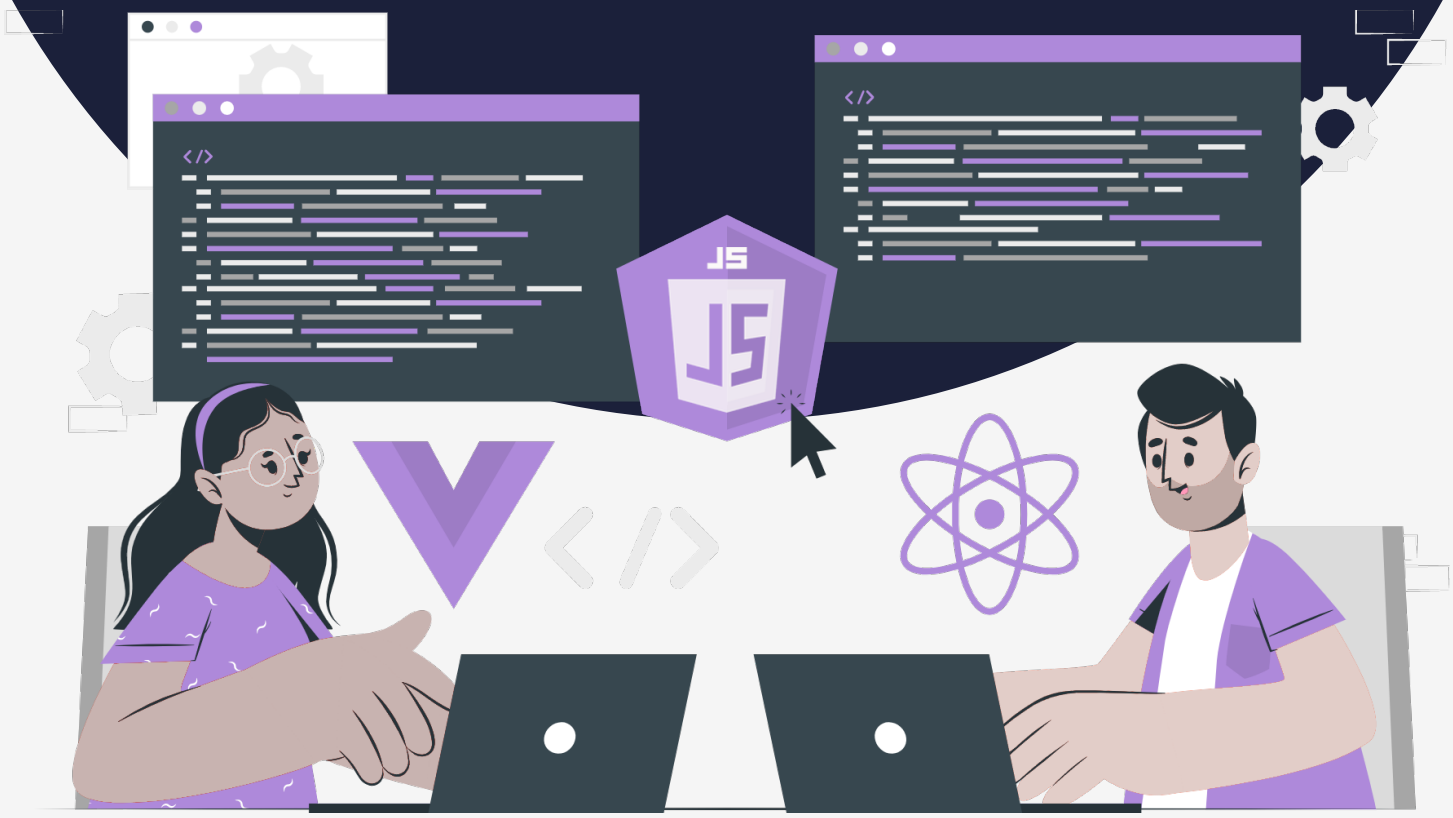
Lesson:

Push the file to GitHub - Start to end guide



Topics Covered

- A step-by-step guide to uploading your project to GitHub

A step-by-step guide to uploading your project to GitHub

- Open your project folder in VSCODE and press the " **Ctrl +** ` " key to open the terminal in VSCODE.
- Then check your proper **Path** in the terminal to initialize git.
- Type the **git init** command and press enter. Then the Git version control system creates an empty Git repository in the current directory. This initializes a new Git repository and sets up the necessary files and data structures needed to start tracking changes in your project.
- Make some code changes so that the changes can be tracked.
- Then you can check the **git status**. This is a Git command that shows the current status of the Git repository. When you run **git status** in the command line, Git will show you information about the files in your local repository, including
    - Which files have been modified since the last commit
    - Which files have been added to the staging area but not yet committed
    - Which files are currently in the staging area
    - Which branch are you currently on

The output of git status can help you understand what changes you've made to your project and what still needs to be committed or pushed to a remote repository. It's a useful command to run frequently to keep track of the status of your Git repository.

If there are any untracked files in your working directory, git status will also show them as untracked, which means they are not being tracked by Git and will not be included in commits or pushed to the remote repository. You can use git add to add these untracked files to the staging area so that they will be tracked by Git.

- Now add your files to the git repository using the git add command. git add is a Git command used to stage changes made to files in a Git repository.

When you modify a file in your project, Git doesn't automatically track those changes. Instead, you need to use git add to stage the changes and tell Git that you want to include them in the next commit. This is done by adding the changes to the staging area, which is a temporary storage area for changes that you want to commit.

You can add specific files to the staging area by using **git add** followed by the file name or path, for example,

```JavaScript
git add filename.txt
```

You can also add all changes in the current directory and its subdirectories by using **git add** . :

```
JavaScript
      git add .
```

Once you've added the changes you want to commit to the staging area, you can use the git commit command to create a new commit that includes those changes.

- After adding changes to the repository you have to use the **git commit** command to save the changes you have made to a git repository. When you use **git commit**, Git creates a new commit object with a unique identifier(a hash) that represents the changes you have made. This allows you to keep track of changes to your codebase over time and collaborate with others on the same project.

```
JavaScript
git commit -m " commit message "
```

This command creates a new commit with the changes that have been added to the staging area. The commit message should be a brief summary of the changes that have been made in this commit. The -m flag indicates that the commit message will be specified in the command line.

- After commit you can directly add remote origin but to push a local repository to GitHub, the official guide recommends that first we use **git branch -M mai**n command to change the **master** branch name to **main**.

```
JavaScript
git branch -M main
```

- **Now git remote add origin** is a Git command that used to add a new remote repository to your local git repository.

The origin is the default name conventionally given to the remote repository from which you cloned your local repository. However, it can be any name of your choice.

The syntax for using the git remote add origin command is as follows:

```
JavaScript
git remote add <remote_name> <remote_repository_URL>
```

Where:

- **<remote_name>** is the name you want to give to the remote repository. It can be any valid name you choose, but origin is typically used.
- **<remote_repository_URL>** is the URL of the remote repository you want to add.

For example, if you want to add a remote repository with the URL **https://github.com/your-username/your-repo.git** and name it origin, you can use the following command:

```
JavaScript
git remote add origin
https://github.com/your-username/your-repo.git
```

- And after setting origin last step is push the code to GitHub repository. For that **git push -u origin main** is a Git command is used to push the changes made in your local main branch to the **main** branch of the remote repository named **origin**.

The **-u** option is used to set the upstream branch for the **main** branch. When you set the upstream branch, Git will remember the remote branch you pushed to and the local branch that it corresponds to. This is useful for future pushes, as you can simply use **git push** without having to specify the remote and branch names each time.

The syntax for using the **git push -u origin** main command is as follows:

```
JavaScript
git push -u <remote_name> <local_branch_name>:<remote_branch_name>
```

Where:

- **<remote_name>** is the name of the remote repository you want to push your changes to.
- **<local_branch_name>** is the name of the local branch that you want to push changes from. In this case, it is main.
- **<remote_branch_name>** is the name of the branch in the remote repository that you want to push changes to. In this case, it is also main.

For example, if you want to push changes made in your local **main** branch to the **main** branch of the remote repository named **origin**, you can use the following command:

```
JavaScript
git push -u origin main
```

Once you have pushed your changes using this command, the changes will be reflected in the **main** branch of the remote repository.

By following this steps you can easily push your code to GitHub.