Fundamentals of Web Development

Chapter 1 – Getting Started

Chapter Overview

VSC is a lightweight but powerful source code editor which runs on your desktop and is available on all major platforms. But, before you can begin writing your first line of code, you need to learn how to open files and/or projects in VSC, how to use VSC to create an HTML document, how to open an HTML in the browser using the **open in a browser** extension, and what an **.editorconfig** file is.

Learning Objectives

By the end of this chapter, you should be able to:

- Describe the folder structure for the Web Development cohort
- Open a file and/or project in VSC
- Identify what an .editorconfig file is and how it is used in a project
- Use VSC to create an HTML document
- Explore the courses GitHub repository

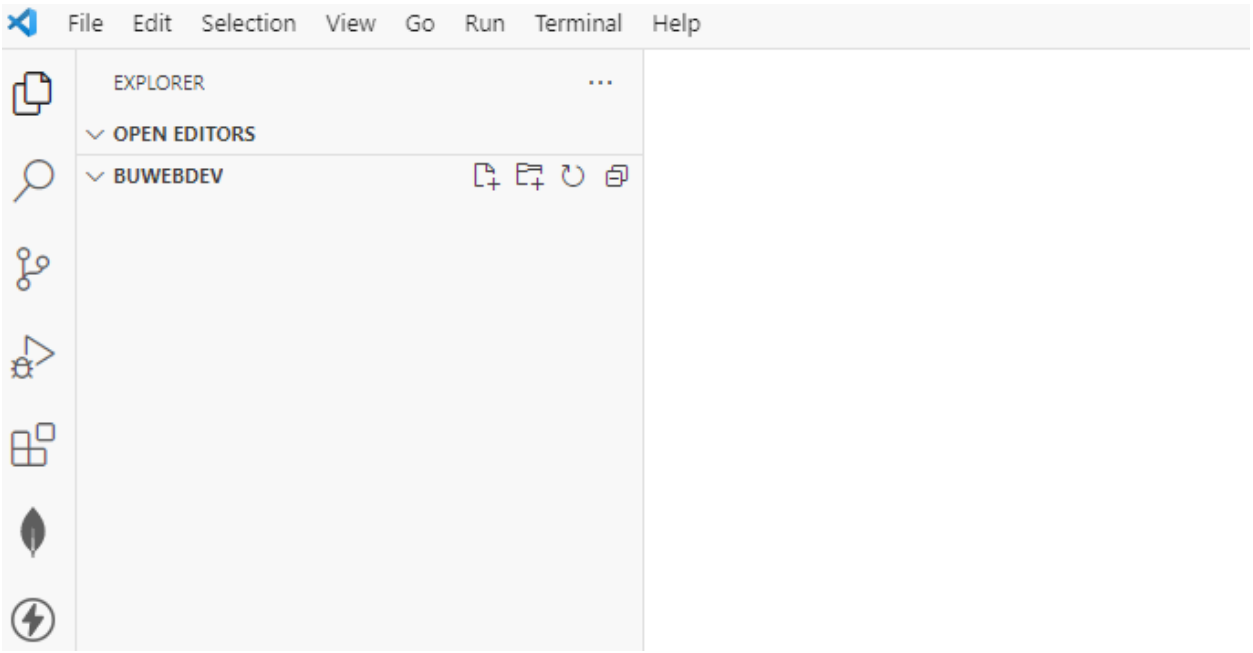
Opening a Project in VSC

In this section we will take a look at how to open a project and/or file in VSC. Before you proceed with this week's assignments, please make sure you have completed the items listed under "Week 0 – Pre-workout."

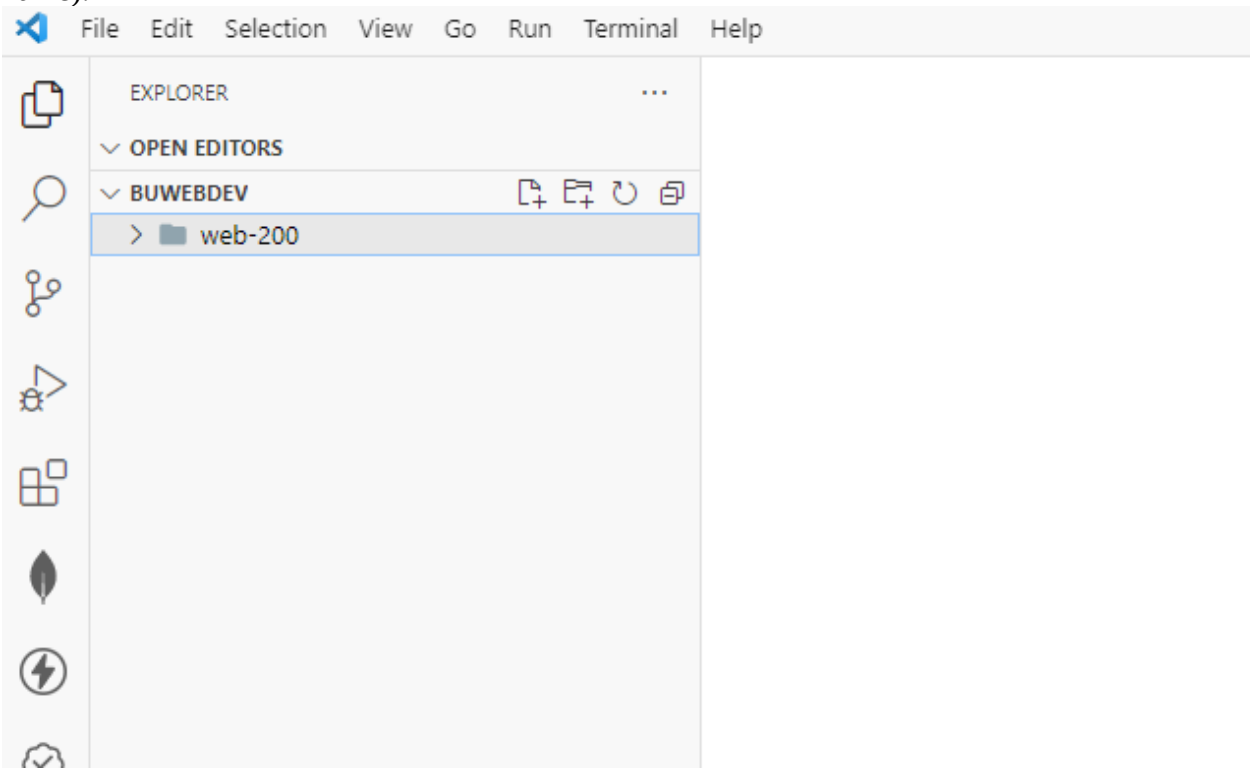
The first thing you will need to do is create a new directory on your computer named **buwebdev**. I recommend placing the **buwebdev** directory in your account directory. For Windows this would be **C:\Users\{Username}** and for macOS this would be **/users/{Username}**.

Open VSC (double-click the VSC icon)

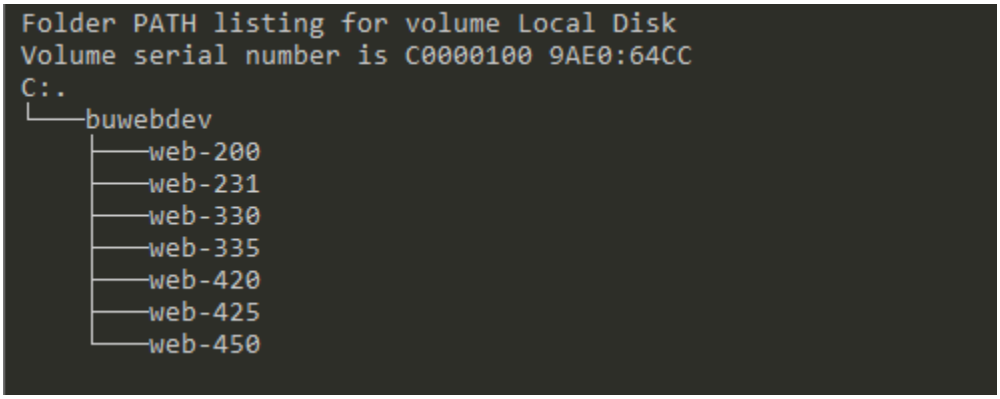
Select File -> Open Folder and navigate to your **buwebdev** directory.



Create a new folder and name it **web-200** (click on the folder icon to the right of the **buwebdev** name).



All programming courses in this cohort will follow this organizational structure. That is, there should be a parent directory named **buwebdev** and child directories for each programming course. For example,



Unless otherwise specified, the work you complete in a programming course will be added to the appropriate directory. For now, all you need is the **buwebdev** and **web-200** directories.

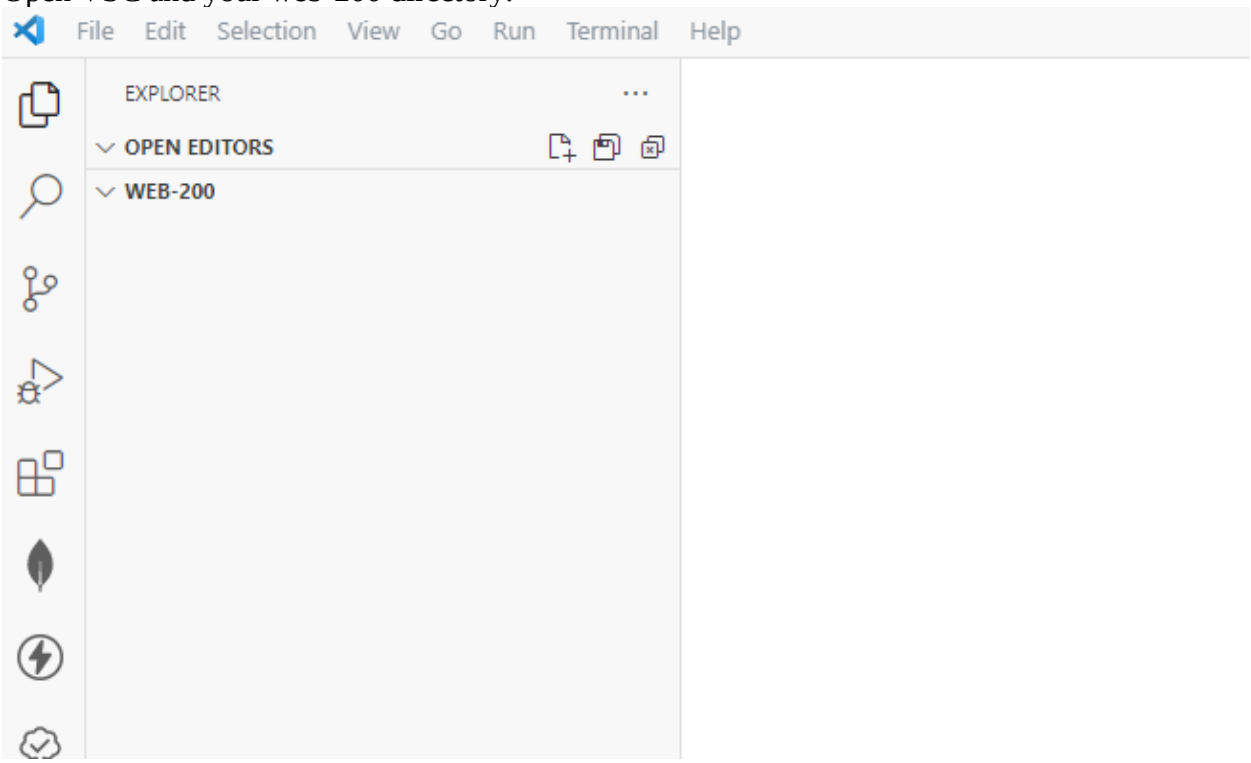
Anytime you want to open a project in VSC, you follow this approach. The most important thing to remember is to always select the directory of the project you want to open. For example, **web-200**. The files and directories you create in this project can all be done from VSC. All you need to do is select the appropriate icons to the right of the root project name (in our example, this would be **buwebdev**).

.editorConfig

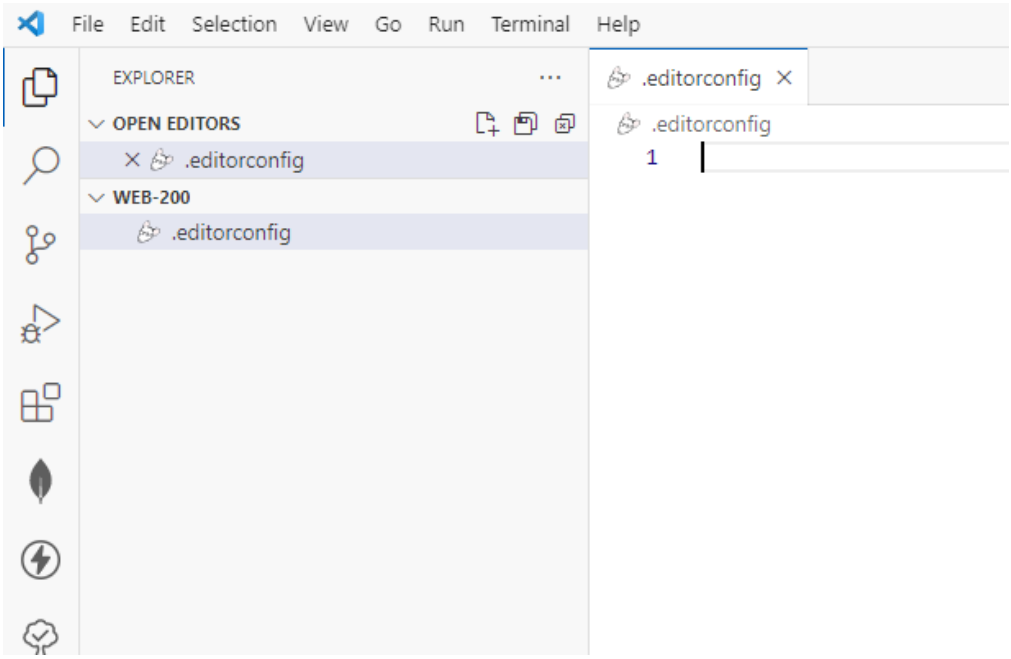
Before you proceed with this section, make sure you have installed the recommend extensions from "Week 0 – Pre-workout." An .editorconf file is a universal document used by most text editors and IDE's to standardize the formatting of a programming project. Effectively, it is a file format for defining coding styles and configurations/settings in a codebase. Styles such as, indent style, tab width, character set, and whitespaces are enforced at the project level. The advantages are: a consistent codebase, clear direction, and universal standards enforced across all development environments/platforms. For now, we are going to focus on the basics, by demonstrating how to enforce rules for character set, indent style, indent size, trailing whitespace, and final new line insertions.

For this course and all HTML, CSS, JavaScript, and TypeScript programming assignments we will be using two spaces for indention, utf-8 character set, spaces for indent style, and trim trailing whitespaces. The extension you added during Week 0 will enforce the rules defined in an editorconfig file. Also, each project you create must include a separate .editorconfig file, unless there is already one at the root of the directory. For example, let's say you create a child folder in the **buwebdev** directory and name it **fitness**. A new .editoconfig file will need to be added to the **fitness** directory; however, let's say you create a child folder under **web-200** and **web-200** already has a .editoconfig at the root of the project, you do not need to create a new .editorconfig file. Styles placed in the .editorconfig file are enforced at the root level of a project.

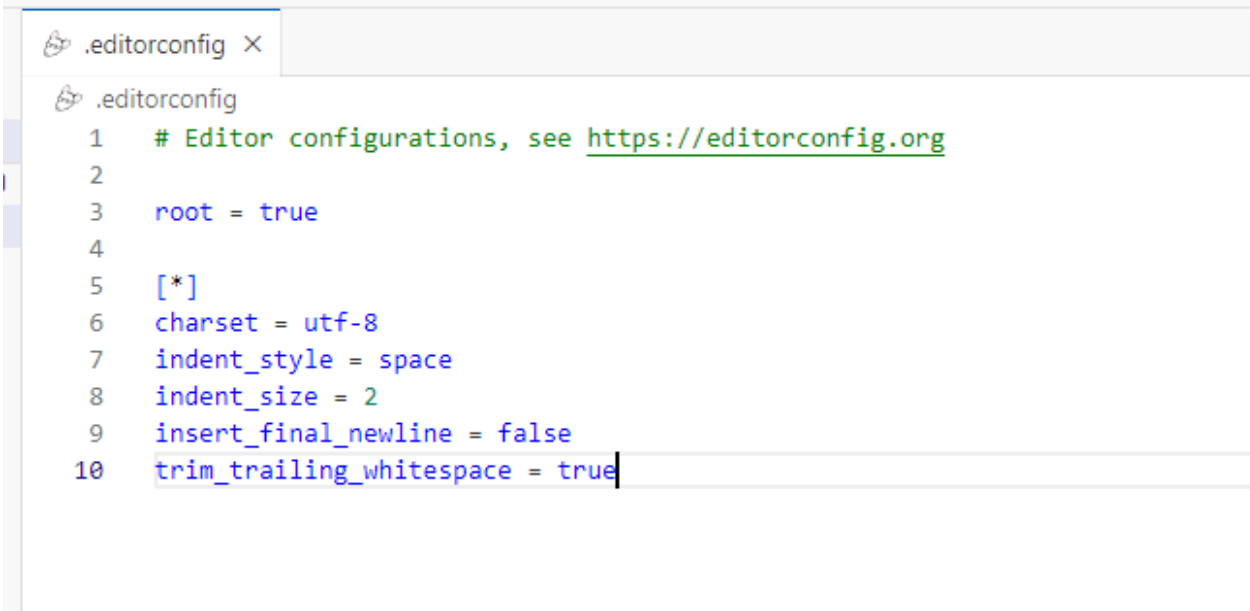
Open VSC and your web-200 directory.



Add a new file to your **web-200** directory (click on the file icon next to the projects name) and name it .editorconfig



Open the .editorconfig file and add the following lines of code:



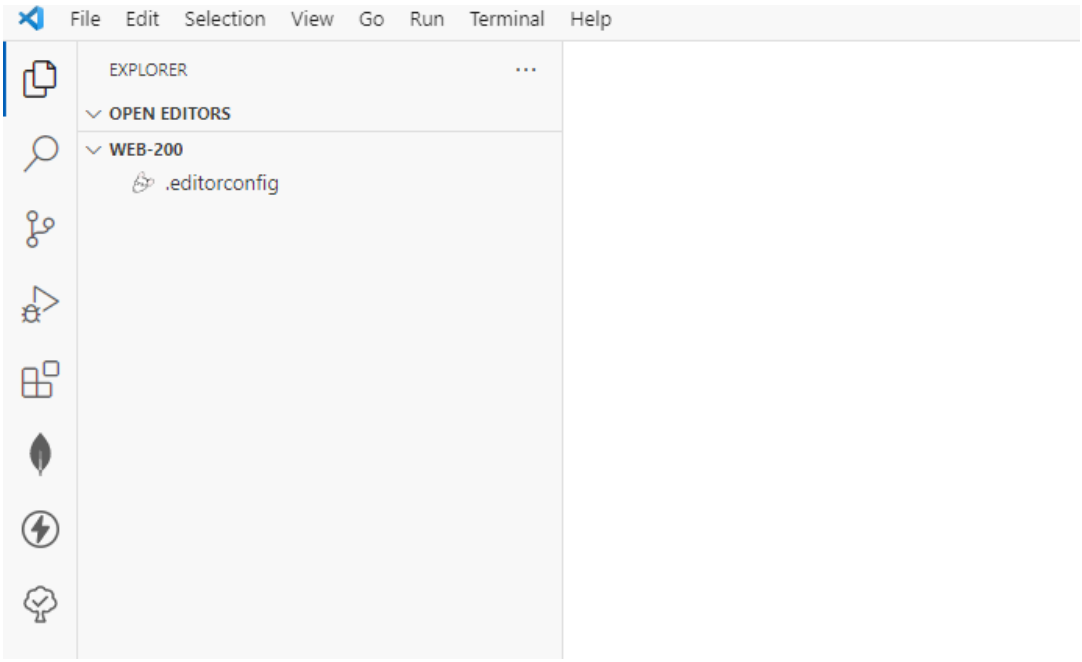
Code comments are created using the # symbol in an .editorconfig file. Line 3 defines the scope of the file. And, in our case, we are setting the scope as the root of the project. Line 5 defines file coverage and in our case, an asterisk is used to account for all file types. Line 6 specifies the character type, which in our case is utf-8. Line 7 specifies the type of indent style. Line 8 specifies the indent size, which in our case is set to two spaces. This means, each time you hit the **Tab** key from your keyboard, the cursor will indent the code two spaces. Line 9 specifies that we do not want to add a final new line at the end of a code sentence and line 10 specifies that we want extra whitespace from a code sentence to be trimmed.

Congratulations, you just learned how to create an .editorconfig file and what basic settings we will use in all HTML, CSS, JavaScript, and TypeScript programming assignments. To see the .editorconifg file in action, move on to the next section.

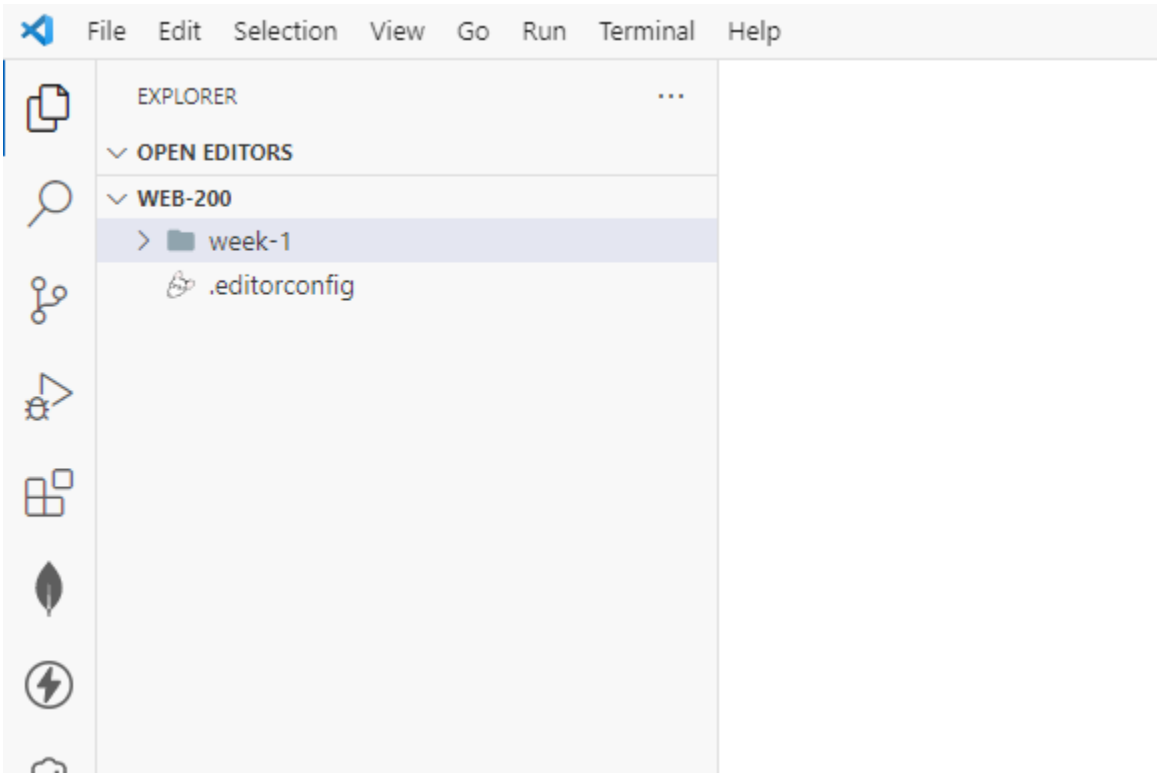
Creating an HTML file in VSC

This course uses HTML 5 for all programming assignments, which will be clearer once you are further along in the course. For now, just know we are using HTML 5 for all programming assignments. During the last section you learned how to create a .editorconfig file. But, an .editorconfig file by itself is not very fascinating. To grasp the concept of an .editorconfig file you are going to learn how to create a new HTML file in VSC.

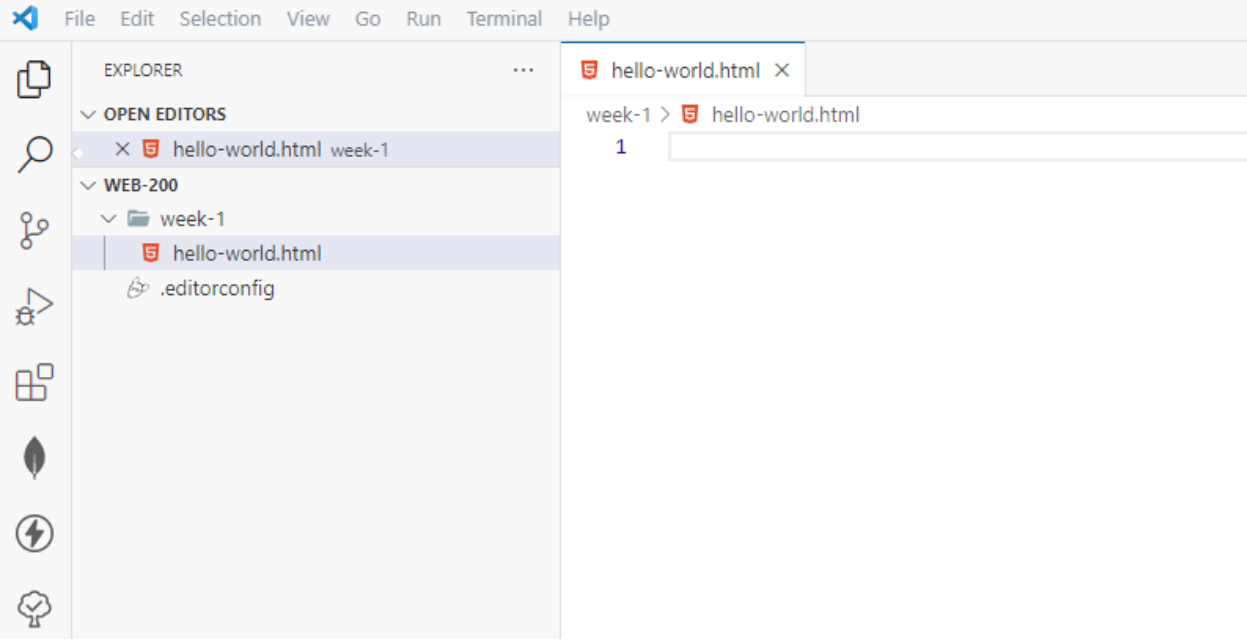
Open VSC and your **web-200** project.



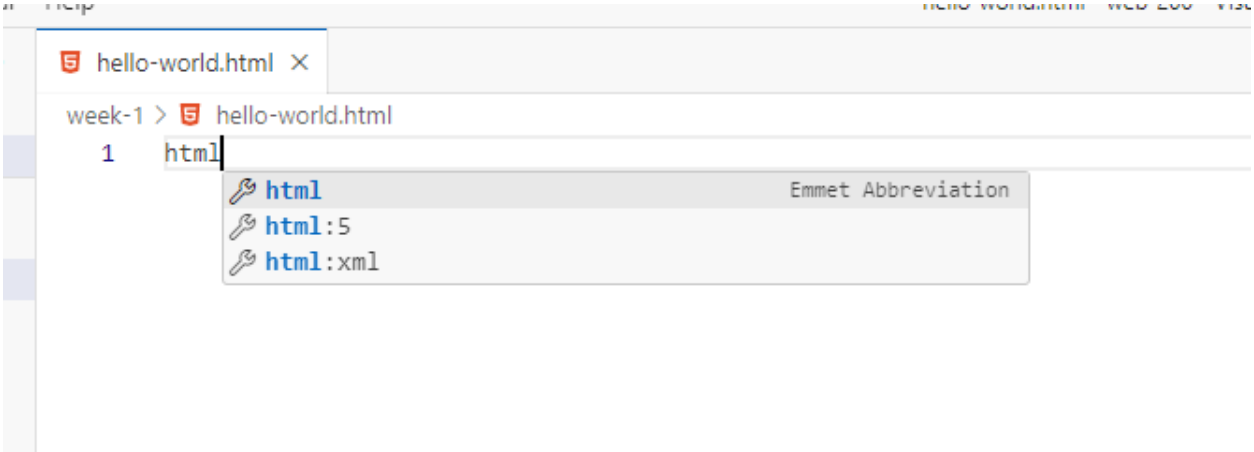
Create a new child folder named **week-1**.



Create a new file under **week-1** named **hello-world.html**.



With the file opened, type **html** and wait for IntelliSense to pick up your text.

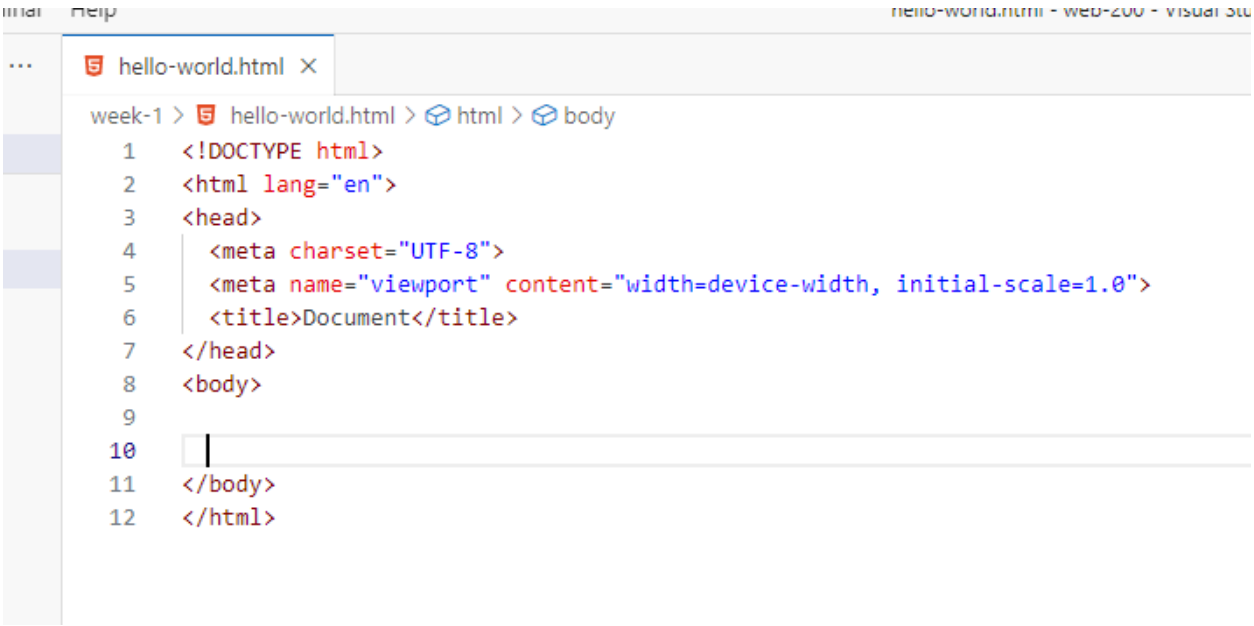


Using your mouse, hover over **html:5** and select it.

Something to consider: mine does not look like yours. There might be a slight delay (2-5 seconds) before IntelliSense acknowledges what you typed. If after waiting 2-5 seconds nothing happens, delete the word and type it again. If that does not work, close VSC and reopen it and try again. If that still does not work, move your cursor to the right of the word and press Ctrl + Space.

Something to consider: The courses textbook does not use an editor to autogenerate the HTML files. Instead, the author creates the files manually. Either approach is fine, but understand if you use the autogenerated approach, the line numbers from your file will not match the line numbers from the exercises in the course's textbook.

On line 9 (see the image above) use your mouse to select the area and press enter with your keyboard.

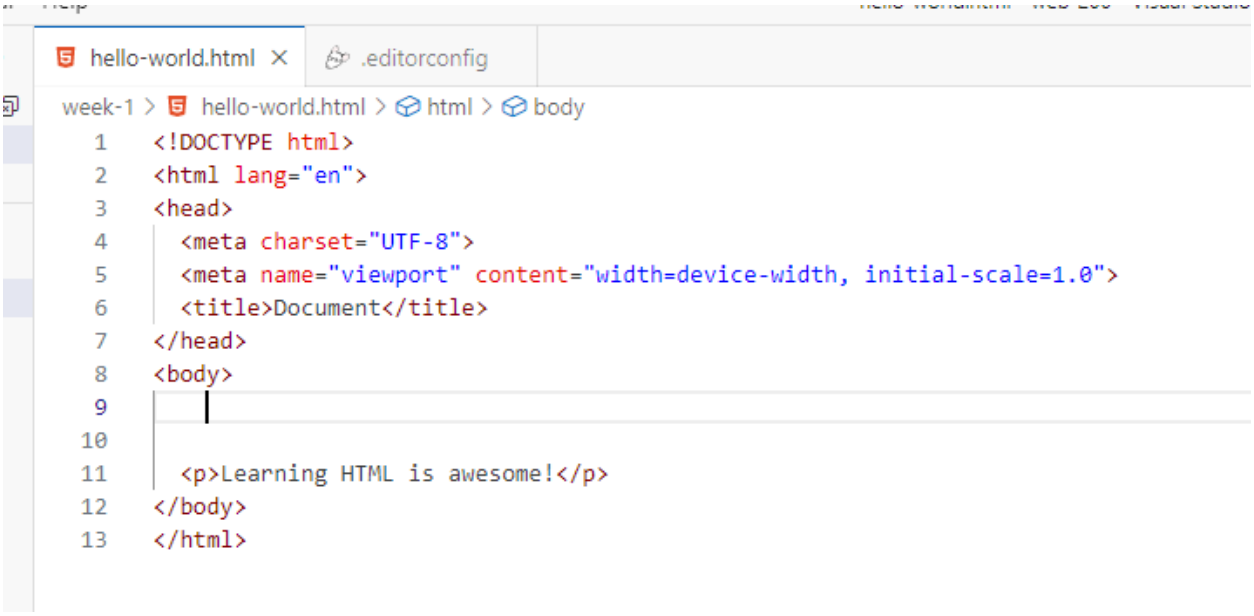


Notice how the cursor is two spaces from the left. Now, lets add a new <p> element with a value of **Learning HTML is awesome!**

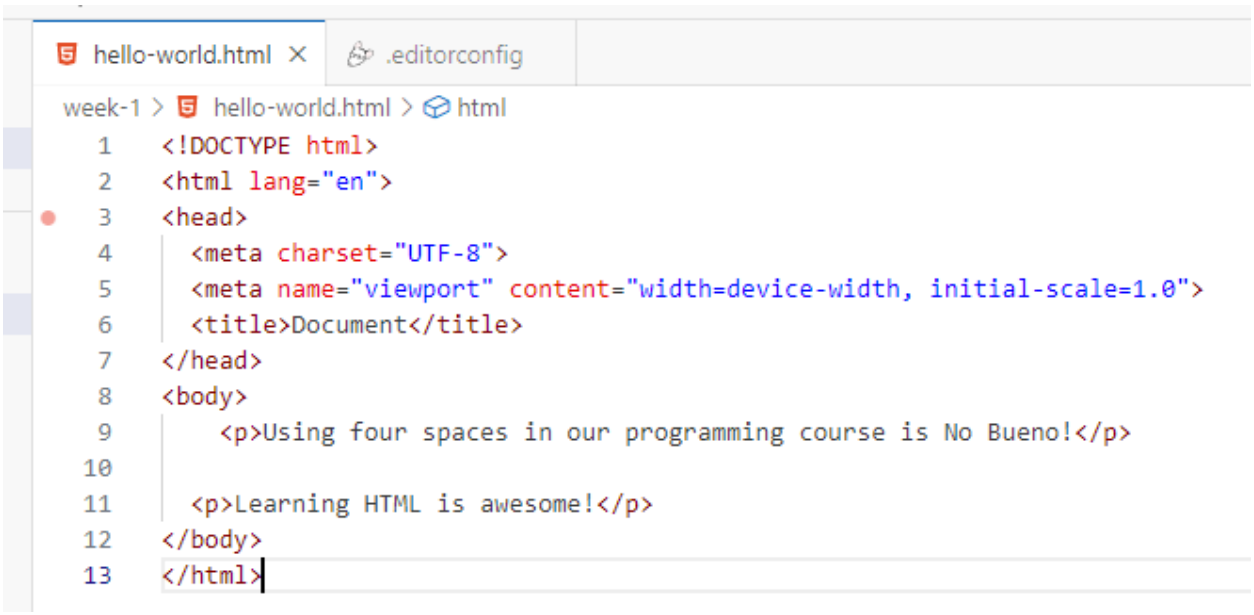
To see the .editorconfig file in action, go head and open the file and change the indent space value to 4.

Something to consider: Make sure you always save your work; otherwise, the changes will not be reflected in the code. File -> Save or Ctrl + S. A better approach, which I highly recommend is to configure VSC to auto-save. This way you never have to worry about the work not being saved. File -> Auto Save.

Switch back to the **hello-world.html** file and move your cursor to the end of line 8, the <body> element, and press enter.



Notice now that the cursor on 9 (above image) is using four spaces for indention. Whereas, the work we completed on line 11 is using 2 spaces (above image). For purely dramatic purposes, lets add a new <p> to line 9 with a value of **Using four spaces in our programming course is No Bueno!**



Can you spot the difference? It is pretty significant. Now image if you are working on a team with many developers and each developer uses their own preference for indention and line spaces. How quickly would things get out of hand? How difficult could it potentially be to read the code? Well this is the entire point of the .editorconfig file. It adds structure and enforces consistency.

Before we go any further, lets circle back and change .editorconfig file so it is using 2 spaces for indention and we also need to correct the No Bueno <p> so it mirrors our code on line 11. For fun, lets change the text on line 9 to **Using two spaces in our programming course is muy bien!**

hello-world.html | .editorconfig

week-1 > hello-world.html > html

```
1  <!DOCTYPE html>
2  <html lang="en">
3  <head>
4    <meta charset="UTF-8">
5    <meta name="viewport" content="width=device-width, initial-scale=1.0">
6    <title>Document</title>
7  </head>
8  <body>
9    <p>Using two spaces in our programming courses is muy bien!</p>
10
11   <p>Learning HTML is awesome!</p>
12 </body>
13 </html>
```

hello-world.html | .editorconfig

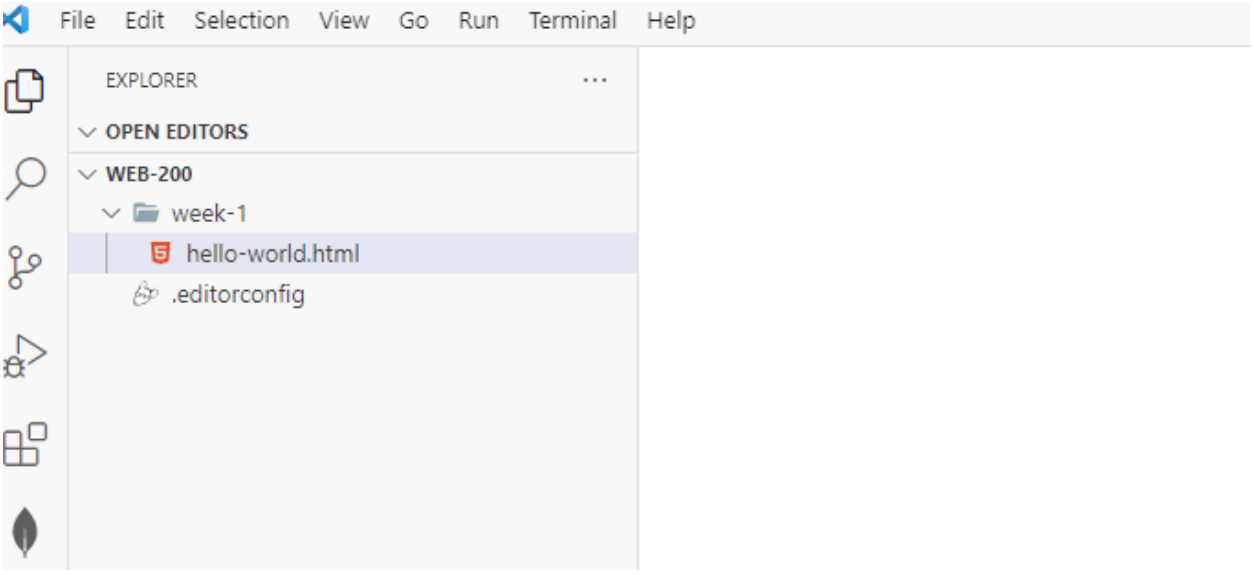
.editorconfig

```
1  # Editor configurations, see https://editorconfig.org
2
3  root = true
4
5  [*]
6  charset = utf-8
7  indent_style = space
8  indent_size = 2
9  insert_final_newline = false
10 trim_trailing_whitespace = true
```

Opening an HTML in VSC

During "Week 0 – Pre-workout" you were asked to install a VSC extension named **open in browser**. If you did not install this extension or skipped over your pre-workout, circle back and complete the beforementioned tasks.

Open VSC and your **web-200** directory. Expand **week-1**.



Open the **hello-world.html** file (make sure it opens in the editor window) and then right-click it. Select **Open in Default Browser**.

Something to consider: If you have multiple browsers installed on your computer, you can optionally select **Open in Other Browsers** and select a different browser to open the file in.

Congratulations, you have learned how to open an HTML file using the VSC extension **open in browser**.

Code Examples/Snippets:

Every programming course in this cohort has an associated GitHub repository with code snippets, examples, and starter projects. The base URL is https://github.com/buwebdev. It is highly recommended that you bookmark this link and frequently visit the repository of the current course you are taking. WEB 200's repository is located here. For now, you do not need to clone the repository or download anything, just make sure you bookmark it. For more information on git and GitHub review the material under **Week 1 Reading and Videos**.

Chapter 2 Coding Standards

Learning Objectives:

- Explain the coding standards for HTML and CSS programming assignments
- Describe the requirements for file header and block code comments
- Experiment with VSC extensions for code formatting
- Review the use of a .gitignore file

HTML and CSS Coding Standards

If you recall from Chapter 1 of this document, you learned that we will be using 2 spaces for indention in all HTML and CSS programming assignments. Additionally, the course's textbook does an excellent job of specifying how code should be organized in an HTML and CSS file. We will follow the authors approach to coding standards.

All programming assignments must include two spaces for indention, include appropriate line spaces, properly indented nested HTML elements, and closing tags must be level with the originating opening tag (reread this week's chapter, if you are not sure what this means).

There are two levels of code comments that are required in all programming files (HTML and CSS files). The first is **file header comments**. **File Header comments** are used to identify who wrote it, the date it was written, and the file name. All program files should have **file header comments** and it should be located at the **TOP** of the file. For the exact Line number, follow the textbook author's instructions. The following examples illustrate the requirements for **file header comments**.

HTML Example:

```
<!—
  Student Name: Wolfgang Mozart
  File Name: contact.html
  Date: May 20, 2021
-->
```

CSS Example:

```
/*
  Student Name: Richard Wagner
  File Name: style.css
  Date: May 20, 2021
*/
```

Something to consider: The student names, file names, and dates are arbitrary. You will need to replace them with your actual name, the files actual name, and the current date.

The second level of code comments is called **block comments**. **Block comments** should be placed at the top of a function or section of code. **Block comments** describe the purpose of the code and any algorithms used to accomplish the goal. **Block comments** are required in all CSS and HTML files. The course's textbook does an excellent job of illustrating how to add **block comments** to an HTML and CSS file. We will follow the authors approach to adding **block comments**.

Plagiarism is loosely defined as an individual claiming credit for another's ideas, work, or creative expressions. Anytime you use the work of another individual, which includes a

classmate, it must be documented in the block of code where it was used. For example, let's say I am stuck on a programming assignment and I use Google to find a solution. If I use the solution I found on Google, I must include a link to the source and a description of what I used above the block of code where it is being used. File Header comments, block comments, indention, line spaces, and organization are all gradable items in a submitted programming assignment.

Students wishing to achieve a letter grade of an A on a programming assignment should expect to exhibit a mastery level skill in the following criteria's:

- Code functionality. Does it work? Does it meet requirements?
- Adherence to standards and conventions (solid naming conventions, properly indented, great use of line spaces, and well-structured).
- Efficiency: Use of language features (are you using the built-in features of the language)
- Documentation: Code is maintainable by others (file header comments and block comments)
- Error trapping/handling (does the program have errors/does it compile)