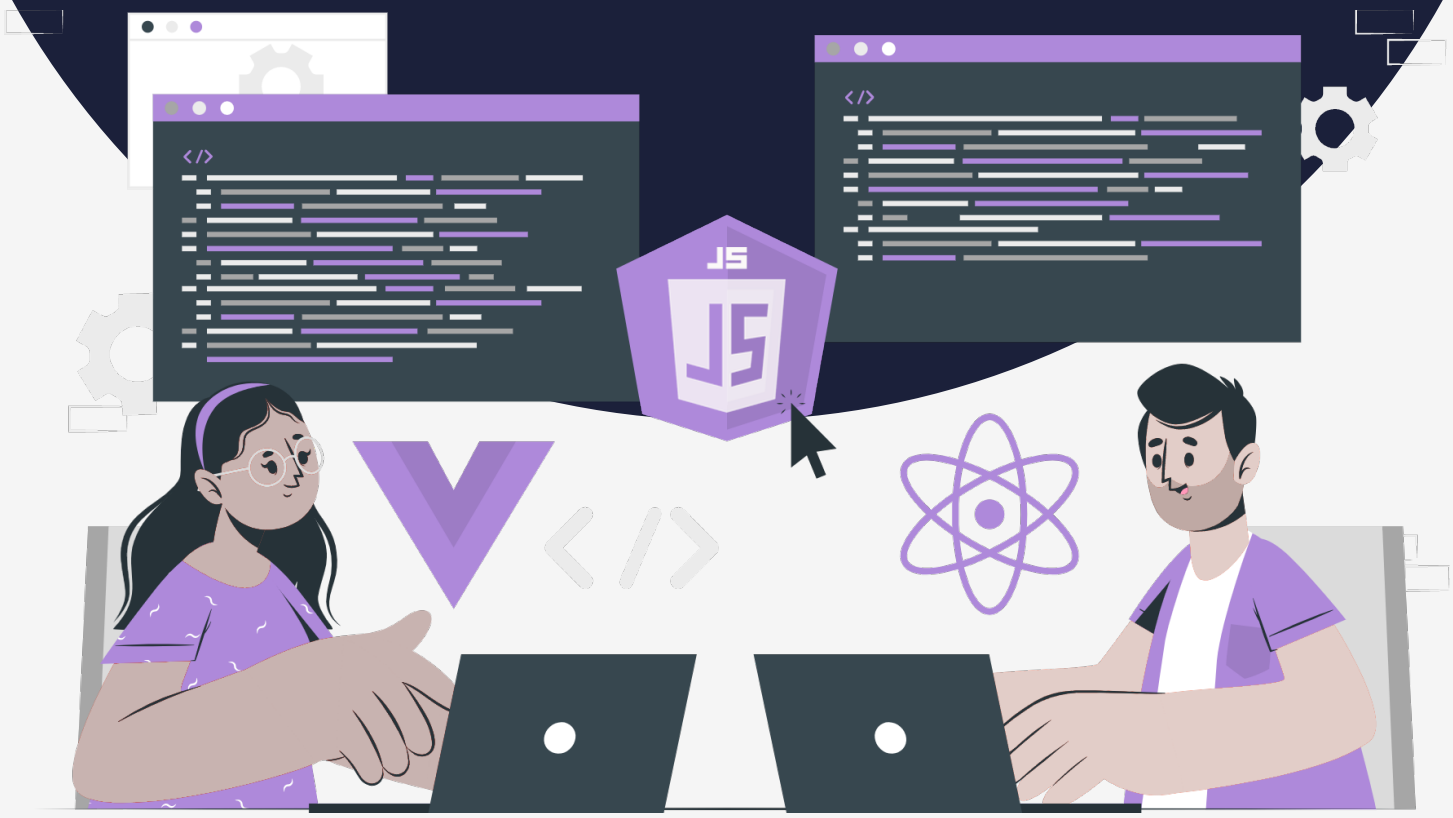
# Lesson:

# Creating Account on GitHub and Creating a repository on GitHub




PW SKILLS

# Topics Covered

- Creating Account on GitHub
- What is Repository
- Creating a Repository

**Creating Account on GitHub**

GitHub is a web-based platform used for version control and collaboration on software development projects. It allows developers to store and manage their code repositories, share code with others, and collaborate on projects with other developers around the world. In order to start using GitHub, you will need to create an account.

Here are the steps to create an account on Github:

1. Go to the GitHub website at **www.github.com.** On the top right corner of the homepage, you will see a "Sign up" button. Click on it.
2. You will be directed to the sign-up page where you will be asked to enter your personal information. This includes your full name, a valid email address, and a password. Choose a strong password that is difficult to guess but easy for you to remember. Then click on the "Create an account" button.
3. After submitting the registration form, you will be taken to a page where you can choose your plan. GitHub offers both free and paid plans, depending on your needs. If you are just getting started, you can choose the free plan, which offers unlimited public repositories and basic collaboration features.
4. Next, you will be asked to verify your email address. GitHub will send you an email with a verification link. Click on the link to verify your account.
5. Once your account is verified, you can start customizing your profile. You can add a profile picture, a bio, and links to your website and social media profiles. This will help other developers find and connect with you on the platform.
6. Congratulations! You have successfully created an account on GitHub. You can now start exploring the platform, creating and collaborating on projects, and contributing to the open-source community.

**What is Repository**

A Git repository is a digital storage space where developers can store their source code and other related files, as well as track changes, and collaborate with other team members. Git is a distributed version control system, which means that multiple developers can work on the same codebase simultaneously, and changes can be easily tracked, merged, and managed.

Here are some key components of a Git repository:

- **Working directory:** This is the directory on your computer where you keep your files, including the source code you want to manage with Git.
- **Index:** Also known as the staging area, this is where you can prepare your changes before committing them to the repository. It allows you to selectively choose which changes you want to commit.
- **Commit:** A commit is a snapshot of the changes you have made to your files at a specific point in time. When you commit your changes, you create a new version of the code that can be easily tracked and managed.

- **Branches:** Git allows you to create multiple branches of your codebase, each representing a different version or set of changes. This allows multiple developers to work on different parts of the codebase without interfering with each other's work.
- **Remote:** A remote repository is a copy of your repository stored on a remote server, such as GitHub, GitLab, or Bitbucket. It allows you to share your code with other developers and collaborate on projects.

**Creating a Repository**

Creating a repository on GitHub is a simple process that involves a few basic steps. A repository is a place where you can store and manage your code, as well as collaborate with other developers.

Here are the steps to create a repository on GitHub:

1. Log in to your GitHub account and click on the "+" button in the top right corner of the screen.
2. From the dropdown menu, select "New repository".
3. On the "Create a new repository" page, you will be asked to enter the following information:

- **Repository name:** Choose a name that is descriptive and easy to remember.
- **Description:** Write a short description of your repository to help other developers understand what it is about.
- **Public or private:** Choose whether you want your repository to be public (visible to anyone) or private (visible only to you and the collaborators you choose).
- **Initialize this repository with a README:** Check this box if you want to create a README file for your repository. A README file is a text file that provides information about your project, such as installation instructions, usage guidelines, and contribution guidelines.

4. Once you have entered all the required information, click on the "Create repository" button. This will create a new repository on GitHub.
5. Or we can use git clone command that allows you to create a copy of a remote Git repository on your local machine. This command creates a local copy of the entire repository, including its history, branches, and tags.

Here's an example of how to use git clone:

```JavaScript
git clone https://github.com/username/repository.git
```

In this example, https://github.com/username/repository.git is the URL of the remote Git repository that you want to clone.

When you run git clone, Git will download the entire repository to your local machine, and create a new directory with the same name as the repository. The directory will contain all the files, folders, and version history of the remote repository.

6. You can now add files to your repository, create branches, and collaborate with other developers. To add files to your repository, you can either drag and drop them onto the repository page or use the command line interface (CLI) to push your files to the repository.

Here are some main Git CLI commands:

- **git init:** Initializes a new Git repository in the current directory. This creates a new .git subdirectory that contains all the files necessary for Git to track changes to your code.
- **git clone:** Copies an existing Git repository to your local machine. This downloads a complete copy of the remote repository, including all files and commit history, to your local machine.
- **git add:** Adds changes in the current directory to the staging area. This is the first step in committing changes to a Git repository. Changes must be staged before they can be committed.
- **git commit:** Commits changes in the staging area to the local repository. This creates a new commit with a unique identifier, timestamp, author, and commit message. The commit message should describe the changes being made.
- **git status:** Displays the status of the working directory and staging area. This shows which files have been modified, added, or deleted, and whether they are currently staged or not.
- **git push:** Pushes local changes to a remote repository. This sends all local commits that have not yet been pushed to the remote repository. This requires permission to push to the remote repository.
- **git pull:** Fetches changes from a remote repository and merges them into the local branch. This downloads any new changes from the remote repository and merges them with the local branch.
- **git branch:** Lists all branches in the local repository and highlights the current branch. This shows all branches in the local repository and indicates which branch you are currently on.
- **git checkout:** Switches between different branches or commits in the repository. This allows you to switch between different branches or commits in the repository.
- **git merge:** Merges changes from one branch into another branch. This combines changes from one branch into another branch, typically the current branch.

7. If you want to invite other developers to collaborate on your repository, you can go to the "Settings" tab of your repository page, and click on the "Collaborators" option. From there, you can add the usernames of the people you want to invite as collaborators, and set their access levels (read-only or read/write).