**File Paths – Quick Notes**

* **Definition of File Paths:**
  + A file path specifies the unique location of a file or folder on a computer.
  + Similar to giving someone detailed directions (e.g., “go to London, then Westminster, then Page Street”).
* **Types of File Paths:**
  + **Absolute File Paths:**
    - **Description:**  
      Start from the root of the file system (e.g., C:\ on Windows or Macintosh HD on Mac).
    - **Usage:**  
      Useful when you need to specify an exact location regardless of your current folder.
    - **Example:**
      * Windows:

plaintext

CopyEdit

C:\Users\YourName\Projects\Images\cat.png

* + - * Mac:

plaintext

CopyEdit

/Macintosh HD/Users/YourName/Projects/Images/cat.png

* + **Relative File Paths:**
    - **Description:**  
      Defined relative to the location of the current file (e.g., your index.html).
    - **Usage:**  
      More flexible for web development—if you move your project folder, these paths remain valid.
    - **Example:**
      * If your index.html is in the Project folder and the image is in an Images subfolder:

html

CopyEdit

<img src="./Images/cat.png" alt="Cat Image">

* **Special Characters in Relative File Paths:**
  + **./ (Single Dot):**
    - Means “current directory.”
    - *Example:*  
      Accessing a file in the same folder as index.html:

html

CopyEdit

<img src="./rabbit.png" alt="Rabbit">

* + **../ (Double Dot):**
    - Means “go up one level” (to the parent directory).
    - *Example:*  
      If index.html is in a subfolder and you need to access a file in the parent folder:

<img src="../dog.png" alt="Dog">

* **Practical Exercise Example:**
  + **Scenario:**  
    You have an index.html and multiple images stored in various nested folders.
  + **Task:**  
    Display images for different animals (rabbit, cat, dog, fish, bird) by writing the correct relative paths.
  + **How-To:**
    - **Rabbit:**  
      Located in the same folder as index.html.

<img src="./rabbit.png" alt="Rabbit">

* + - **Cat:**  
      Located in a subfolder (e.g., Folder3) relative to index.html.

<img src="./Folder3/cat.png" alt="Cat">

* + - **Dog:**  
      Located in a folder outside the current one. Use ../ to move up a level.

<img src="../dog.png" alt="Dog">

* + - **Fish and Bird:**  
      Use similar strategies:
      * Fish might be in Folder1:

<img src="../Folder1/fish.png" alt="Fish">

* + - * Bird might be nested further (e.g., in Folder1/Folder2):

html

CopyEdit

<img src="../Folder1/Folder2/bird.png" alt="Bird">

* **Tips for Working with File Paths:**
  + Use VS Code’s autosuggest to reduce typing errors.
  + A broken image icon usually indicates an incorrect file path.
  + Practice by experimenting with different relative paths to understand how ./ and ../ work.
* **Next Steps in Web Development:**
  + Later lessons will extend file path usage to include linking different HTML pages, enabling the creation of multi-page websites.

### Multi-Page Websites & File Paths – Quick Notes

#### 1. Multi-Page Website Structure

* **Project Folder Setup:**
  + Contains a main file (typically **index.html**).
  + May include additional HTML files (e.g., **about.html**, **contact.html**) in the same folder or in subfolders (like a **public** folder).
  + Contains asset folders (like **assets/images**) for images and other resources.
* **Purpose:**
  + Allows you to create a website with multiple interconnected pages (Home, About, Contact, etc.).

#### 2. File Paths & Linking Pages

* **Relative File Paths:**
  + **./ (Dot Slash):** Refers to the current directory.
    - Example: In **index.html**, linking to **about.html** would be <a href="./about.html">About</a>.
  + **../ (Double Dot):** Means “go up one folder.”
    - Important: Use it only when you need to navigate out of the current directory.
* **Linking Pages:**
  + Use the <a> (anchor) tag with the **href** attribute to connect different pages.
  + Example: If **contact.html** is in a subfolder called **public**, the link is:

<a href="./public/contact.html">Contact Me</a>

#### 3. Working with Images & File Paths

* **Image Element:**

<img src="./assets/images/cat.png" alt="Cat" />

* **Clickable Images:**
  + You can wrap an <img> tag inside an <a> tag to make the image a link.
  + Example:

<a href="./public/about.html">

<img src="./assets/images/cat.png" alt="Cat linking to about page" />

</a>

#### 4. Quiz Review: Correct File Path for an Image

* **Scenario:** From **index.html**, show **cat.png** located inside **assets/images**.
* **Incorrect Options:**
  + Using .. (double dots) navigates up a directory incorrectly.
  + Using href in an image element (should use src).
  + Using an anchor tag instead of an image tag.
* **Correct Answer:**

<img src="./assets/images/cat.png" alt="A descriptive text for cat image" />

#### 6. Key Concepts Recap

* **File Paths:** Understanding how to navigate directories is critical for linking files correctly.
  + **Relative Paths** are essential for local development.
* **Element-Specific Attributes:**
  + **href** is used with anchor tags for hyperlinks.
  + **src** is used with image tags to load images.
* **Accessibility:** Always include **alt** text with images to improve accessibility for visually impaired users.

**HTML Boilerplate – Quick Notes**

* **Definition & Purpose:**
  + A starting template for every new HTML document.
  + Provides a consistent structure, similar to how a letter has a heading, body, and closing.
* **Key Components:**
  + **Doctype Declaration:**
    - Specifies the HTML version.
    - For HTML5:

<!DOCTYPE html>

* + **<html> Element:**
    - Acts as the root of the document.
    - Contains all other HTML elements.
    - Includes the lang attribute to define the language (e.g., lang="en" for English).
  + **<head> Element:**
    - Contains meta-information and data not visible on the webpage.
    - Common elements in <head>:
      * **Character Set Meta Tag:**
        + Ensures characters display correctly.
        + Example:

<meta charset="UTF-8">

* + - * **Title Tag:**
        + Sets the title displayed in the browser's tab.
        + Example:

<title>My Website</title>

* + - * **Other Meta Tags:**
        + *X-UA-Compatible*: For IE compatibility (often omitted now due to deprecation).
        + *Viewport*: Instructs the browser on how to control the page’s dimensions and scaling.

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

* + **<body> Element:**
    - Contains all the content visible to users (headings, images, links, etc.).
    - This is where you add your actual webpage content.
* **Structure Analogy:**
  + Think of the HTML file as a burger:
    - **Buns:** <html> opening and closing tags.
    - **Head:** Like the top layer with metadata (the "address" part of a letter).
    - **Body:** The main content (like the meat and toppings inside a burger).
  + Proper **nesting** and **indentation** help keep your code organized and readable.
* **VS Code Shortcut:**
  + In VS Code, type ! in a new .html file and press Enter.
  + This auto-generates the HTML boilerplate, saving time.
* **Sample Boilerplate Code:**

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

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

<title>My Website</title>

</head>

<body>

<!-- Your content goes here -->

</body>

</html>

### Web Hosting FREEusing github

#### 1. What is Web Hosting?

* **Definition:**
  + The process of making your website available on the Internet.
  + Files (HTML, images, etc.) are stored on a web server that's connected 24/7.
* **Why It Matters:**
  + Unlike local development (files on your computer), hosting lets anyone worldwide access your website.

#### 2. Local Development vs. Web Hosting

* **Local Development:**
  + Your website files reside on your computer.
  + Only you can view them unless you manually share the files.
* **Web Hosting:**
  + Upload your files to a web server.
  + Makes your website accessible globally.

#### 3. Hosting Your Website with GitHub Pages

* **GitHub Account Setup:**
  + Sign up or log in at [GitHub.com](https://github.com).
* **Create a New Repository:**
  + Click the plus icon (➕) on the top right.
  + Name your repository (e.g., html-portfolio).
  + Ensure the repository is **public** (required for free GitHub Pages hosting).
  + (Optional) Add a description and a README file.
* **Upload Your Website Files:**
  + Navigate to your local HTML Portfolio Project folder.
  + **Important:**
    - Your home page must be named exactly index.html (lowercase) in the root directory.
    - Drag and drop the **contents** of your project folder (not the folder itself) into GitHub.
  + Commit the changes to upload all files and folders (e.g., images, public HTML pages).

#### 4. Setting Up GitHub Pages

* **Navigate to Settings:**
  + Inside your repository, click on **Settings**.
* **Configure GitHub Pages:**
  + Find the **Pages** section.
  + Under **Branch**, change the option from "none" to your main branch.
  + Click **Save**.
* **Go Live:**
  + GitHub will process your site (this can take 1–10 minutes).
  + Refresh the page until you see a notification that your site is live.
  + Click **Visit site** to view your live website.
  + Share your URL with friends and family.

#### 5. Troubleshooting & Tips

* **Filename Accuracy:**
  + Verify that index.html is spelled correctly and located in the root.
* **Asset Verification:**
  + Ensure that folders (e.g., images) and other HTML files are correctly uploaded.
* **Error Messages:**
  + If you see errors like "No server is currently available," wait a moment and refresh.
* **Next Steps:**
  + The next lesson will cover CSS for styling, which will transform your basic HTML site into a beautifully styled one.

These notes provide a concise overview of the steps and concepts involved in hosting your website using GitHub Pages.