Experiment No: 6

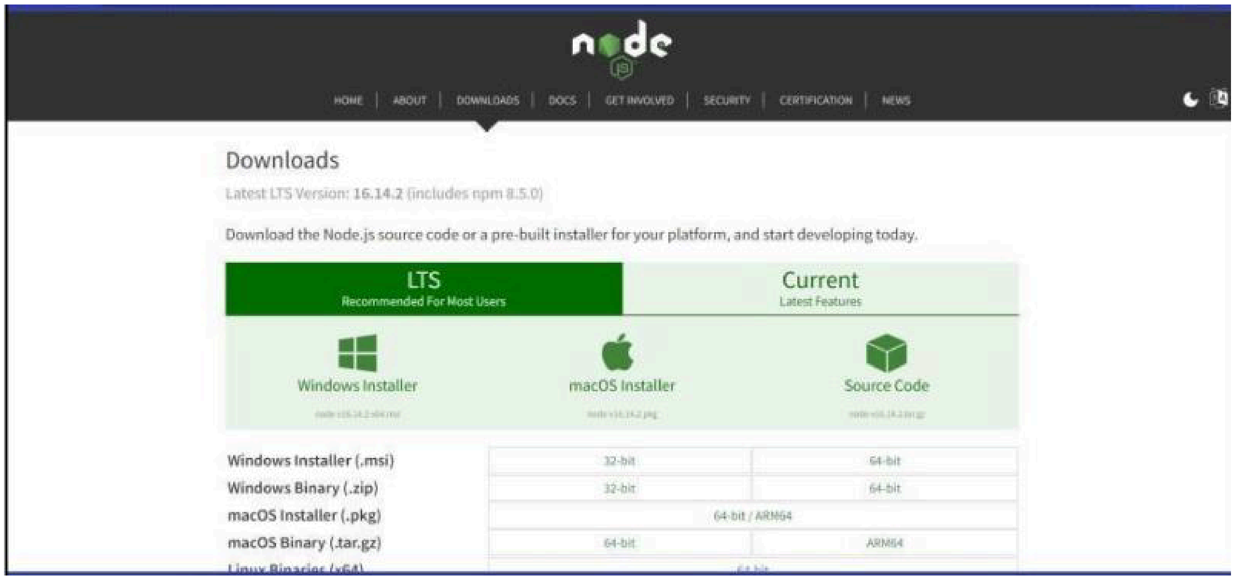
Aim: Building a simple website with typescript

Theory:

TypeScript is a natural extension of JavaScript that's used in many projects in place of JavaScript. However, not everyone knows how it actually works. TypeScript is a superset of the JavaScript language that lets us produce safe and predictable code. The code that it produces can be run in any JavaScript runtime. It provides us with type checks which JavaScript lacks. TypeScript isn't the solution to all the problems in JavaScript projects. We should know what TypeScript is good for. TypeScript is focused on developer productivity through static typing.

It makes JavaScript type system easier to work with. It provides us access control keywords and enhances the class constructor syntax. We can write TypeScript code with JavaScript or TypeScript-exclusive syntax. All the code goes through the TypeScript compiler, which is built to JavaScript. The combination of JavaScript and TypeScript provides us with a flexible combination of both. We can apply TypeScript features selectively. TypeScript stands in an unusual relationship to JavaScript. TypeScript offers all of JavaScript's features, and an additional layer on top of these: TypeScript's type system. For example, JavaScript provides language primitives like string and number, but it doesn't check that you've consistently assigned these. TypeScript does. This means that your existing working JavaScript code is also TypeScript code. The main benefit of TypeScript is that it can highlight unexpected behavior in your code, lowering the chance of bugs. This tutorial provides a brief overview of TypeScript, focusing on its type system

1) Install node from its website


HOME | ABOUT | DOWNLOADS | DOCS | GET INVOLVED | SECURITY | CERTIFICATION | NEWS

Downloads

Latest LTS Version: 16.14.2 (includes npm 8.5.0)

Download the Node.js source code or a pre-built installer for your platform, and start developing today.

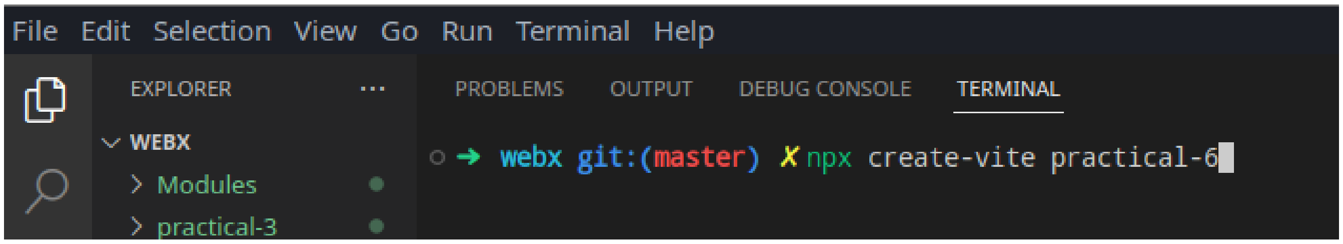
LTS
Recommended For Most Users
Current
Latest Features
Windows Installer
macOS Installer
Source Code

Windows Installer (.msi)	32-bit	64-bit
Windows Binary (.zip)	32-bit	64-bit
macOS Installer (.pkg)	64-bit / ARM64	
macOS Binary (.tar.gz)	64-bit	ARM64
Linux Binaries (x64)	64-bit	

2) Create a folder and open in vscode.

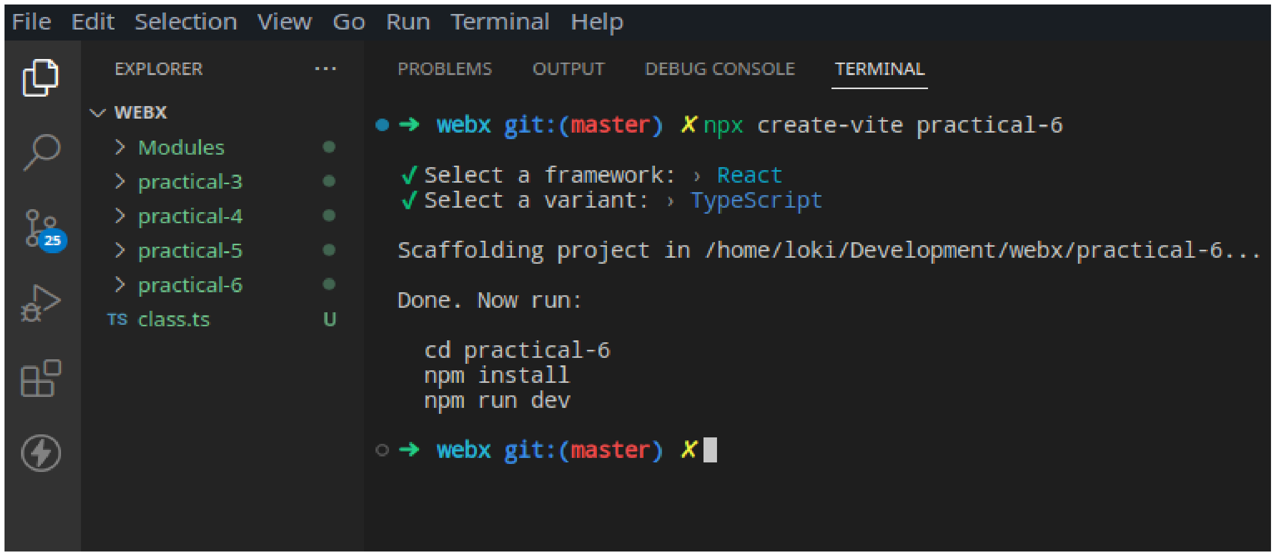
3) Run the below command:

- npx create-vite appName



```
○ ➜ webx git:(master) ✗ npx create-vite practical-6
```

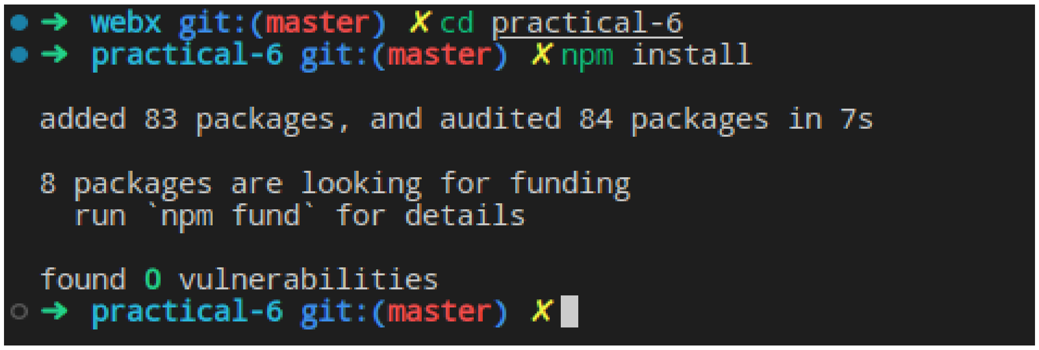
4) Select the framework as React and variant as Typescript



```
● ➜ webx git:(master) ✗ npx create-vite practical-6

  ✓ Select a framework: › React
  ✓ Select a variant: › TypeScript

 Scaffolding project in /home/loki/Development/webx/practical-6...

 Done. Now run:

   cd practical-6
   npm install
   npm run dev

○ ➜ webx git:(master) ✗
```

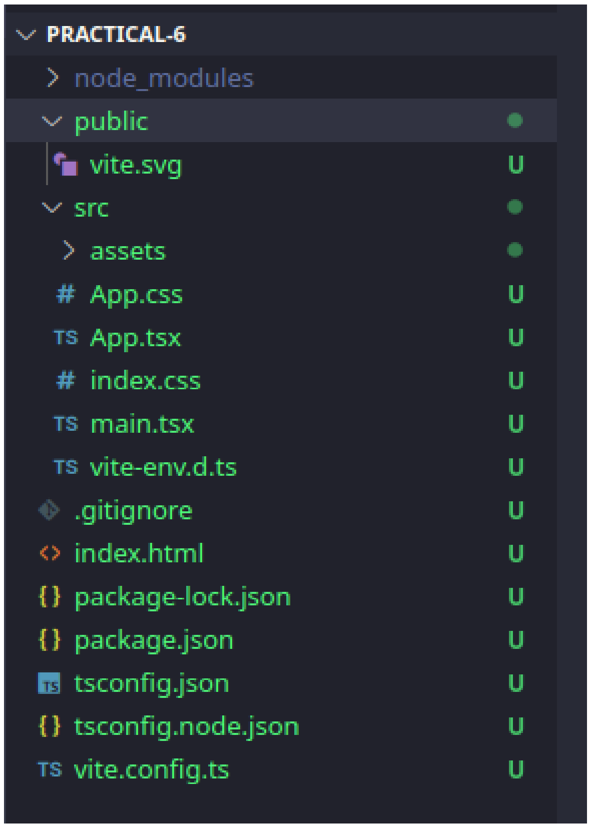
5) Run the command :

- npm install



```
● ➜ webx git:(master) ✗ cd practical-6
● ➜ practical-6 git:(master) ✗ npm install

  added 83 packages, and audited 84 packages in 7s

  8 packages are looking for funding
    run `npm fund` for details

  found 0 vulnerabilities
○ ➜ practical-6 git:(master) ✗
```

6) Directory Structure



```
PRACTICAL-6
  node_modules
  public                    ●
    vite.svg                U
  src                       ●
    assets                  ●
    App.css                 U
    App.tsx                 U
    index.css               U
    main.tsx                U
    vite-env.d.ts           U
  .gitignore                U
  index.html                U
  package-lock.json         U
  package.json              U
  tsconfig.json             U
  tsconfig.node.json        U
  vite.config.ts            U
```

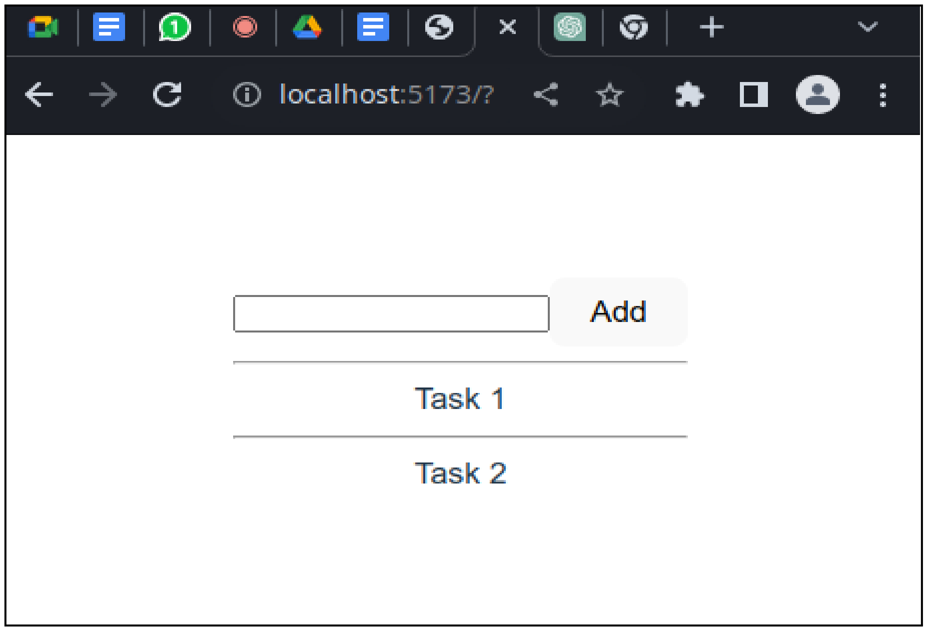
7) To run the app ,run the command:

- npm run dev

8) Code for App.tsx:

```
import { useState } from 'react'
import './App.css'

function App() {
 const [keyword, setKeyword] = useState<string>("");
 const [keywordList, setKeywordList] = useState<string[]>([]);  function
onChange(event: React.ChangeEvent<HTMLInputElement>){
 setKeyword(event.target.value + "");
 }
 function handleSubmit(event: React.FormEvent<HTMLFormElement>){
event.preventDefault();
 const list = keywordList;
 list.push(keyword);
 console.log(keywordList);
 setKeywordList(keywordList);
 }
 return (
 <div className="App">
 <form onSubmit={handleSubmit}>
 <input value={keyword} onChange={onChange} type={"text"}/>  <button
type={"submit"}>Add</button>
 </form>
 {
 keywordList.map((item,index) => <div key={index}>
 <hr></hr>
 <div>{item}</div>
 </div>)
 }
 </div>
 )
}

export default App
```

Output:



Conclusion: Developing a simple website using TypeScript enhances front-end development skills while leveraging the advantages of static typing. TypeScript improves code maintainability, reduces runtime errors, and integrates well with modern frameworks.