**The big three: HTML5, CSS and JS**

Introduction to HTML5, CSS, and JavaScript

Welcome to the world of web development. In this tutorial, you will get an introduction to three core technologies used to build modern websites:

* **HTML5** (Hypertext Markup Language)
* **CSS** (Cascading Style Sheets)
* **JavaScript**

These technologies work together to create interactive, well-designed, and functional websites. Let's start with the basics:

1. **HTML5: Structure of the Web**

HTML is the foundation of web pages. It defines the structure and content of a webpage, such as headings, paragraphs, links, and images.

**Basic HTML Example:**

<!DOCTYPE html>  
<html lang="en">  
<head>  
  <meta charset="UTF-8">  
  <meta name="viewport" content="width=device-width, initial-scale=1.0">  
  <title>HTML5 Example</title>  
</head>  
<body>  
  <header>  
    <h1>Welcome to HTML5</h1>  
  </header>  
  <section>  
    <p>This is a paragraph in HTML5.</p>  
    <a href="https://www.example.com" target="\_blank">Click me</a>  
  </section>  
</body>  
</html>

In this example:

* <!DOCTYPE html> specifies the document type (HTML5).
* <header> contains a header section.
* <section> contains content like a paragraph and a link.

2. **CSS: Styling the Web**

CSS is used to control the layout and appearance of web pages. You can adjust colors, fonts, spacing, and more to make your site look appealing.

**Basic CSS Example:**

/\* CSS to style the webpage \*/  
body {  
  font-family: Arial, sans-serif;  
  background-color: #f0f0f0;  
  margin: 0;  
  padding: 0;  
}  
  
header {  
  background-color: #333;  
  color: white;  
  padding: 10px;  
  text-align: center;  
}  
  
a {  
  color: #007BFF;  
  text-decoration: none;  
}  
  
a:hover {  
  color: #0056b3;  
}

In this example:

* The body tag controls the general layout and background color.
* The header tag styles the header area.
* The a tag styles links, including the hover effect when you mouse over the link.

3. **JavaScript: Making the Web Interactive**

JavaScript allows you to create dynamic and interactive elements on the web page, such as buttons, forms, and animations.

**Basic JavaScript Example:**

<!DOCTYPE html>  
<html lang="en">  
<head>  
  <meta charset="UTF-8">  
  <meta name="viewport" content="width=device-width, initial-scale=1.0">  
  <title>JavaScript Example</title>  
</head>  
<body>  
  <h1>JavaScript Example</h1>  
  <button onclick="changeText()">Click Me!</button>  
  <p id="message"></p>  
  
  <script>  
    function changeText() {  
      document.getElementById("message").innerHTML = "Hello, JavaScript is working!";  
    }  
  </script>  
</body>  
</html>

In this example:

* The <button> element calls the changeText() function when clicked.
* The changeText() function updates the content of the <p> element with the id="message".

---------------------------------------------------------------

**Introduction to Git**

Welcome to the world of **Git**, a powerful version control system that allows developers to track changes in their code, collaborate with others, and manage software projects efficiently. In this guide, we’ll introduce you to the basic concepts of Git, demonstrate common commands, and provide resources for further learning.

1. **What is Git?**

Git is a distributed version control system, meaning it helps developers manage different versions of code in a project. It tracks changes, allows for collaboration, and makes it easier to revert to previous versions if something goes wrong.

With Git, you can:

* Track changes to files over time.
* Collaborate with others on a project by merging changes.
* Revert or recover previous versions of your project.

2. **Basic Git Commands**

Here’s a brief overview of some common Git commands that you’ll use in your workflow:

* **Create a new Git repository:**  
  To start tracking a project with Git, navigate to your project directory and run git init. This initializes a new Git repository in your project folder.
* **Clone an existing repository:**  
  If you want to work on an existing project, you can clone it with the command git clone https://github.com/username/repository-name.git.
* **Check the status of your repository:**  
  To see which files have been changed, added, or are untracked, use git status.
* **Add files to the staging area:**  
  Before committing changes, you need to add files to the staging area. You can do this with git add <file-name> (for a specific file) or git add . (to add all files).
* **Commit your changes:**  
  Once your changes are staged, commit them with a message using git commit -m "Your commit message here".
* **Push changes to a remote repository:**  
  After committing, you can push your changes to a remote repository (like GitHub or GitLab) with git push origin main (assuming the main branch).
* **Pull updates from a remote repository:**  
  To fetch and merge changes from a remote repository into your local branch, use git pull origin main.
* **View commit history:**  
  To see the history of your commits, use git log.

3. **Git Workflow Example**

Let’s walk through a basic workflow using Git:

1. **Initialize a repository:**  
   If you’re starting a new project, navigate to your project folder and run git init.
2. **Create and edit files:**  
   Add some code or content to your project.
3. **Stage the changes:**  
   After editing your files, run git add . to stage all files for commit.
4. **Commit the changes:**  
   Save your progress with a commit by running git commit -m "Added initial code".
5. **Push the changes:**  
   If you’re working with a remote repository (e.g., on GitHub), push your changes by running git push origin main.
6. **Pull updates from others:**  
   If you're collaborating, you'll want to pull changes from the remote repository using git pull origin main.

**Steps to Work with GitHub and Git Bash**

Create GitHub folder:

mkdir GitHub

Change directory:

cd GitHub

See the folder:

(The highlighted folder confirms that the folder has been created.)

ls -la

Clear the terminal:

clear

Initialize a bare repository:

(Use --bare to create a repository that others can clone. Without this, cloning the folder won't work.)

git init --bare

Clone the repository:

git clone ../GitHub

Create a new file:

touch filename

Edit a file using Git Bash:

vim nav.html

* Press i to enter Insert Mode.
* Type your content.
* Press Esc to exit Insert Mode.

Delete lines in vim:

to delete two lines (adjust the number for the desired count).

d2↓

Save and exit in vim:

:wq!

View file contents:

Example: cat index.html

cat filename

Check the status of files:

(Red text indicates untracked or modified files.)

git status

Stage files:

git add .

Check status again:

(Green text indicates that files are staged.)

git status

Commit the changes:

git commit -m "your commit message"

Push changes:

git push

View commit logs:

(Shows the history of committed changes.)

git log

Unstage a specific file:

Example: git restore --staged 3 prevents the file from being pushed.

git restore --staged <file>