**WEEK 1-WDD 130 (pixlr website to edit pictures)**

**Setup: Code Editor - Visual Studio Code**

**Overview**

The purpose of this setup activity is to install Microsoft's **Visual Studio Code**. VS Code is free to install on Mac, Windows, and Linux operating systems. This is the editing tool that will be used in this course to write and edit web page code.

**Instructions**

**Download and Install VS Code**

1. In your browser, navigate to [Visual Studio Code](https://code.visualstudio.com/).
2. You will be prompted to download VS Code for your operating system. Click **download** and follow the directions to download and install VS Code.

**Demonstrations**  
🎦 [Installing VS Code on **Windows**](https://youtu.be/Epiixq4F1E0)  
🎦 [Installing VS Code on **Mac**](https://youtu.be/KiU3Hf5FsLU)

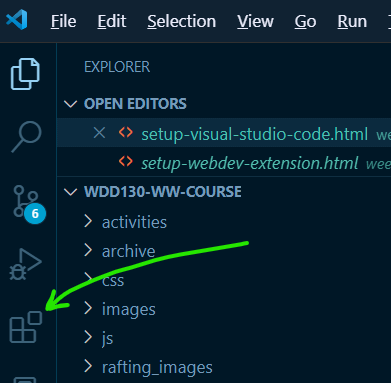
**Set your VS Code Environment Preferences**

1. Install your VS Code interface **preferences**. This video, even though it ends abruptly, covers a very brief introduction to VS Code and shows how you can edit some interface preferences in the program.  
   🎦 [Visual Studio Code Introduction](https://youtu.be/YPPeNcbZy1o).

**Install the Live Server Extension**

One of the reasons that VS Code is popular and useful is that custom extensions can be added. In this class we will use the **Live Server** extension to display the web pages locally, in our browser, without having to be on the web.

1. In your browser, navigate to the [Live Server](https://marketplace.visualstudio.com/items?itemName=ritwickdey.LiveServer) extension in Extension Marketplace.
2. Click **Install** and follow the prompts carefully to install the extension within VS Code.

Figure 1: Extension Icon in the Activity Bar

Note that it is very common to [manage extensions](https://code.visualstudio.com/docs/editor/extension-marketplace) using the Extensions View. The Extensions View icon is found in the Activity Bar.

1. A demonstration of how to use the extension will be shown later. Basically, any file with the .html extension can be opened using Live Server from within VS Code and the default browser will launch and render the page as if on a web server. This way we can test locally without being on the internet.

**Visual Studio Code: Additional Resources**

* [Documentation](https://code.visualstudio.com/docs)
* [Tips and Tricks](https://code.visualstudio.com/docs/getstarted/tips-and-tricks)
* 📑 [Keyboard Shortcuts for Windows](https://code.visualstudio.com/shortcuts/keyboard-shortcuts-windows.pdf)
* 📑 [Keyboard Shortcuts for macOS](https://code.visualstudio.com/shortcuts/keyboard-shortcuts-macos.pdf)

[**Next ➡️ Web Dev Extension**](https://byui-cse.github.io/wdd130-ww-course/week01/setup-webdev-extension.html)

<https://byui-cse.github.io/wdd130-ww-course/week01/setup-webdev-extension.html>

**Setup: Browser Web Developer Extension**

**Overview**

To aid our development cycle and reviews, we will use a browser extension named **Web Developer**. This extension provides access to some endorsed validation and page checking tools including [W3C HTML validation](https://validator.w3.org/nu/), [W3C CSS validation](https://jigsaw.w3.org/css-validator/), and [WAVE accessibility](https://wave.webaim.org/).

"The Web Developer extension adds various web developer tools to a browser. The extension is available for Chromium based browsers, Firefox, and Opera, and will run on any platform that these browsers support including Windows, macOS, and Linux." - chrispederick.com

**Prepare**

**Extensions**

Browser extensions are small, dependent, software modules that allow users to customize and enhance their browsing experience. Browser extensions are managed within the browser settings. Access to the settings vary but are similar between browsers using the upper right menu access button ( ••• ≡ ) in the browser window. For example, in Google Chrome, you can manage extensions by navigating to **chrome://extensions**.

For a comprehensive overview of browser extensions, visit [Wikipedia's Browser extension](https://en.wikipedia.org/wiki/Browser_extension) article.

**Document Validation**

We will use the **Web Developer** extension to help us validate our HTML markup and CSS. Validation means that we check our document to see if it follows the rules of the language being checked. This is similar to checking the spelling and grammar in a document of a spoken language.

Tools like this help developers find issues that may cause the page to render in unexpected ways.

**Instructions**

You will install the extension on your browser in this activity to be used throughout the course. In the end, the extension should appear and be ready to use on your browser toolbar.

1. Install the [Web Developer](https://chrome.google.com/webstore/detail/web-developer/bfbameneiokkgbdmiekhjnmfkcnldhhm?hl=en-US) browser extension.

You may elect to add an extension to your selected browser using the browser tools that you are familiar with versus navigating to these instructions and link. For example, using the [Chrome Web Store](https://chrome.google.com/webstore/category/extensions) to find and install extensions.

1. Follow the directions and install the browser extension.
2. Once installed, you should see the extension icon on your browser toolbar.

🎦 [Web Developer Extension](https://www.loom.com/share/011b610fa093414e9745398365177acf) - A brief video demonstrating the use of the Web Developer extension.

**Additional Resources**

📄 [Install and manage extensions](https://support.google.com/chrome_webstore/answer/2664769?hl=en) (Chrome) - support.google.com  
📄 [Find and install add-ons](https://support.mozilla.org/en-US/kb/find-and-install-add-ons-add-features-to-firefox) - (Firefox) - support.mozilla.org

[**Next ➡️ GitHub Account**](https://byui-cse.github.io/wdd130-ww-course/week01/setup-github.html)

**Setup: GitHub Account**

**Overview**

The purpose of this setup activity is to obtain a free **GitHub** account. GitHub is a free, popular collaboration and version control online service. You can think of the GitHub repositories as a cloud-based storage system for your projects. GitHub can be used as a web hosting service with a special web service named **GitHub Pages**. We will use GitHub Pages to publish assignments in this course. You will work locally and then when you are ready to publish to the web, you will upload or push the content to the remote, duplicate repository.

If you already have a GitHub account, you are good to go and do **not** need to create a new account with GitHub for this course. You can skip this step and move on confirming **git** installation and create a new repository named **wdd130** which is outlined in the course content.

**Instructions**

**Sign Up for a free GitHub Account**

1. In your browser, visit the [github.com](https://github.com/) site.
2. Click the **Sign Up** button.
3. Enter an email address and click **Continue**.
4. Create a **password** that has to be at least 15 characters or 8 characters with a number and lowercase letter. Then select **Continue**.
5. Enter a **username** all in lowercase. Use a professional name since this username will show up in the domain of your web projects. You might show these links to future prospective employers. If you get a red '❌' to the left of your username, then someone else has already used that username and you need to choose a different one.
6. Record your username and password in a secure, logical location.
7. Enter 'n' for the next step unless you want email from them.
8. At this point it wants to make sure you are human, so it has you solve a few simple puzzles. Click **Start** puzzle and follow the directions. Then click **Create Account**.
9. It will send a launch code to the email you listed. Use that launch code and enter it.
10. You can skip any personalization questions by selecting the Skip Personalization link near the bottom.

Demonstration: 🎦 [Setting Up a GitHub Account](https://youtu.be/eMbF5kYqPt0)

<https://byui-cse.github.io/wdd130-ww-course/week01/setup-github-pages.html>

**Setup: GitHub Pages**

**Overview**

In this course, you will be publishing your work to your own website which is freely served through GitHub Pages. Your peers, graders, and the instructor will then be able to access your site for collaboration and feedback. You have already established a GitHub account and installed git technology so that you can commit and push/upload your work to your GitHub repository. We will be doing that inside of VS Code.

"GitHub Pages is a static site hosting service designed to host your personal, organization, or project pages directly from a GitHub repository" - [docs.github.com](https://docs.github.com/en/pages/getting-started-with-github-pages/about-github-pages)

**Instructions**

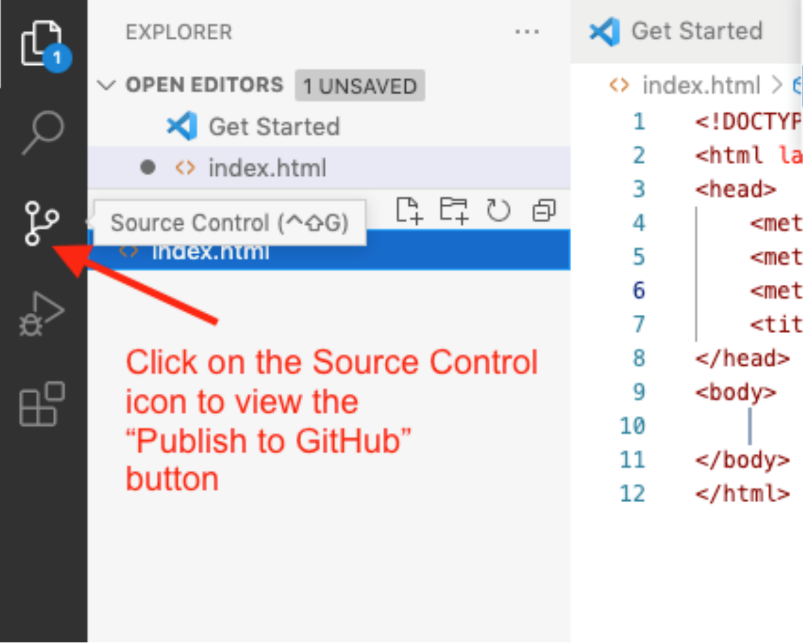
**Publish the wdd130 folder to GitHub**

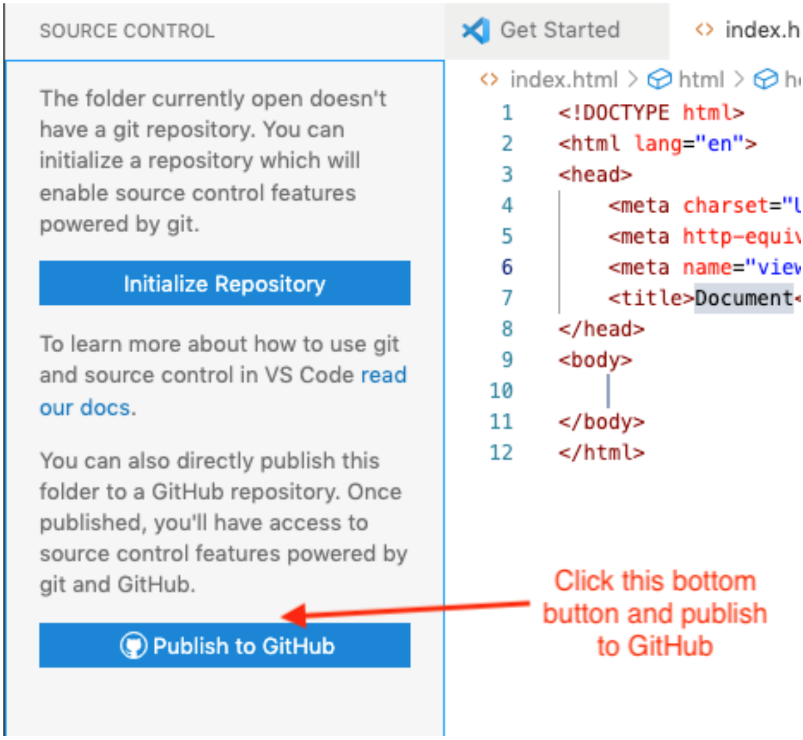
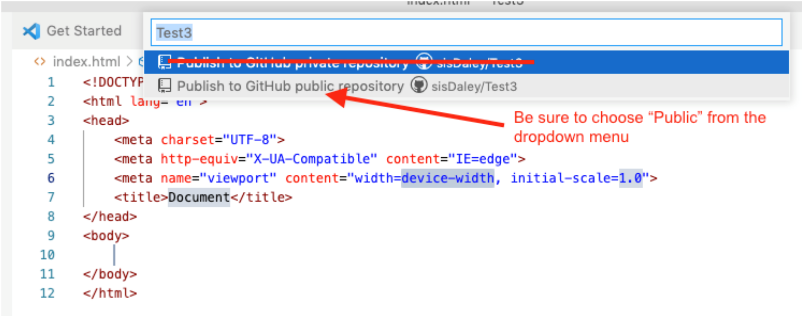
1. In VS Code, **open your wdd130 folder** if it is not already open.

The system may prompt you to trust the file source. Click **Trust** to continue.

1. Select the **Source Control** icon in VS Code's Activity Bar.

By default, the **Activity Bar** is found on the left-hand side of the screen and will have up to five icons. If the Activity Bar is not visible, click **View -> Appearance -> Activity Bar** to turn it visible.

Figure 1: Source Control Icon in VS Code's Activity Bar

1. Click the blue button that says **Publish to GitHub**. If you don't see this button, make sure you have all other git repo folders closed.Figure 2: Publish to GitHub Button in the Source Control panel
2. Click **Allow** if you get a message saying "*The extension 'GitHub' wants to sign in using GitHub.*"
3. If you are prompted for an authorization for Visual Studio Code to Access GitHub, click **Continue**.
4. If you are asked to allow the page to open "Visual Studio Code", click **Allow**.
5. You may also be asked to allow an extension to open this URI, click **Open**.
6. VSCode will choose the folder you have open, the wdd130 folder as the the repository and give you the option to "Publish to GitHub public repository". Choose the one that says **"public"**, not private. This will make a new public repository in our GitHub account.Figure 3: Publish to GitHub public repository
7. It will show a list of the files in the wdd130 folder. They will be selected by default so just click **OK** to include the files in the new repository.
8. You may get a prompt to "Authorize Git Credential Manager" here. Click **Authorize**.
9. If you get a message asking if you'd like to periodically run fetch you can say "Yes", but it is not necessary.

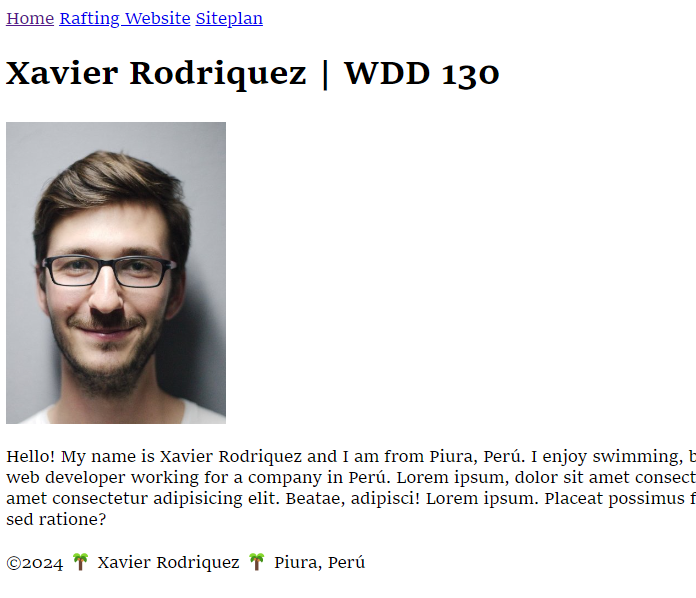
If you look at your GitHub account on github.com, a new repository named wdd130 will be there along with any files or folders that were in your local wdd130 folder. This local folder is now connected to the remote repository on GitHub and is a repository itself.

Demonstration: 🎦 [Setting up VSCode to push to a GitHub repository](https://youtu.be/mrGMxZkkIzg)

**Enable GitHub Pages**

1. Login to your [GitHub](https://github.com/) account if needed.
2. Open your **wdd130** repository.
3. Click **⚙️ Settings** in the repository menu.
4. Click **Pages** in the submenu that appears on the right of the settings panel.
5. Under the **Build and deployment** section of GitHub Pages, select the **main** branch.  
   Leave the next pull down menu with the default value of **📁 / root**.

If your branch is named **master** instead of **main**, that is OK.

1. In the top section, a message should display something like **"Your site is live at https://*yourgithubusername*.github.io/wdd130"**. This is your course home page URL. Note and bookmark this URL link for quick access in your browser.
2. That is all that needs to be done to enable GitHub Pages site publishing. After a few minutes, **test** the URL. Your course home page should open, rendered in the browser.Figure 1: Example Student Course Home Page Rendered in the Browser
3. **🧪 Test** the content of your page using this [Page Evaluation](https://byui-cse.github.io/wdd130-ww-course/grader/w01-homepage.html) tool.
4. **Share** your GitHub Pages wdd130 URL with a family member or friend.
5. Remember to **Bookmark** or **Favorite** your GitHub Pages enabled wdd130 URL.

Demonstration: 🎦 [GitHub Pages Setup](https://youtu.be/_jzY3dryS5A)

**Submission**

1. After you have completed the work, return to I-Learn to submit your GitHub Pages URL which will have the following format:

<https://byui-cse.github.io/wdd130-ww-course/week01/prove-html-start.html>

**Building Your Course Home Page**

**Overview**

This assignment applies your learning by having you create a basic home page for the course in HTML. You will maintain this home page throughout the course with updates, links to assignments, and other enhancements.

**Instructions**

1. Create a course folder on your computer named "**wdd130**" (lower case, no spaces).
2. In VS Code, open that wdd130 folder using **File -> Open Folder**.
3. Create a new file in that wdd130 folder named "**index.html**". This will be your course home page.

"index" is a standard page name on web servers that will load by **default** unless changed by the server administrator. This page will load if only the folder is referenced in a URL. Try it. In a browser window, go to ibm.com and then go to ibm.com/index.html. The same page loads.

1. In your index.html file, write the HTML markup to build the **basic HTML page anatomy** including:

For a reminder on the purpose of these elements, refer to the Learning Activity [An Introduction to HTML](https://byui-cse.github.io/wdd130-ww-course/week01/prepare-html-introduction.html)

* + the **document type**,
  + the **html** element with language attribute,
  + the **head** element within the html element's opening and closing tags, and
  + the **body** element within the html element and after the head.

Check Your Work

1. Within the **head** element, include the following from the required meta information:

Review the Learning Activity [An Introduction to HTML](https://byui-cse.github.io/wdd130-ww-course/week01/prepare-html-introduction.html) for details on the elements contained in the <head> element.

* + Meta charset attribute
  + Meta viewport element
  + Title element

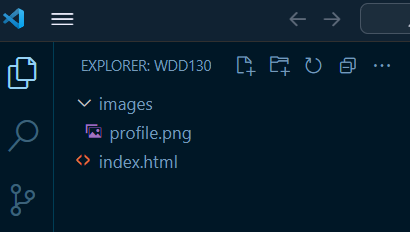
1. Set the **title** content to **[Your Full Name] | WDD 130** where [Your Full Name] is replaced with your actual, preferred full name.Check Your Work
2. Within the **body** element, include three, semantic, child elements:
   * **header**
   * **main**
   * **footer**

to create the major sections of the page.

1. Within the **header** element, add a **nav** element with three anchor tag **a** elements with the following **href** attribute values. Two of the links are for future assignments
2. **<header>**
3. **<nav>**
4. **<a href="#">Home</a>**
5. **<a href="wwr/">Rafting Website</a>**
6. **<a href="wwr/site-plan-rafting.html">Rafting Site Plan</a>**
7. **</nav>**
8. **</header>**
9. **<main>**
10. **</main>**
11. **<footer>**

**</footer>**

The # value for the Home page href anchor link refers to the current page and will not reload the page. This is the behavior and structure that we want.

1. Within the **main** element:
   * add a **h1** heading element that contains **Your Full Name | WDD 130**
   * add an **[img](https://developer.mozilla.org/en-US/docs/Web/HTML/Element/img" \t "_blank)** element with the following attributes:
     + **src**: The src specifies the URL of the image file. It consists of the path and the file name with its extension. Use the following path: **src="images/profile.\_\_\_"** where \_\_\_ will be your file extension.
     + **alt**: The alt attribute is required as part of accessibility to provide alternative text to display when the image is not rendered by the browser. Go ahead and include your name in the alternative text.
     + **width**: Set the layout initial width of the image to **200**.
   * add a **p** paragraph element that contains information that you want to share about yourself.
2. To support the image reference, add a new folder named "**images**" to your wdd130 folder.Figure 1: Example File Structure
3. Copy or move a **profile picture** file of yourself into this **images** folder. The acceptable [image file types](https://developer.mozilla.org/en-US/docs/Web/Media/Formats/Image_types) for this assignment include:
   * png
   * webp
   * jpg
4. Rename the image file **profile.** and, of course, use your image file extension.Check Your Work

Demonstration: 🎦 [Adding an Image](https://video.byui.edu/media/t/1_yux5rj8f)

1. Within the **footer** element, add a **p** paragraph element. The paragraph should contain the following:
   * The copyright symbol ©️ and a copyright year.

To display the copyright symbol, use the [HTML entity](https://www.w3schools.com/html/html_symbols.asp) **&copy;** or use a built-in emoji.

**Built-in Emoji Menu**  
Windows: Hold the 🪟window key and press period (.)  
macOS: Hold down the following three keys at the same time: **Command+Control+Spacebar** or press the **fn** key.

* + Your name.
  + Your state or country.

Each of these items in the footer will be separated by syntax of your choice.Check Your Work

1. Check your rendered page by right mouse clicking on the index.html file name in the Explorer panel and selecting **Open with Live Server** or by using the menu item at the bottom of VS Code. Live Server will open up the page in your default browser.

**Complete Example**

**<!DOCTYPE html>**

**<html lang="en-US">**

**<head>**

**<meta charset="utf-8">**

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

**<title>Xavier Rodriquez | WDD 130</title>**

**</head>**

**<body>**

**<header>**

**<nav>**

**<a href="#">Home</a>**

**<a href="wwr/">Rafting Site</a>**

**<a href="wwr/site-plan-rafting.html">Rafting Site Plan</a>**

**</nav>**

**</header>**

**<main>**

**<h1>Xavier Rodriquez | WDD 130</h1>**

**<img src="images/profile.jpg" alt="Hello, my name is Xavier Rodriquez" width="200">**

**<p>Hello! My name is Xavier Rodriquez and I am from Piura, Perú. I enjoy ... </p>**

**</main>**

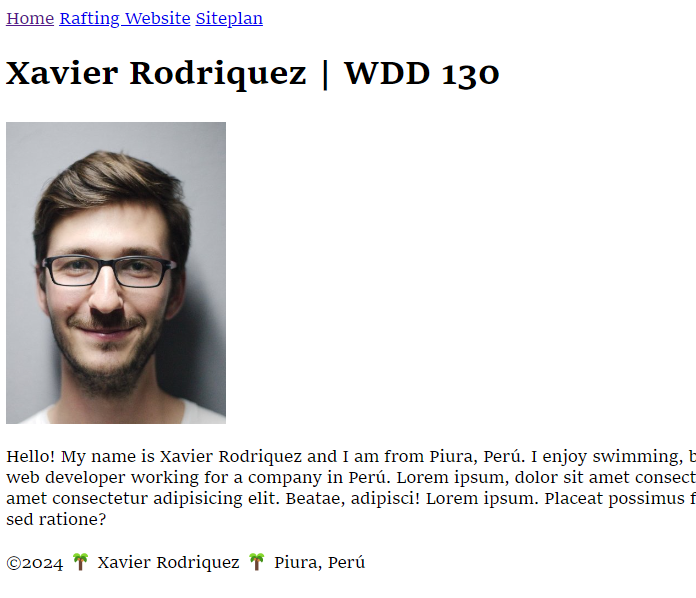
**<footer>**

**<p>©️2024 🌴 Xavier Rodriquez 🌴 Piura, Perú</p>**

**</footer>**

**</body>**

**</html>**

Figure 1: Screenshot of Example Student Course Home Page

1. **Validate** your HTML by running the **Web Developer** extension in your browser.
   * Render the page on your browser using the **Live Server** tool in VS Code.
   * Using the Web Developer extension, click **Tools -> Validate Local HTML** to get an HTML validation report.
   * **Correct** any issues that you observe after testing the page.

Feel free to post your questions to MS Teams.

1. Continuously **save** your work.

It is recommended that you turn on Auto Save in VS Code. Go to **File -> Preferences -> Settings** and then type "*Auto Save*" in the search bar at the top of the panel. Use the pull-down menu to select **afterDelay**. Set the **Files: Auto Save Delay** field to something like 1000 ms (1 second).  
📑[Save / Auto Save](https://code.visualstudio.com/docs/editor/codebasics#_save-auto-save) - code.visualstudio.com

**Submission**

1. There is nothing to submit at this point. You will be committing and pushing/uploading your work to your remote wdd130 repository on GitHub for publication in the next activity using a service called GitHub Pages.

<https://byui-cse.github.io/wdd130-ww-course/week01/prepare-images.html>

**Working with Web Images**

**Overview**

The multimedia web uses images. You will work with many different image formats and some of those image types are better suited for the web. This activity explores the images for the web and how to optimize images to be smaller, supporting smaller page weights thus download time.

"The older formats like PNG, JPEG, GIF have poor performance compared to newer formats like WebP and AVIF, but enjoy broader "historical" browser support. The newer image formats are seeing increasing popularity as browsers without support become increasingly irrelevant (i.e. have virtually zero market share)." - [MDN](https://developer.mozilla.org/en-US/docs/Web/Media/Formats/Image_types)

**Prepare**

* Watch: 🎦 [Image File Types (JPG, PNG, SVG, GIF) - Web Design Tutorial](https://youtu.be/Zx0CacsiDZ4)
* Read: 📄 [Image file type and format guide](https://developer.mozilla.org/en-US/docs/Web/Media/Formats/Image_types) - MDN
* Watch: 🎦 [Image Optimization for Web Pages](https://youtu.be/85X16eLvNYY)

**Basic Steps of Image Optimization**

Often the original photos that we work with are very large in terms of dimensions AND file size. Large photos need to be scaled down and optimized to a size that is actually needed on the page. This size is determined by the design and layout of the page. Software can be used to crop, reduce, and optimize the images. These are the basic steps supported by most photo editing packages:

1. **Crop** the original image to the critical content keeping in mind design principles, like the [rule of thirds](https://www.slrlounge.com/glossary/rule-of-thirds-definition/), and the site or section purpose.
2. **Resize** the image to the maximum size that you actually need in the design of your site. That image will then be saved in smaller formats, possibly with slightly different design to match the design and layout in smaller views, meaning, smart phone, tablet.
3. **Reduce the quality** of the image to an acceptable level, thus reducing its file size.

Optimizing images is an easy way to improve page performance without requiring a significant time investment. We will cover more techniques for handling responsive images in future courses.

**Activity Instructions**

**Optimize an Image**

1. In your browser, navigate to the online application, [Squoosh.app](https://squoosh.app/" \t "_blank).
2. Click on the button in the middle of the page and select an image from your computer.
3. In the **Edit** panel on the right, toggle **Resize** and change the width to ~200 pixels.
4. Under **Compress**, change the file type to **WebP**.
5. Under **Compress**, change the **Quality** to ~70%.
6. Note the overall change in file size displayed in those panels.
7. Click the download button (bottom right) to save the image to your computer.