining snippet names. It is used to separate property name and value when Emmet tries to fuzzy match the given abbreviation to one of the snippets.
CSS snippet values	Can end with `;`	Do not add `;` at end of snippet value. Emmet will add the trailing `;` based on the file type (css/less/scss vs sass/stylus) or the emmet preference set for `css.propertyEnd`, `sass.propertyEnd`, `stylus.propertyEnd`
Cursor location	`${cursor}` or `` ` ``	`` ` `` can be used

HTML Emmet snippets

HTML custom snippets are applicable to all other markup flavors like `haml` or `pug`. When snippet value is an abbreviation and not actual HTML, the appropriate transformations can be applied to get the right output as per the language type.

For example, for an unordered list with a list item, if your snippet value is `ul>li`, you can use the same snippet in `html`, `haml`, `pug` or `slim`, but if your snippet value is `<ul><li></li></ul>`, then it will work only in `html` files.

If you want a snippet for plain text, then surround the text with `{}`.

CSS Emmet snippets

Values for CSS Emmet snippets should be a complete property name and value pair.

CSS custom snippets are applicable to all other stylesheet flavors like `scss`, `less` or `sass`. Therefore, don't include a trailing `;` at the end of the snippet value. Emmet will add it as needed based on whether the language requires it.

Do not use `:` in the snippet name. `:` is used to separate property name and value when Emmet tries to fuzzy match the abbreviation to one of the snippets.

Tab stops and cursors in custom snippets

The syntax for tab stops in custom Emmet snippets follows the Textmate snippets syntax (https://manual.macromates.com/en/snippets).

- Use `${1}`, `${2}` for tab stops and `${1:placeholder}` for tab stops with placeholders.
- Previously, `|` or `${cursor}` was used to denote the cursor location in the custom Emmet snippet. This is no longer supported. Use `${1}` instead.

Emmet configuration

Below are Emmet settings (/docs/getstarted/settings) that you can use to customize your Emmet experience in VS Code.

- `emmet.includeLanguages`

  Use this setting to add mapping between the language of your choice and one of the Emmet supported languages to enable Emmet in the former using the syntax of the latter. Make sure to use language IDs for both sides of the mapping.

  For example:

  ```
  "emmet.includeLanguages": {
      "javascript": "javascriptreact",
      "plaintext": "pug"
  }
  ```

- `emmet.excludeLanguages`

  If there is a language where you do not want to see Emmet expansions, add it in this setting which takes an array of language ID strings.

- `emmet.syntaxProfiles`

  See Emmet Customization of output profile (https://docs.emmet.io/customization/syntax-profiles/#create-your-own-profile) to learn how you can customize the output of your HTML abbreviations.

  For example:

  ```
  "emmet.syntaxProfiles": {
      "html": {
          "attr_quotes": "single"
      },
      "jsx": {
          "self_closing_tag": true
      }
  }
  ```

- `emmet.variables`

  Customize variables used by Emmet snippets.

  For example:

```
"emmet.variables": {
    "lang": "de",
    "charset": "UTF-16"
}
```

- `emmet.showExpandedAbbreviation`

  Controls the Emmet suggestions that show up in the suggestion/completion list.

Setting Value	Description
`never`	Never show Emmet abbreviations in the suggestion list for any language.
`inMarkupAndStylesheetFilesOnly`	Show Emmet suggestions only for languages that are purely markup and stylesheet based ('html', 'pug', 'slim', 'haml', 'xml', 'xsl', 'css', 'scss', 'sass', 'less', 'stylus').
`always`	Show Emmet suggestions in all Emmet supported modes as well as the languages that have a mapping in the `emmet.includeLanguages` setting.

  Note: In the `always` mode, the new Emmet implementation is not context aware. For example, if you are editing a JavaScript React file, you will get Emmet suggestions not only when writing markup but also while writing JavaScript.

- `emmet.showAbbreviationSuggestions`

  Shows possible emmet abbreviations as suggestions. It is `true` by default.

  For example, when you type `li`, you get suggestions for all emmet snippets starting with `li` like `link`, `link:css`, `link:favicon` etc. This is helpful in learning Emmet snippets that you never knew existed unless you knew the Emmet cheatsheet (https://docs.emmet.io/cheat-sheet/) by heart.

  Not applicable in stylesheets or when `emmet.showExpandedAbbreviation` is set to `never`.

- `emmet.extensionsPath`

  Provide the location of the directory that houses the `snippets.json` file which in turn has your custom snippets.

- `emmet.triggerExpansionOnTab`

  Set this to true to enable expanding Emmet abbreviations with `Tab` key. We use this setting to provide the appropriate fallback to provide indentation when there is no abbreviation to expand.

- `emmet.showSuggestionsAsSnippets`

  If set to `true`, then Emmet suggestions will be grouped along with other snippets allowing you to order them as per `editor.snippetSuggestions` setting. Set this to `true` and `editor.snippetSuggestions` to `top`, to ensure that Emmet suggestions always show up on top among other suggestions.

- `emmet.preferences`

  You can use this setting to customize Emmet as documented in Emmet Preferences (https://docs.emmet.io/customization/preferences/). The below customizations are currently supported:

  - `css.propertyEnd`
  - `css.valueSeparator`
  - `sass.propertyEnd`
  - `sass.valueSeparator`

- `stylus.propertyEnd`
- `stylus.valueSeparator`
- `css.unitAliases`
- `css.intUnit`
- `css.floatUnit`
- `bem.elementSeparator`
- `bem.modifierSeparator`
- `filter.commentBefore`
- `filter.commentTrigger`
- `filter.commentAfter`
- `format.noIndentTags`
- `format.forceIndentationForTags`
- `profile.allowCompactBoolean`
- `css.fuzzySearchMinScore`

The format for the `filter.commentAfter` preference is different and simpler in Emmet 2.0.

For example, instead of the older format

```
"emmet.preferences": {
    "filter.commentAfter": "\n<!-- /<%= attr('id', '#') %><%= attr('class', '.') %> -->"
}
```

you would use

```
"emmet.preferences": {
    "filter.commentAfter": "\n<!-- /[#ID][.CLASS] -->"
}
```

If you want support for any of the other preferences as documented in Emmet Preferences (https://docs.emmet.io/customization/preferences/), please log a feature request (https://github.com/microsoft/vscode/issues/new).

Next steps

Emmet is just one of the great web developer features in VS Code. Read on to find out about:

- HTML (/docs/languages/html) - VS Code supports HTML with IntelliSense, closing tags, and formatting.
- CSS (/docs/languages/css) - We offer rich support for CSS, SCSS and Less.

Troubleshooting

Custom tags do not get expanded in the suggestion list

Custom tags when used in an expression like `MyTag>YourTag` or `MyTag.someclass` do show up in the suggestion list. But when these are used on their own like `MyTag`, they do not appear in the suggestion list. This is designed so to avoid noise in the suggestion list as every word is a potential custom tag.

Add the following setting to enable expanding of Emmet abbreviations using tab which will expand custom tags in all cases.

```
"emmet.triggerExpansionOnTab": true
```

My HTML snippets ending with `+` do not work

HTML snippets ending with `+` like `select+` and `ul+` from the Emmet cheatsheet (https://docs.emmet.io/cheat-sheet/) are not supported. This is a known issue in Emmet 2.0 Issue: emmetio/html-matcher#1 (https://github.com/emmetio/html-matcher/issues/1). A workaround is to create your own custom Emmet snippets (/docs/editor/emmet#_using-custom-emmet-snippets) for such scenarios.

Abbreviations are failing to expand

First, check if you're using custom snippets (if there is a `snippets.json` file being picked up by the `emmet.extensionsPath` setting). The format of custom snippets changed in VS Code release 1.53. Instead of using `|` to indicate where the cursor position is, use tokens such as `${1}`, `${2}`, etc. instead. The default CSS snippets file (https://github.com/emmetio/emmet/blob/master/snippets/css.json) from the `emmetio/emmet` repository shows examples of the new cursor position format.

If abbreviations are still failing to expand:

- Check the builtin extensions (/docs/editor/extension-marketplace#_extensions-view-filters) to see if Emmet has been disabled.
- Try restarting the extension host by running the **Developer: Restart Extension Host** (`workbench.action.restartExtensionHost`) command in the Command Palette (/docs/getstarted/userinterface#_command-palette).

Where can I set all the preferences as documented in Emmet preferences (https://docs.emmet.io/customization/preferences/)?

You can set the preferences using the setting `emmet.preferences`. Only a subset of the preferences that are documented in Emmet preferences (https://docs.emmet.io/customization/preferences/) can be customized. Please read the preferences section under Emmet configuration (/docs/editor/emmet#_emmet-configuration).

Any tips and tricks?

Of course!

- In CSS abbreviations, when you use `:`, the left part is used to fuzzy match with the CSS property name and the right part is used to match with CSS property value. Take full advantage of this by using abbreviations like `pos:f`, `trf:rx`, `fw:b`, etc.
- Explore all other Emmet features as documented in Emmet Actions (https://docs.emmet.io/actions/).
- Don't hesitate to create your own custom Emmet snippets (/docs/editor/emmet#_using-custom-emmet-snippets).