31748 Programming on the Internet
32516 Internet Programming

Autumn 2026 (UTS)

Agenda

- Self-introduction
- Key Information about the Subject
- Basic Environment Setup
- CSS Layouts: Flexbox vs. Grid (on top of Lecture01 slides)
- In-class Practices: HTML and CSS

Key Dates

Last day to enroll: Monday, 2 March 2026 (Week 3)

Census date: Thursday, 19 March 2026 (Week 5)

Assessment Tasks

Assessment Task	Weight	Due date and time
Quiz 1	10%, individual	29 March 2026, 11:59pm
Assignment 1	40%, individual	5 April 2026, 11:59pm
Quiz 2	10%, individual	17 May 2026, 11:59pm
Assignment 2	40%, group-based, group and individually assessed	24 May 2026, 11:59pm
Quiz 1	10%, individual	29 March 2026, 11:59pm

- There is no final exam in the subject.
- 72 hours automatic extension does not apply to Quizzes (Assessment Task 3).
- In order to pass the subject, a student must achieve an overall mark of 50% or more.

Note: The "**Subject Information**" page may not be up-to-date.

Check the "**Modules**" section for accurate information about the subject.

Attendance Policy

Attendance is NOT required, though highly recommended.

Feel free to attend a different tutorial from the one you were originally allocated to, although we cannot officially change your class allocation.

Consequently, **be cautious when completing SFS**:

- Make sure you are rating the tutor you actually worked with.
- Refrain from rating a tutor if you haven't taken her/his classes – *instead, leave feedback on the tutor you actually worked with as part of your feedback on the subject.*

Where to get help?

Important things are worth mentioning a second time!

Questions about content of the subject

- Ask lecturer/tutors during lectures and tutorial hours (online/in-class)
- Post questions to the subject's discussion board

For extensions of

- **No more than 3 days**: request via "Assignment Extensions" on the subject site.
- **No more than 7 days**: Email the subject coordinator
- **More than 7 days**: Submit an eRequest through Student Admin, Accessibility Service, Special Consideration, etc.

Health issues, Learning difficulty, or Review of assessment results

- Book for consultation or email the subject coordinator

Tutors' Contacts

- Ms. Yong Pov, Yong.Pov@uts.edu.au
- Mr. Nimish, Nimish.Nimish@uts.edu.au
- Mr. Anthony Dang, Anthony.Dang@uts.edu.au
- Mr. Henil Patel, HenilPareshkumar.Patel@uts.edu.au (only in 32516)
- Dr. Xianzhi Wang, xianzhi.wang@uts.edu.au (only in 32516)

* All tutors and class allocations are available at the "**Classes and teaching staff**" page under "Modules".

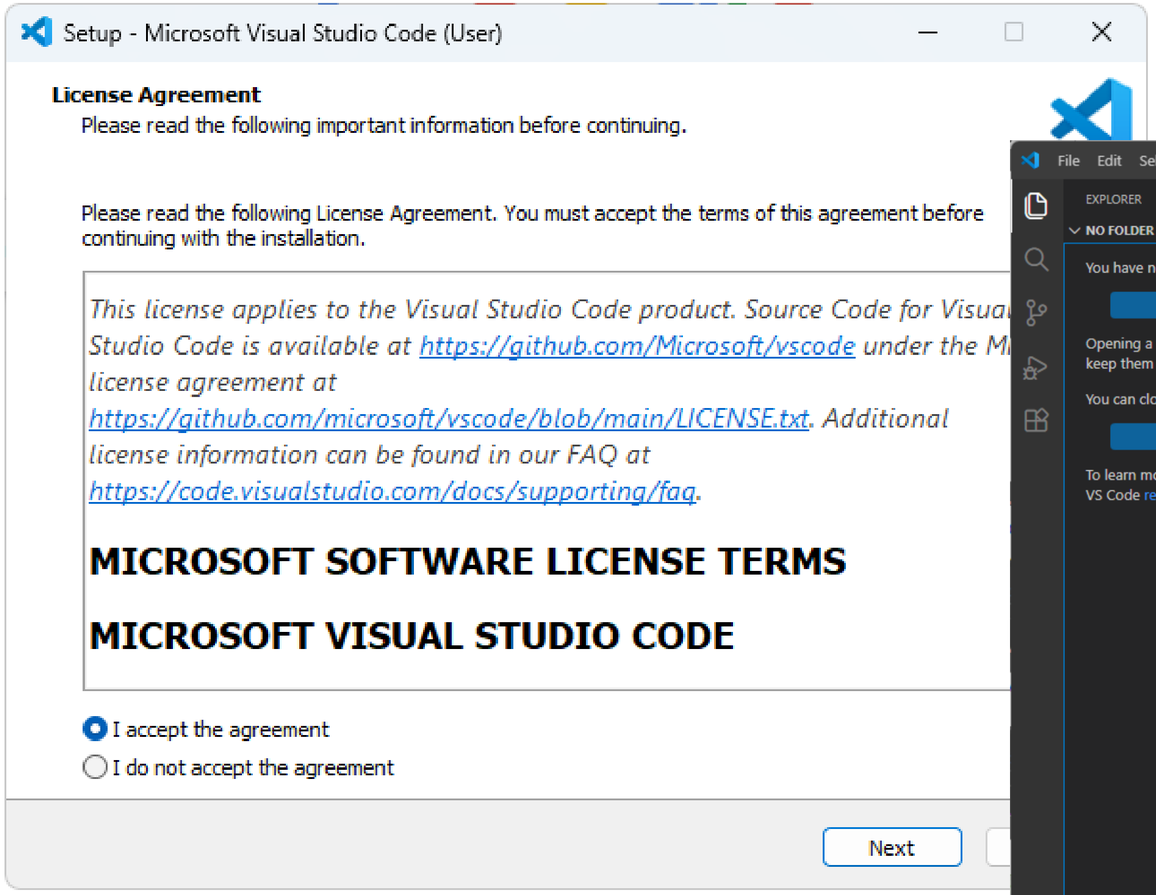
Basic Environment Setup

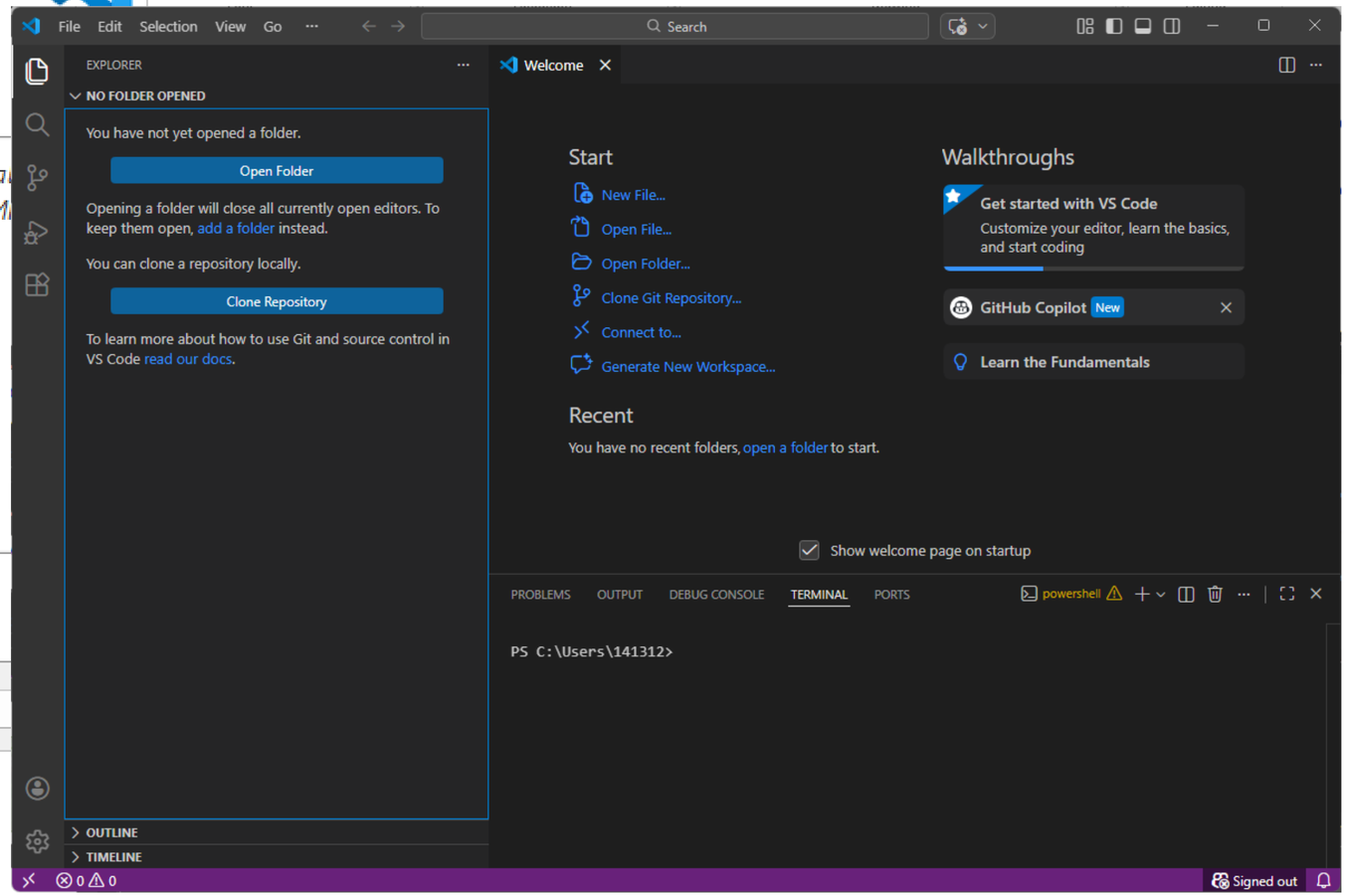
VS Code, Git, GitHub Copilot

Web Development Environment

- **Visual Studio Code (VS Code)**
  - A free, lightweight, and powerful open-source code editor developed by Microsoft for Windows, macOS, and Linux.
- **Git for Windows**
  - A free and open-source distributed version control system designed to handle everything from small to very large projects with speed and efficiency.
- **GitHub Copilot Extension (for VS Code)**
  - You typically get two extensions:
    - GitHub Copilot: Provides inline code suggestions as you type.
    - GitHub Copilot Chat: Provides conversational AI assistance.

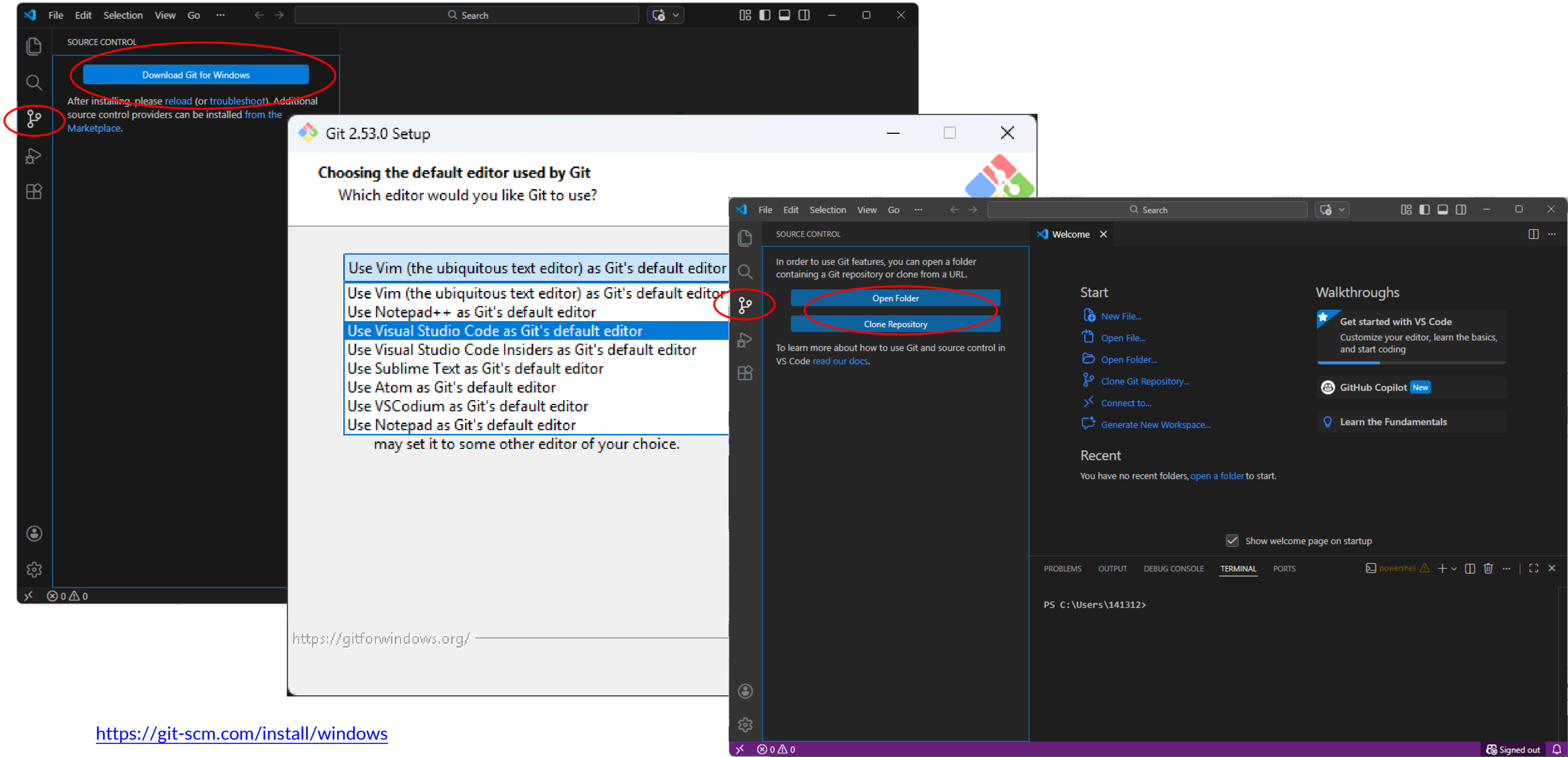
Visual Studio Code





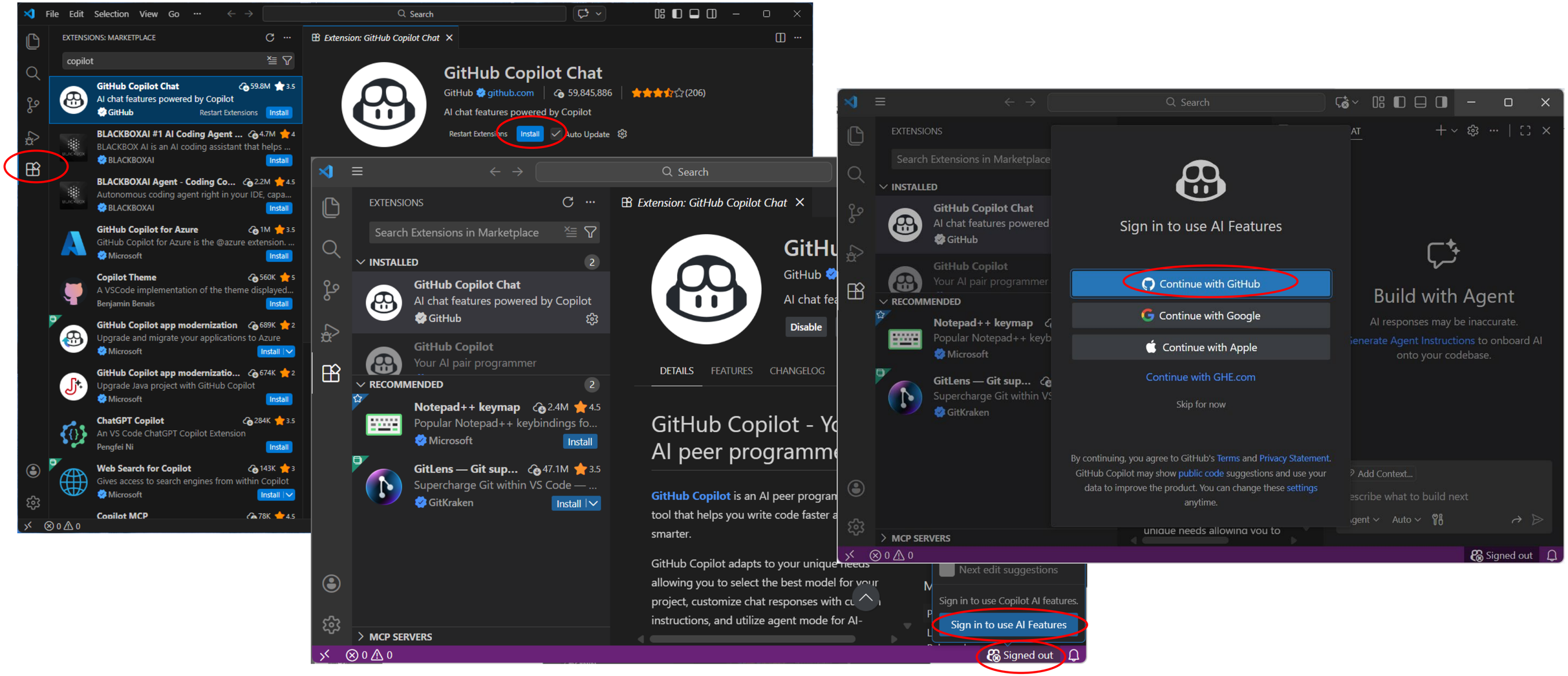
https://code.visualstudio.com/download

Git for Windows

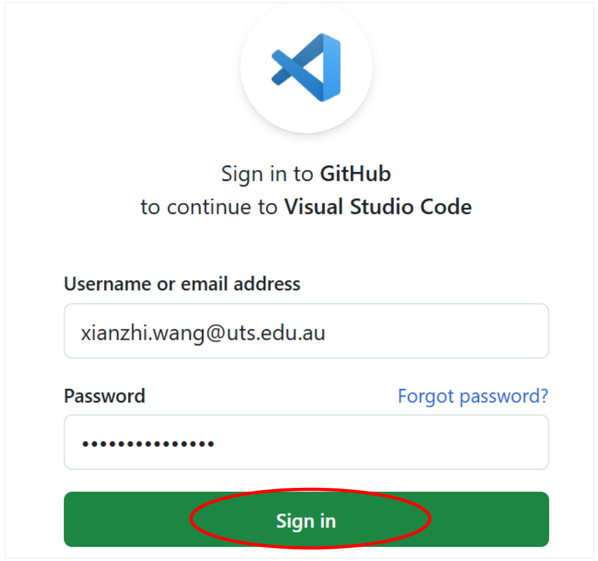


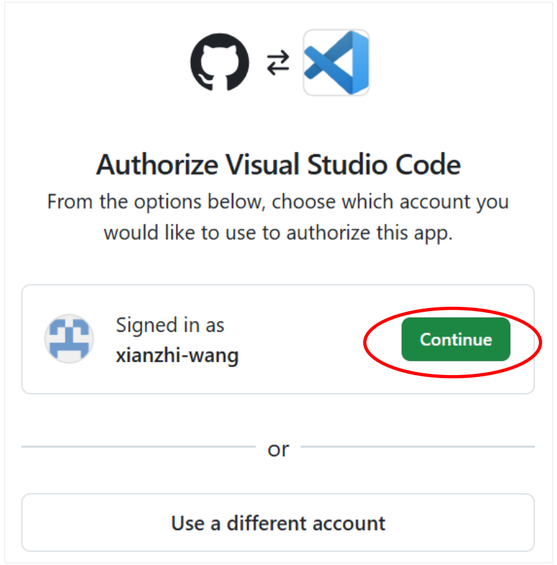
https://git-scm.com/install/windows

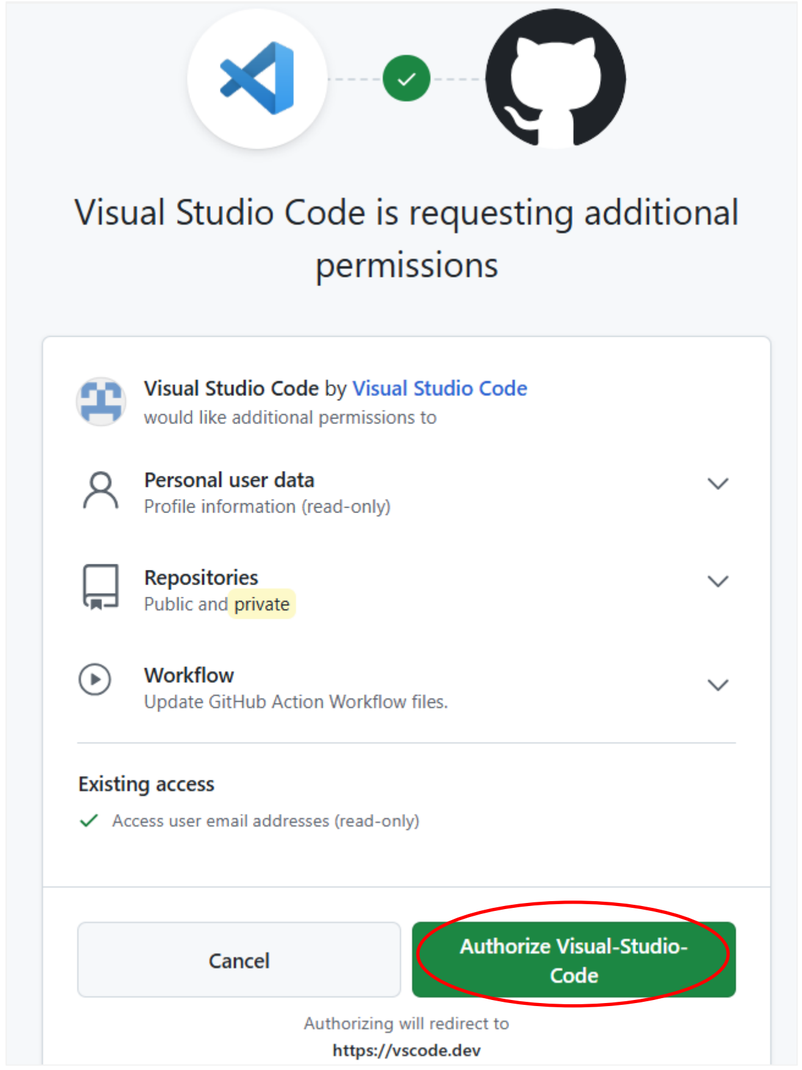
VS Code: GitHub Copilot Extension

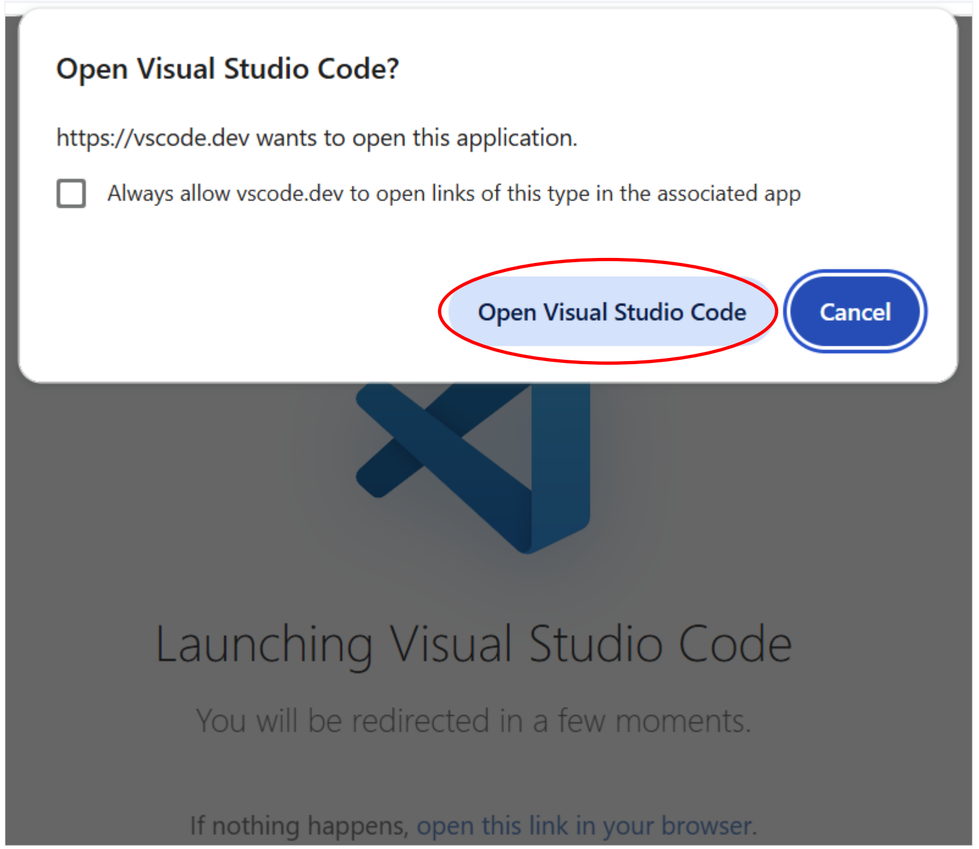


GitHub: sign-up, sign-in

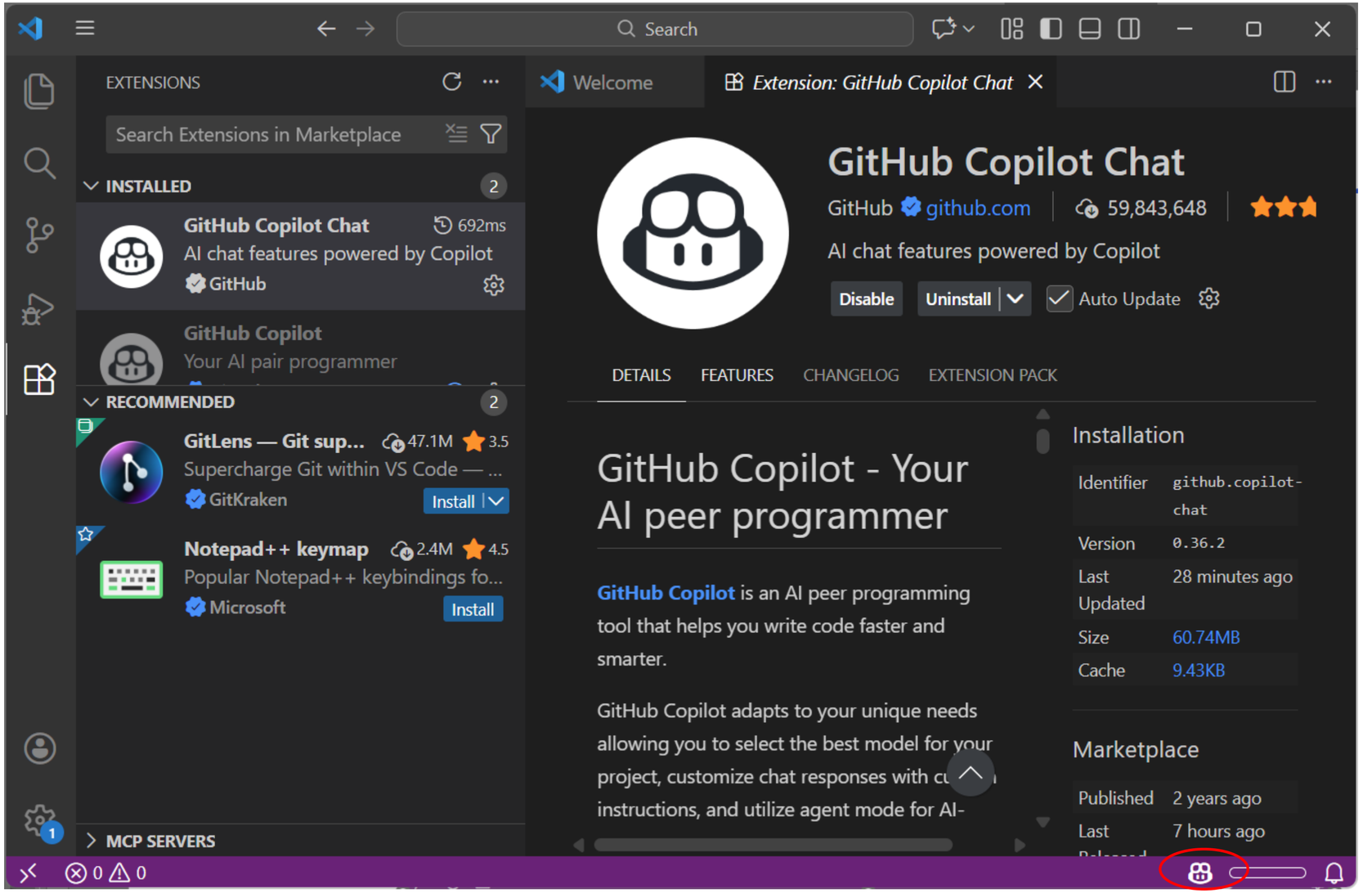




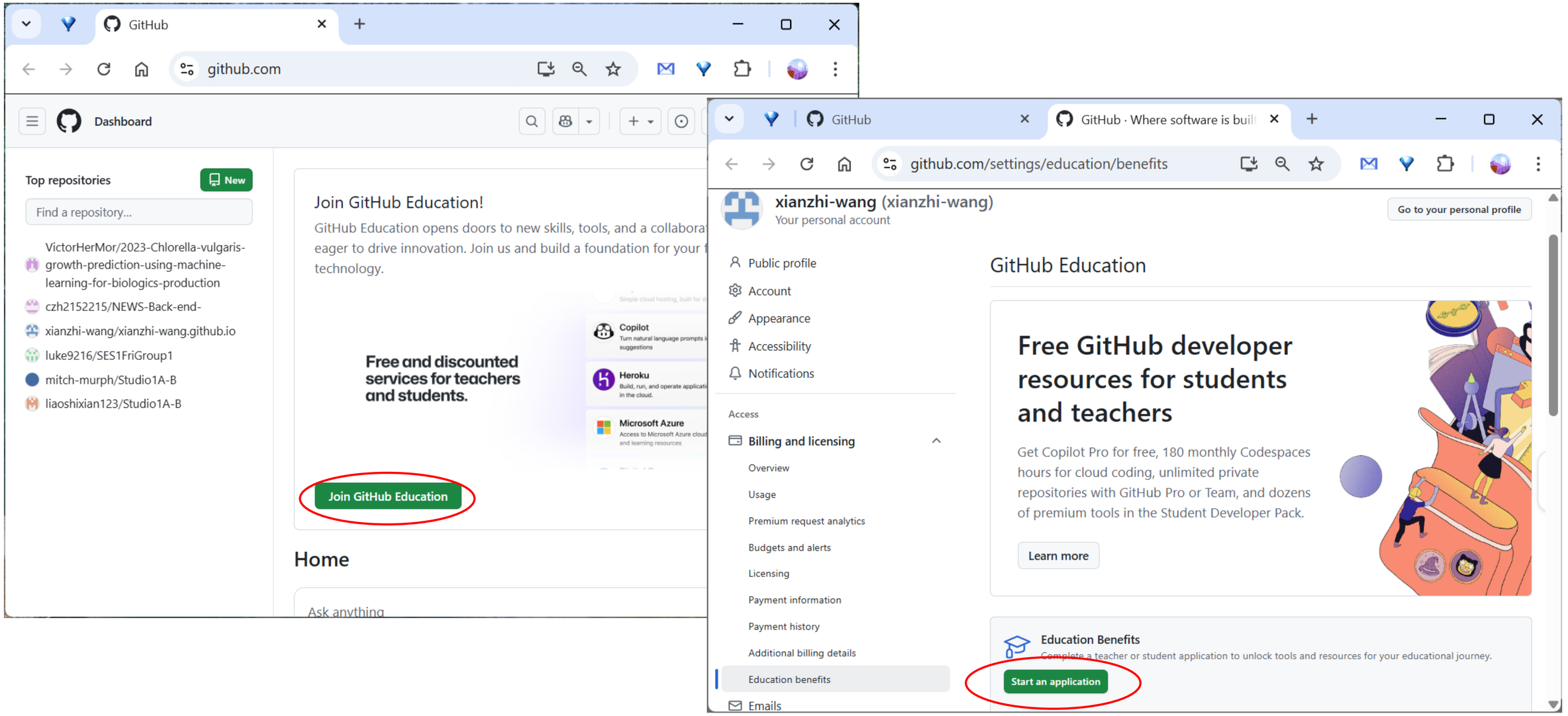




You must sign in to GitHub to use **Copilot** in VS Code to verify your user account has an active Copilot plan and for secure, authorized access to the service's AI model.



GitHub Education



CSS Layouts

Flexbox vs. Grid

* You’re expected to have reviewed Lecture01 slides when starting this section.

CSS Flexbox

CSS Flexible Box Layout module (Flexbox) is a one-dimensional layout system that arranges items either horizontally or vertically, in a flexible and responsive way.

A flexbox always consists of:

- **A Flex Container:** The parent (container) element, where the display property is set to flex or inline-flex.
- **One or more Flex Items**: The direct children of the flex container, which automatically become flex items.

```
.flex_container {
  display: flex;
}
```

https://www.w3schools.com/css/css3_flexbox.asp

CSS Flexbox (cont'd)

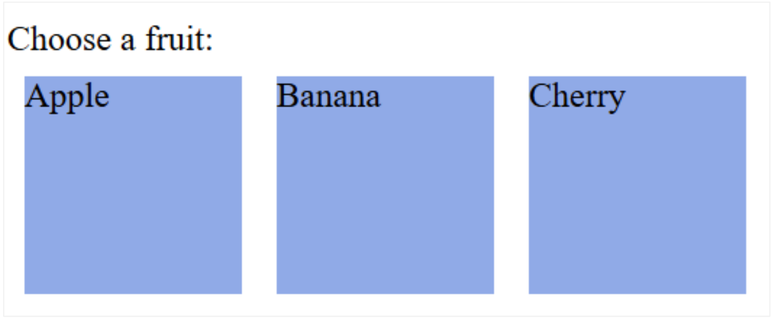
The .fruit-list class applies display: flex, designating the as a flex container.

Each inside the .fruit-list class automatically becomes a flex item, styled to have equal *width* and *height* within the container, with *margins* for spacing.

```html
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Flexbox Example</title>
  <link rel="stylesheet" href="css/test.css">
</head>
<body>
  Choose a fruit:
  <div class="fruit-list">
    <div>Apple</div>
    <div>Banana</div>
    <div>Cherry</div>
  </div>
</html>
```

```css
/* Make parent element a flex container */
.fruit-list {
  display: flex; /* or inline-flex */
}

/*Style the child elements (flex items) */
.fruit-list div {
  width: 100px;
  height: 100px;
  background-color: #8cacea;
  margin: 8px;
}
```



 elements align vertically if Flexbox is not used.

CSS Flexbox (cont'd)

Some useful Flexbox properties for aligning and justifying content within the container:

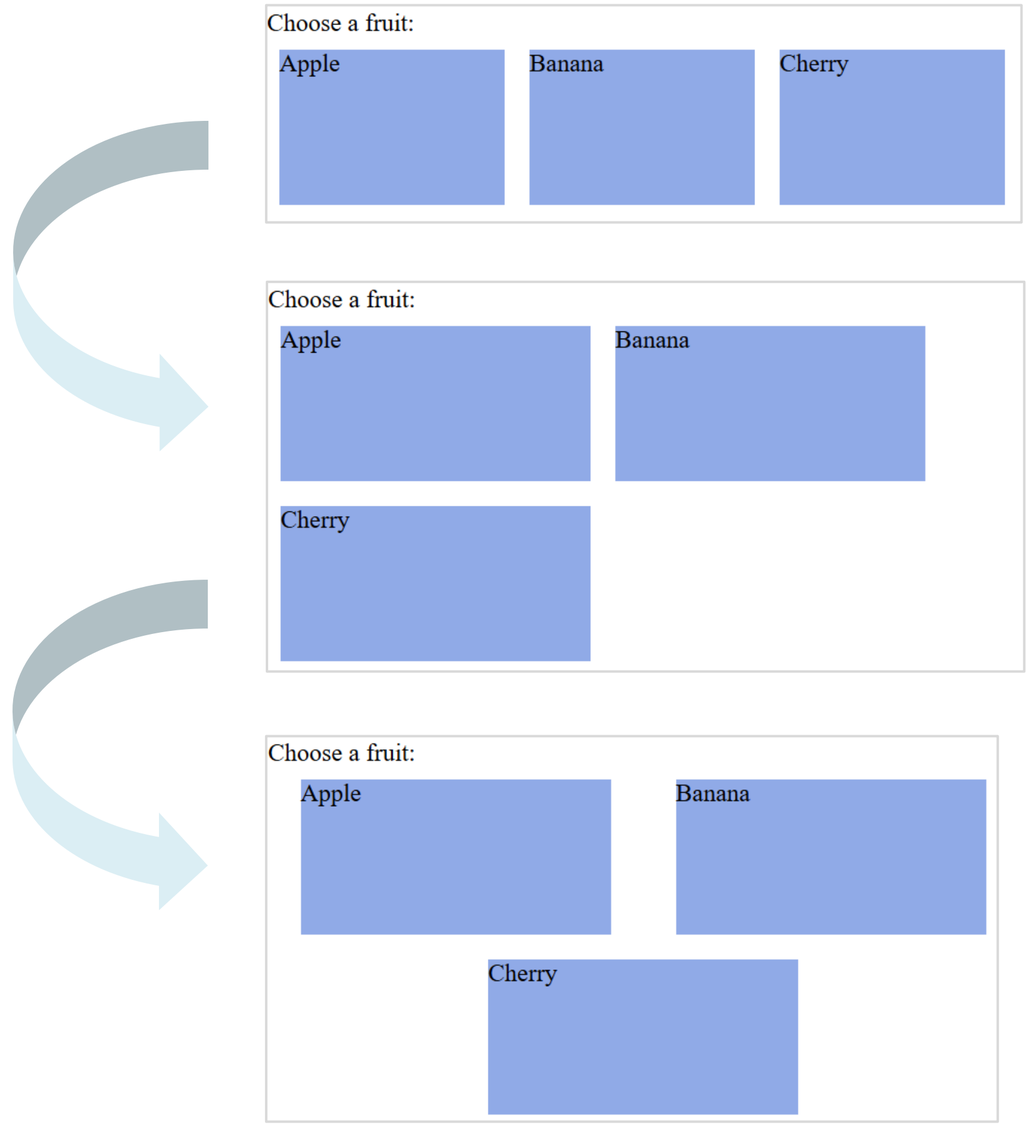
- **flex-direction:** column**:** arranges flex items vertically (*horizontally* by default).
- **flex-wrap:** wrap: allows items to flow onto new lines (not allowed by default)
- **justify-content**: defines how flex items are laid out along the main axis, e.g., flex-start (default), flex-end, center, space-between, space-around.
- **align-items**: defines how flex items are laid out along the cross axis (similar to justify-content by on a different axis).
- **align-content**: controls how the flex-items are aligned in a multi-line flex container (similar to align-items).

https://medium.com/free-code-camp/understanding-flexbox-everything-you-need-to-know-b4013d4dc9af

An Example

```css
/* Make parent element a flex container */
.fruit-list {
  display: flex; /* or inline-flex */
  flex-wrap: wrap; /* allow items to wrap to the next line */
}

/*Style the child elements (flex items) */
.fruit-list div {
  width: 200px;
  height: 100px;
  background-color: #8cacea;
  margin: 8px;
}
```




```css
/* Make parent element a flex container */
.fruit-list {
  display: flex; /* or inline-flex */
  flex-wrap: wrap; /* allow items to wrap to the next line */
  justify-content: space-around; /* distribute items evenly with space around them */
}

/*Style the child elements (flex items) */
.fruit-list div {
  width: 200px;
  height: 100px;
  background-color: #8cacea;
  margin: 8px;
}
```

https://www.w3schools.com/css/css_grid.asp

CSS Grid

The CSS Grid Layout module (**CSS Grid**) is a two-dimensional layout system that arranges items in rows and columns, in a flexible and responsive way.

A grid always consists of:

- **A Grid Container**: The parent (container) element, where the display property is set to grid or inline-grid.
- **One or more Grid Items**: The direct children of the grid container automatically becomes grid items.

```
.grid-container {
  display: grid;
}
```

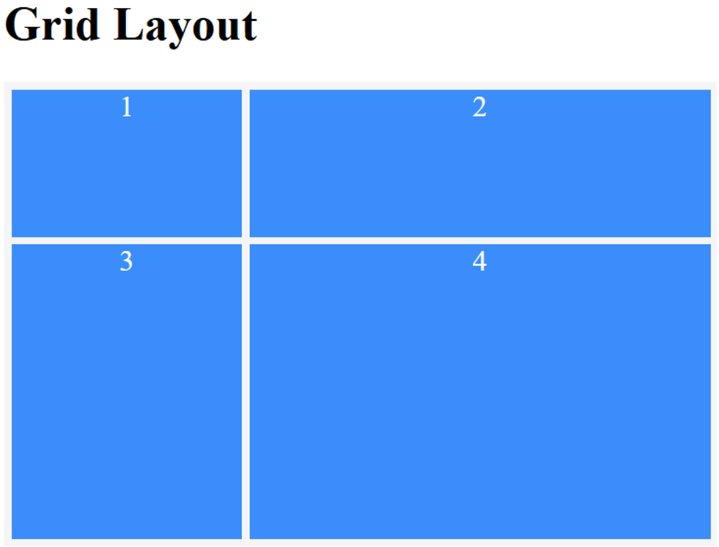
https://www.w3schools.com/css/css_grid.asp

CSS Grid (cont'd)

grid-template-rows and grid-template-columns: specify the number of rows/columns using fixed lengths, percentages, or even the *fr* unit (a fraction of the available space)

```html
<html>
<head>
  <link rel="stylesheet" href="css/test.css">
</head>
<body>
  <h1>Grid Layout</h1>
  <div class="grid-container">
    <div>1</div>
    <div>2</div>
    <div>3</div>
    <div>4</div>
  </div>
</body>
</html>
```

```css
.grid-container {
  display: grid; /* CSS Grid with 2 rows and 2 columns */
  grid-template-rows: 100px 200px; /* Heights of the rows */
  grid-template-columns: 1fr 2fr; /* Widths of the columns */
}
```

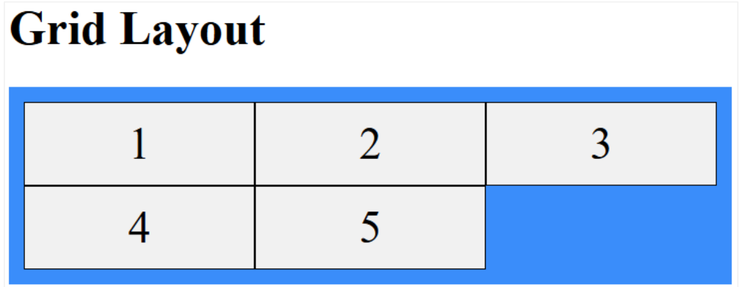


CSS Grid (cont'd)

You can use the auto keyword to let the browser automatically determine the width of a column based on its content.

```html
<html>
<head>
  <link rel="stylesheet" href="css/test.css">
</head>
<body>
  <h1>Grid Layout</h1>
  <div class="container">
    <div>1</div>
    <div>2</div>
    <div>3</div>
    <div>4</div>
    <div>5</div>
  </div>
</body>
</html>
```

```css
.container {
  display: grid;
  grid-template-columns: auto auto auto;
  background-color: dodgerblue;
  padding: 10px;
}
.container div {
  background-color: #f1f1f1;
  border: 1px solid black;
  padding: 10px;
  font-size: 30px;
  text-align: center;
}
```



https://www.w3schools.com/css/css_grid.asp

CSS Grid (cont'd)

There are more properties of CSS Grid, many being similar to those of Flexbox:

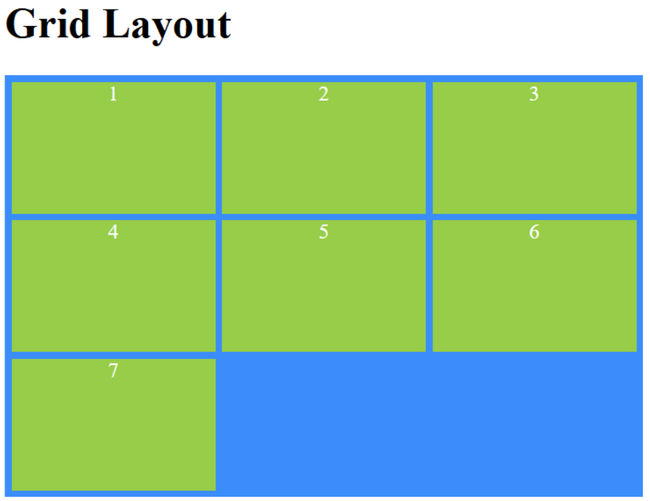
- column-gap | row-gap | gap: specify the size of the gap between rows, columns, both rows and columns within the grid.
- justify-items | align-items: determine how grid items are aligned along the grid's inline (row) and block (column) axes.
- justify-content | align-content: determine how the entire grid container's content is aligned along the row and column axes.

https://www.w3schools.com/css/css_grid.asp

An Example

```html
<html>
<head>
  <link rel="stylesheet" href="css/test.css">
</head>
<body>
  <h1>Grid Layout</h1>
  <div class="grid-container">
    <div>1</div>
    <div>2</div>
    <div>3</div>
    <div>4</div>
    <div>5</div>
    <div>6</div>
    <div>7</div>
  </div>
</body>
</html>
```

```css
.grid-container {
  display: grid; /* CSS Grid with 2 rows and 2 columns */
  grid-template-rows: 100px 100px 100px; /* Heights of the rows */
  grid-template-columns: 1fr 1fr 1fr; /* Widths of the columns */
  background-color: DodgerBlue;
  gap: 5px; /* Gap between grid items */
  padding: 5px; /* Padding around the grid container */
}
.grid-container div {
  background-color: yellowgreen;
  color: white; /* Text color */
  text-align: center; /* Center text horizontally within each cell */
}
```



Flexbox vs. Grid

Grid is mostly defined on the parent element. In **Flexbox**, most of the layout (beyond the very basics) happen on the children.

Grid is better at overlapping. Getting elements to overlap in Flexbox requires breaking out of the flex behavior. Grid inherently supports placing items on overlapping grid lines, or even within the same exact grid cells.

Grid is sturdier. The way a flex item is sized gets rather complicated. Grid sets up grid lines and places items within them right into place.

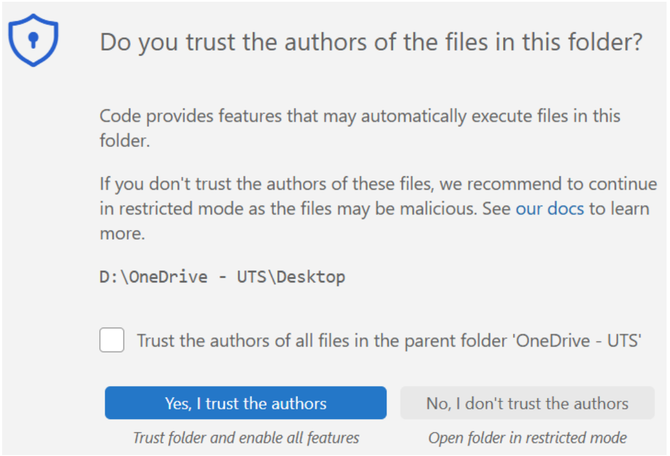
https://css-tricks.com/quick-whats-the-difference-between-flexbox-and-grid/

In-class Practices

HTML and CSS

Preparation

1. Make sure VS Code is appropriately set up on your Computer.
2. Download **Tutorial01_questions.zip** under the Tutorial 1 folder in Canvas.
3. Extract it into a new folder in your preferred location.
4. Open this new folder with VS Code.
   - Select "Yes, I trust the authors" if prompted
   - This folder is now your project folder in VS Code.



5. This folder contains the practices with incomplete code for you to start with.

*If you are getting stuck in the middle, check our solutions in **Tutorial01_solutions.zip**.*

1. Birdwatching Site

This practice tests your ability to 1) choose appropriate structural semantics to build a page; 2) make a few additions to the page content; 3) apply CSS layouts to the page.

The site structure needs to consist of the following:

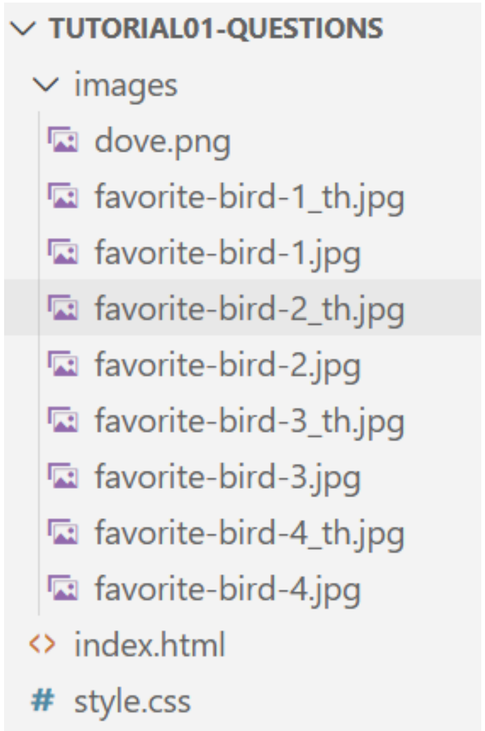
- A **header** that wraps the top-level page heading and the navigation menu list.
- An additional wrapper around the **navigation** menu list.
- A **main content** area containing two columns — a main **article** to contain the welcome text, and a **sidebar** to contain the image thumbnails.
- A **footer** containing the copyright information and credits.

Content additions:

- Inside the <h1> element, add an <img> element that includes the dove logo on the page.
- The "Home", "Get started", "Photos", "Gear", and " Forum" text items should be turned into a navigation menu – *you may mark them up as an unordered list.*
- Replace the <!-- Link images here. --> comment with a set of four thumbnail images. Each one should be wrapped in an <a> element that links to the full-sized equivalent.

https://developer.mozilla.org/en-US/docs/Learn_web_development/Core/Structuring_content/Structuring_a_page_of_content

1. Birdwatching Site



You will need the following images in your page:

- dove.png: The website logo
- favorite-bird-1.jpg: Full-size version of the 1st sidebar image
- favorite-bird-1_th.jpg: Thumbnail of the 1st sidebar image
- favorite-bird-2.jpg: Full-size version of the 2nd sidebar image
- favorite-bird-2_th.jpg: Thumbnail of the 2nd sidebar image
- favorite-bird-3.jpg: Full-size version of the 3rd sidebar image
- favorite-bird-3_th.jpg: Thumbnail of the 3rd sidebar image
- favorite-bird-4.jpg: Full-size version of the 4th sidebar image
- favorite-bird-4_th.jpg: Thumbnail of the 4th sidebar image

Hint: *You may refer to style.css to get some idea about which tags could be used.*

https://developer.mozilla.org/en-US/docs/Learn_web_development/Core/Structuring_content/Structuring_a_page_of_content

BIRDWATCHING

HOME | GET STARTED | PHOTOS | GEAR | FORUM

WELCOME

Welcome to our fake birdwatching site. If this were a real site, it would be the ideal place to come to learn more about birdwatching, whether you are a beginner looking to learn how to get into birding, or an expert wanting to share ideas, tips, and photos with other like-minded people.

So don't waste time! Get what you need, then turn off that computer and get out into the great outdoors!

FAVORITE PHOTOS

Dove icon by Lorc.

2. Feedback Form

This practice tests your ability to design and apply appropriate CSS styles to an existing web page.

The site has several parts that require better styling:

- **<body>**: you may want to apply a comfortable background color, etc. to the whole area.
- **Headings and text**: it is desirable to apply consistent designs of font sizes, margins and padding to the same type of HTML elements.
- **Form structure and individual form items**: fieldset, separator, label, textarea, button, etc.

Hints:

- There is no one-size-fits-all solution to this practice.
- You can use **AI-generated code and comments** in VS Code to start with.

https://developer.mozilla.org/en-US/docs/Learn_web_development/Core/Structuring_content/Forms_challenge to

We want your feedback!

We're very excited that you visited the little house in the woods, and we want to hear what you thought of it! Please fill in the below sections. You don't need to provide your name or contact details, but if you do, we'll enter you into a prize draw where you'll have a chance to win prizes.

Facilities

Was the porridge

- (•) Too hot?
- ( ) Too cold?
- ( ) Just right?

Were the beds

- (•) Too hard?
- ( ) Too soft?
- ( ) Just right?

Describe the chairs (select all you agree with)

- [ ] Comfy
- [ ] Luxurious
- [ ] Hi-tech
- [ ] Pretty
- [ ] Majestic

About your hosts

Who's your favorite bear? Papa bear

Which greeting did you prefer? Wave

Any other feedback?

Give us your comments

Kindly vacate the classroom 10 mins before next class.