

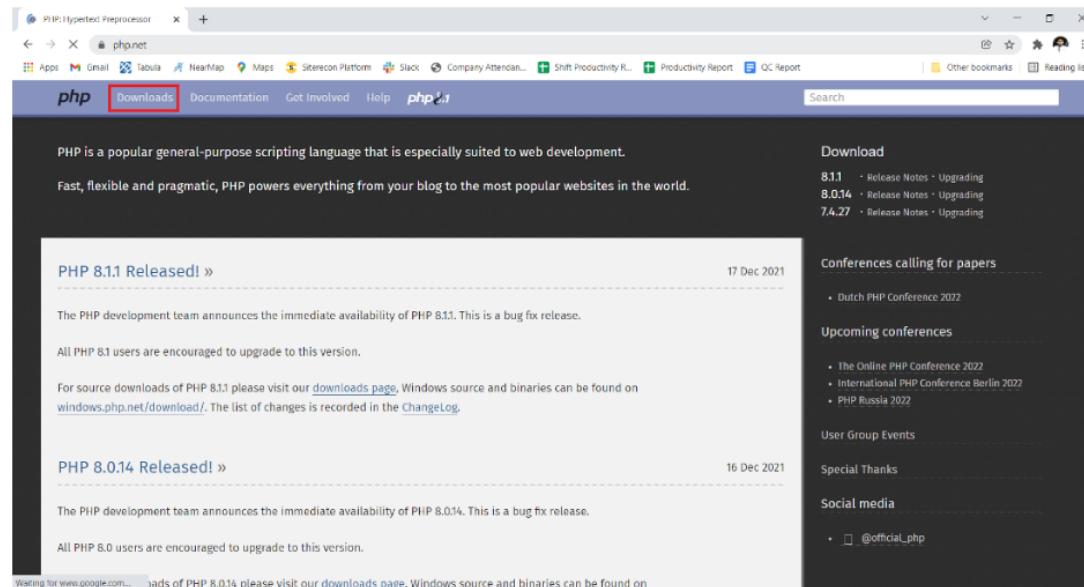
How to install PHP in windows 10 ?

PHP is a general-purpose scripting language geared towards web development. PHP is open-source software (OSS). PHP is free to use and download. PHP stands for PHP Hypertext Preprocessor. PHP files have an extension called .php. PHP supports many databases MySQL, Oracle, etc. PHP is free to download and use. It was created by a Danish Canadian and designed by Rasmus Lerdorf in 1994. The common websites that are used are eCommerce, Social Platforms, Email Hosting, etc.

Installing PHP on Windows

Follow the below steps to install PHP on Windows:

Step 1: Visit <https://www.php.net/downloads> website using any web browser and click on Downloads.



Step 2: Click on the Windows “Downloads” button.

Current Stable PHP 8.1.1 (Changelog)

- [php-8.1.1.tar.gz \(sig\)](#) [19165Kb] sha256: 4ee+c1f18438311f16c55cd21de8f26634ea3cd4a5e77c88357cbec4a2d671d 16 Dec 2021
- [php-8.1.1.tar.bz2 \(sig\)](#) [14,926Kb] sha256: 8f8bb9c9cadcc127edc111f7db8a189745e2f638770a161b3c22a2953f79b40e 16 Dec 2021
- [php-8.1.1.tar.xz \(sig\)](#) [11,454Kb] sha256: 33c69d76daabb5dd930d9dd32e6bf44e9efcf867563759eb5492c3aff8856 16 Dec 2021
- [Windows downloads](#)

[GPG Keys for PHP 8.1](#)

Old Stable PHP 8.0.14 (Changelog)

- [php-8.0.14.tar.gz \(sig\)](#) [17,088Kb] sha256: e67ebd8c4c77247ad1a88829e5b95d51a19edf3d8781443de261e20a63ea20 16 Dec 2021
- [php-8.0.14.tar.bz2 \(sig\)](#) [13,187Kb] sha256: bb381fdff817ad7c2a232ea7f77cad68dceb86eb3ac1a37acedf8ad0a0cd4b 16 Dec 2021
- [php-8.0.14.tar.xz \(sig\)](#) [10,606Kb] sha256: fbd682a7ac70e4de73449d9fefc8b495d373b5be9c18cdb645fb431c91156e3 16 Dec 2021
- [Windows downloads](#)

[GPG Keys for PHP 8.0](#)

Supported Versions

Check the supported versions page for more information on the support lifetime of each version of PHP.

[Documentation download](#)

[PHP logos](#)

[Development sources \(git\)](#)

[Old archives](#)

Step 3: The new webpage has different options, choose the Thread safe version, and click on the zip button and Download it.

PHP For Windows

This site is dedicated to supporting PHP on Microsoft Windows. It also supports ports of PHP extensions or features as well as providing special builds for the various Windows architectures.

If you like to build your own PHP binaries, instructions can be found on the [Wiki](#).

PECL For Windows

PECL extensions for Windows is being worked on. Windows DLL can be downloaded right from the [PECL website](#).

The PECL extension [release](#) and [snapshot](#) build directories are browsable directly.

Which version do I choose?

IIS

If you are using PHP as FastCGI with IIS you should use the Non-Thread

PHP 8.2 (8.2.6)

- [Download source code](#) [26.01MB]
- [Download tests package \(.phar\)](#) [15.52MB]

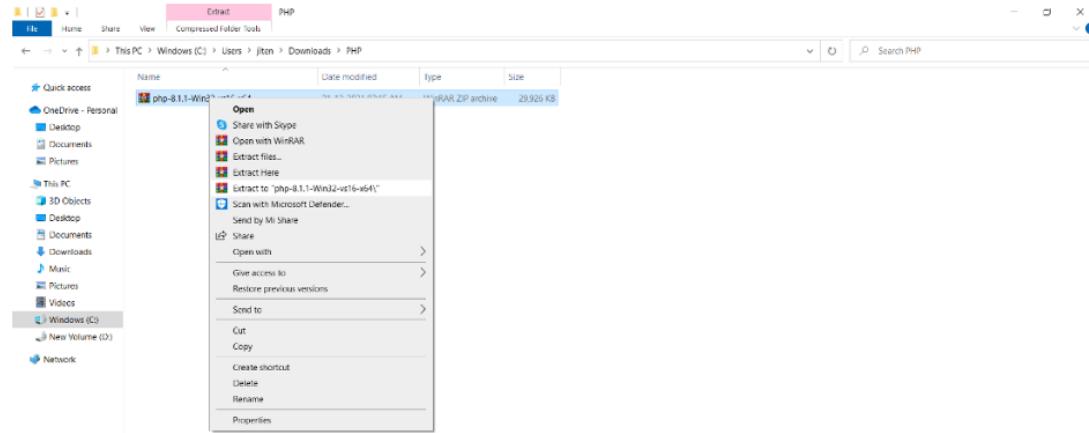
VS16 x64 Non Thread Safe (2023-May-09 16:47:36)

- [Zip](#) [30.23MB] sha256: 22068ea2bld83398e794652018445c37852bac60ef2ac4715854d59130912516
- [Debug Pack](#) [24.5MB] sha256: 2a66faa629d19bfb25da947fb3bc65463cec8ca3ebab54bfae9651d2980ddc
- [Development package \(SDK to develop PHP extensions\)](#) [1.23MB] sha256: e14024f2d6f50a296df48db1f38040bf9032c9ca7134c1a10aa0c5fd6723570

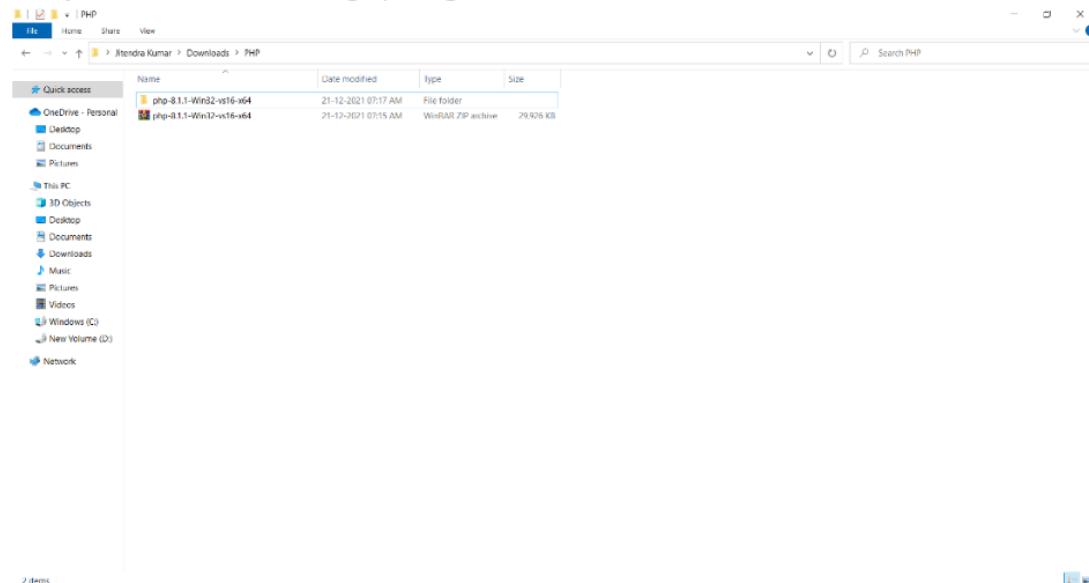
VS16 x64 Thread Safe (2023-May-09 16:37:30)

- [Zip](#) [30.33MB] sha256: e6c795c8a7fb9642923919c54192ae7bc26353ff6b69e95caf7cf3a545914b9d
- [Debug Pack](#) [24.52MB] sha256: 1175dac1cdca5529bf82ed7b749a44309330ccc591ba780e7fa58a586220f4
- [Development package \(SDK to develop PHP extensions\)](#) [1.24MB] sha256: 1d14dc15c6a97ac838b950936c21b5255c5d2eb19baae9c99b7ab8204c77f9d

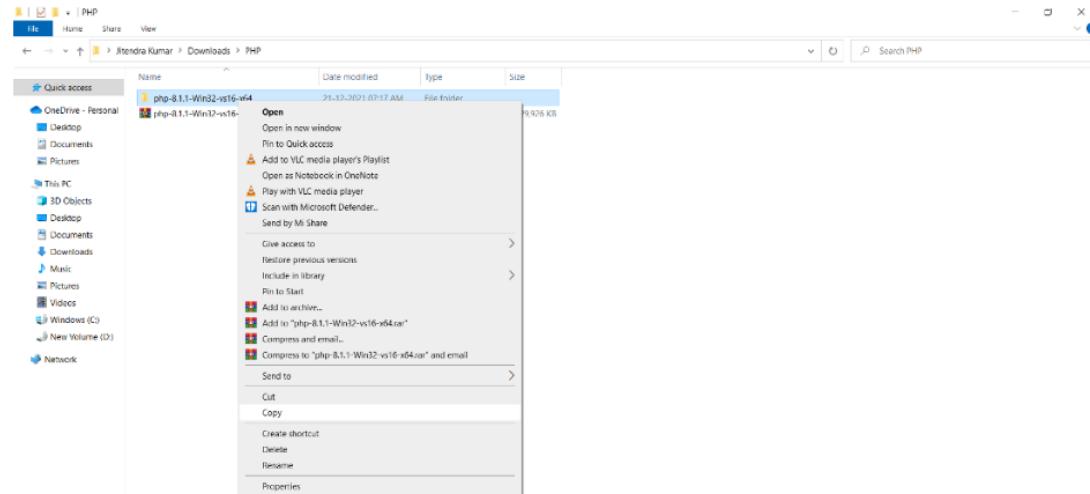
Step 4: Now check for the zip file in downloads in your system and extract it.



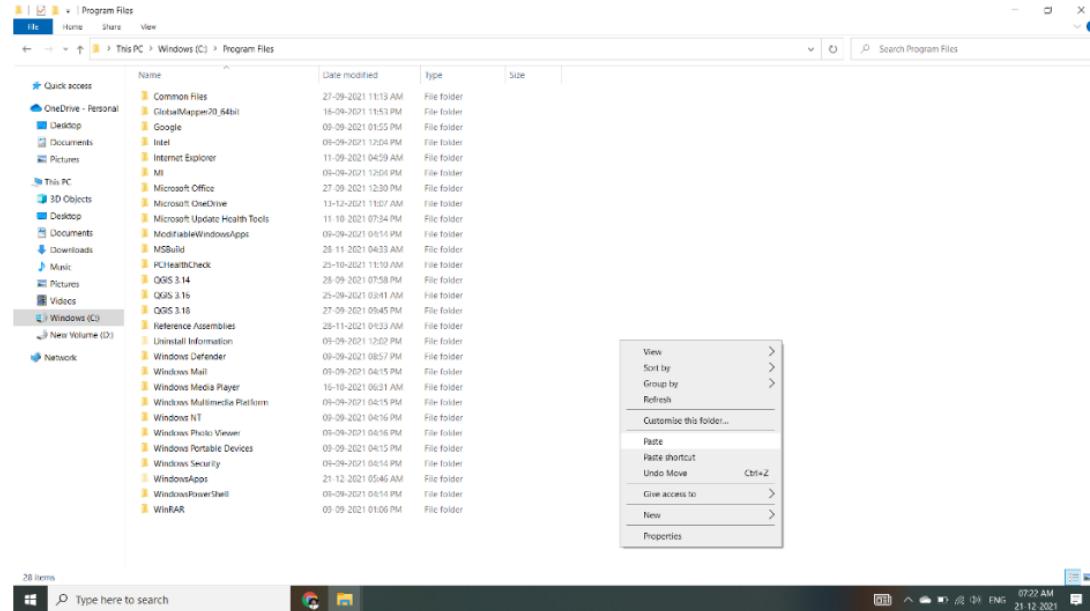
Step 5: After extracting, you get the extracted folder.



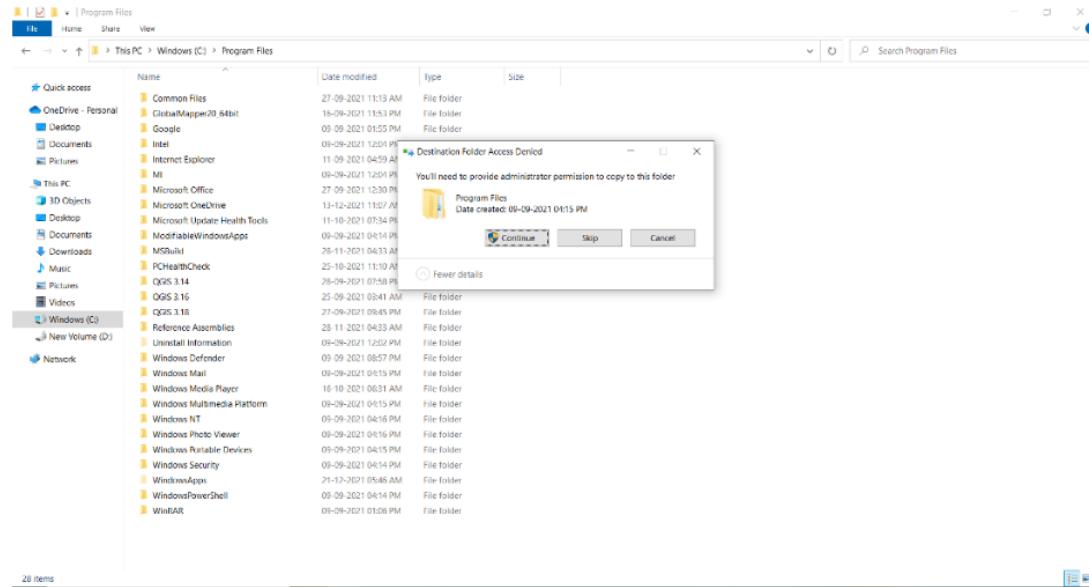
Step 6: Now copy the extracted folder.



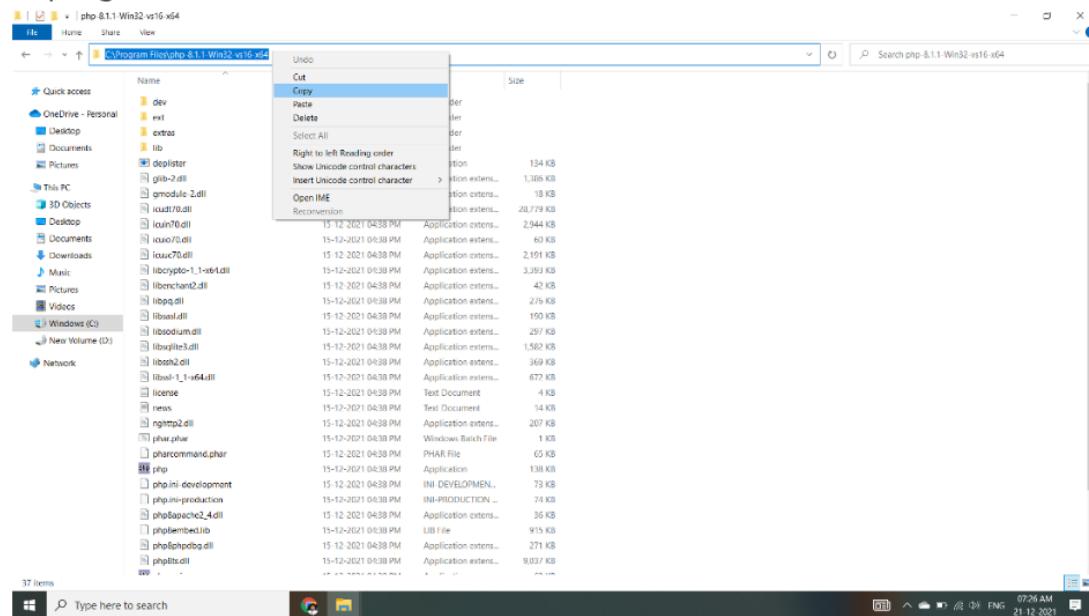
Step 7: Now paste the copy folder in your windows drive in the Program files folder.



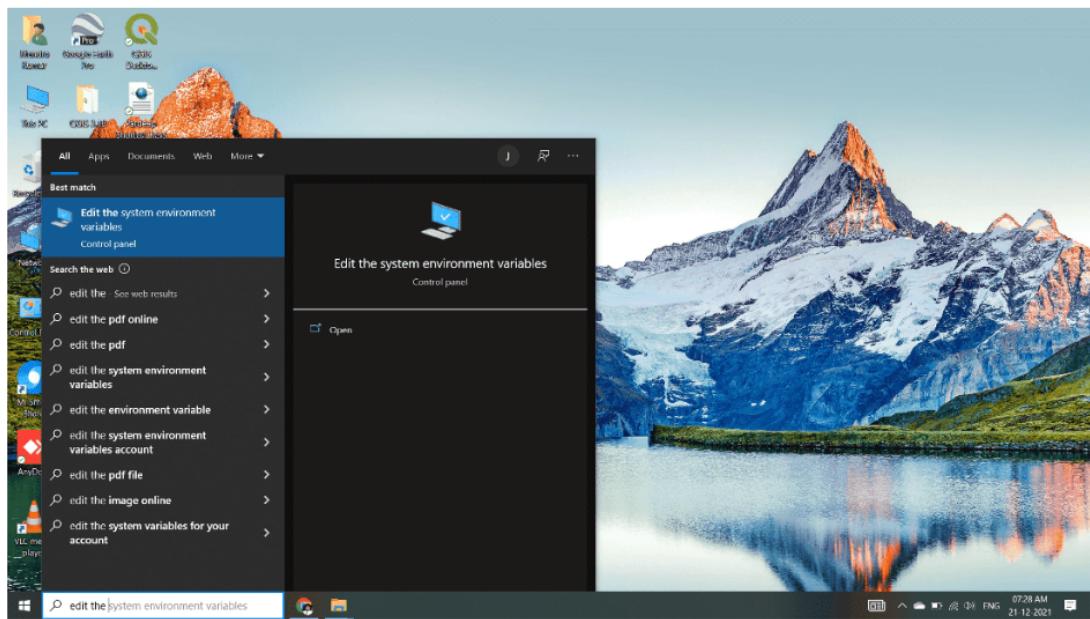
Step 8: Now the Permission Windows appears to paste the folder in program files then click on “Continue”.



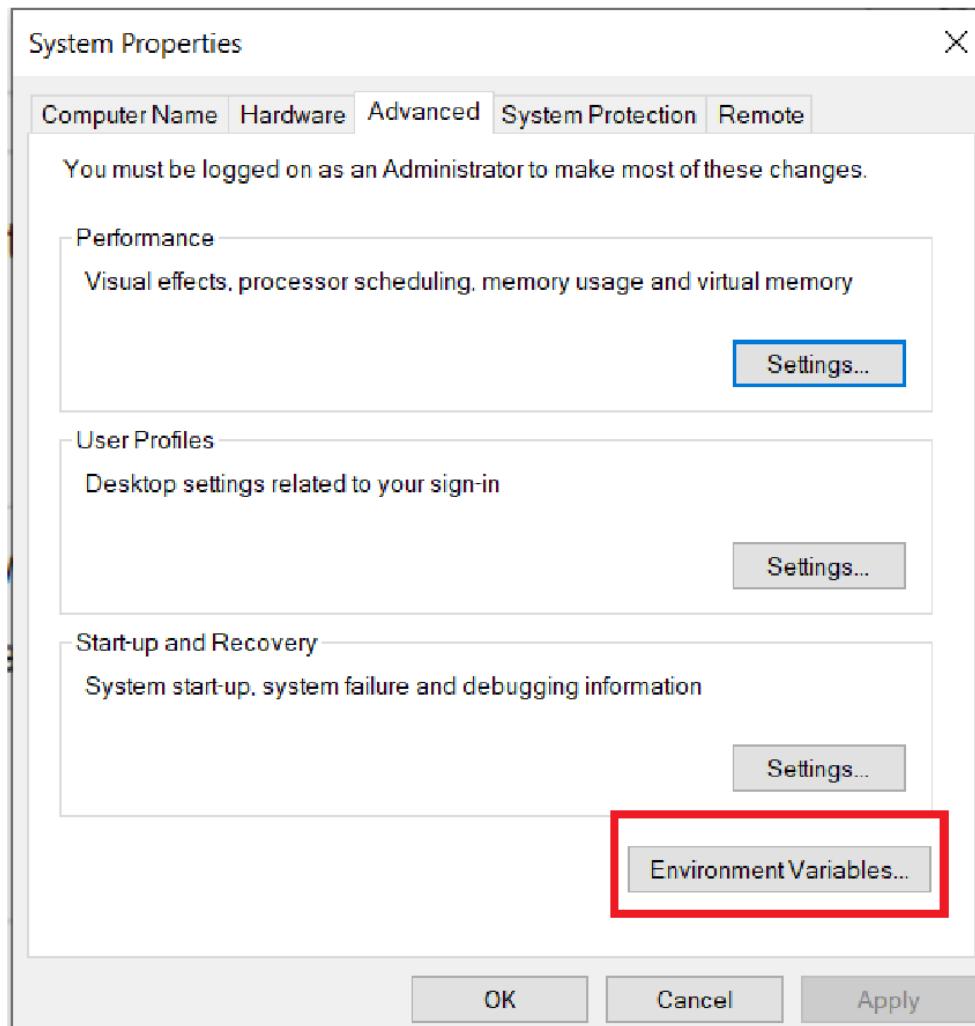
Step 9: Now after pasting the folder then copy the address of the folder in program files.



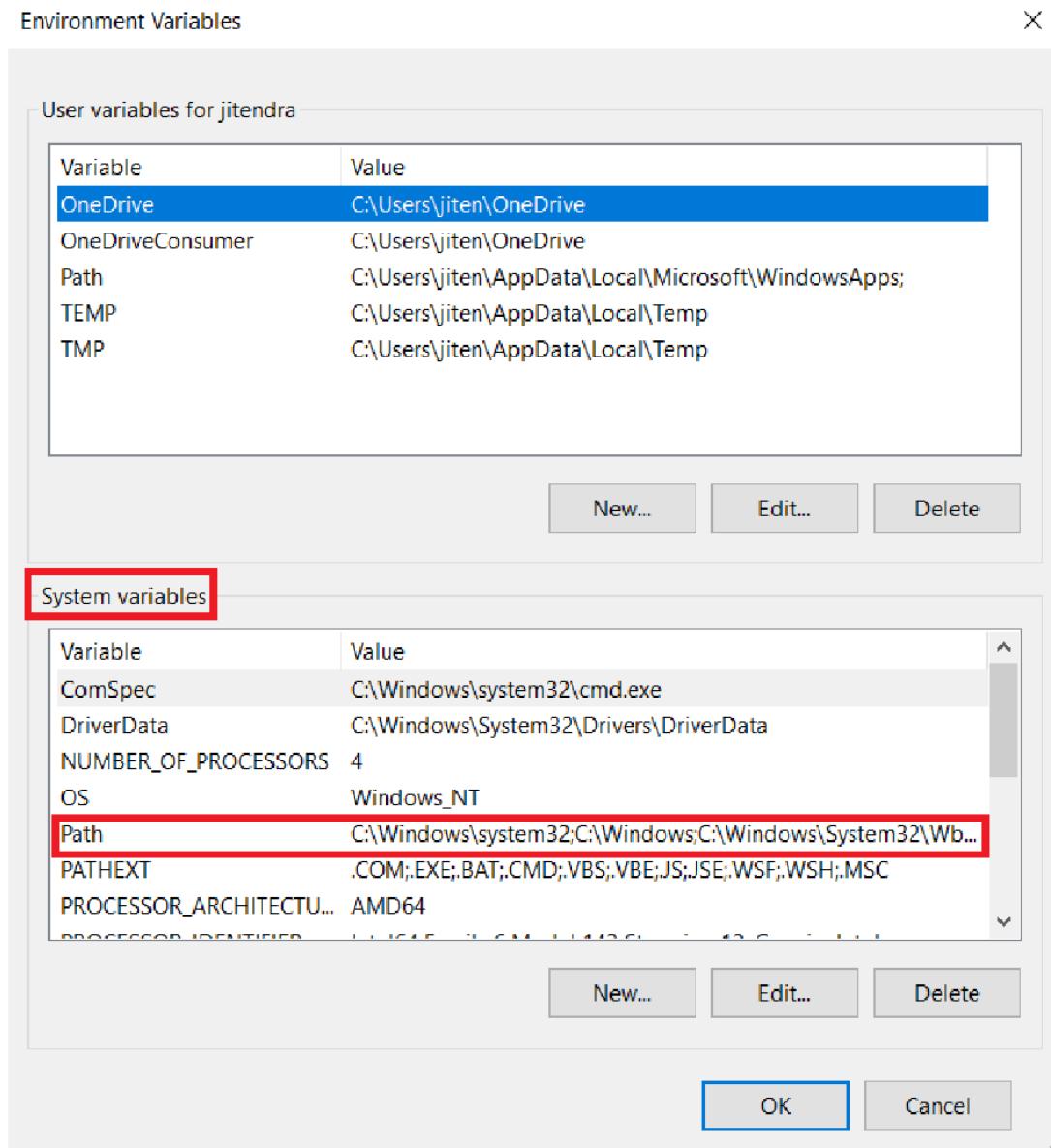
Step 10: Now click on Start Menu and search “Edit the system environment variables” and open it.



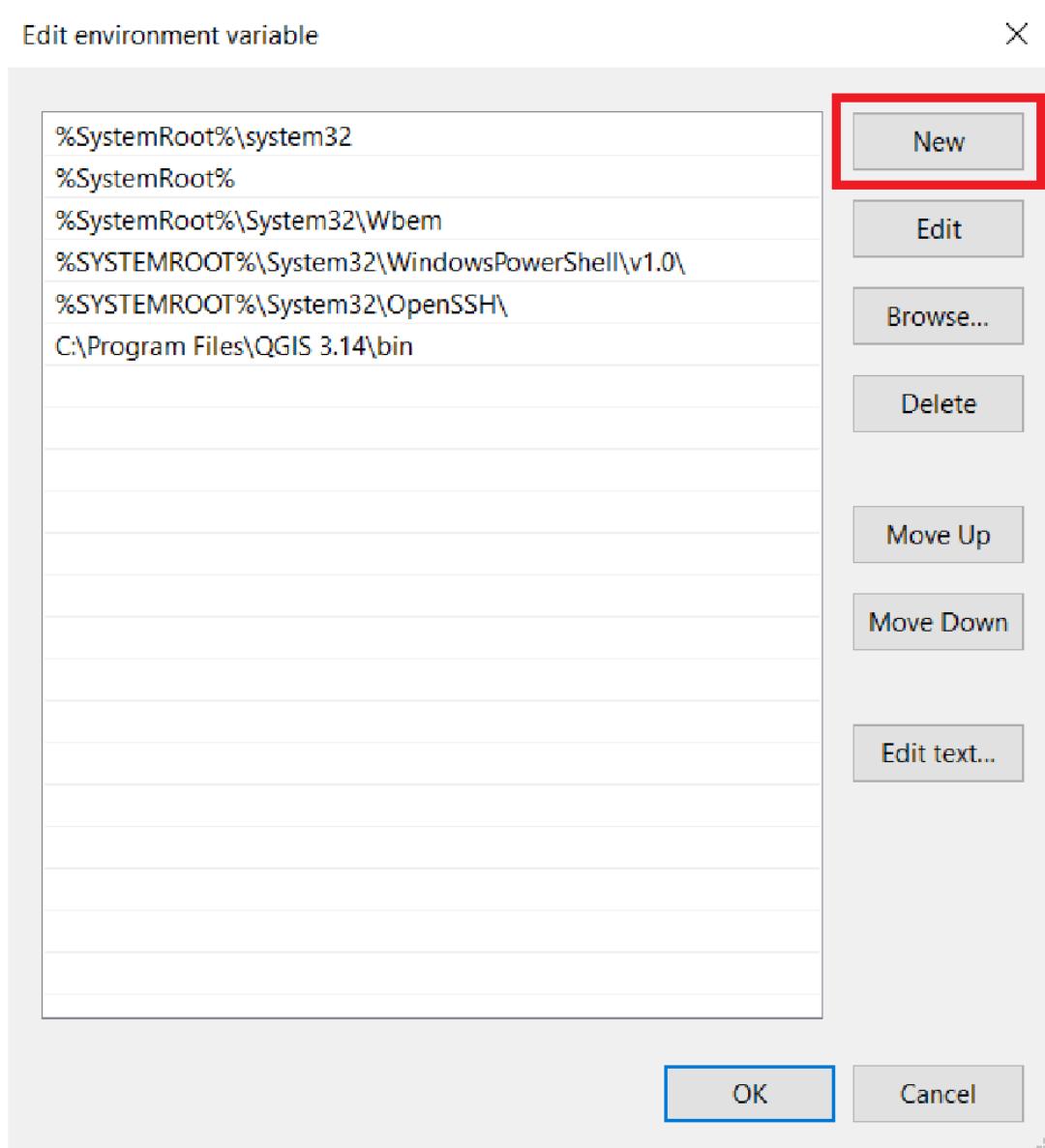
Step 11: After opening System, Variable New window appears, and click on “Environment Variables...”



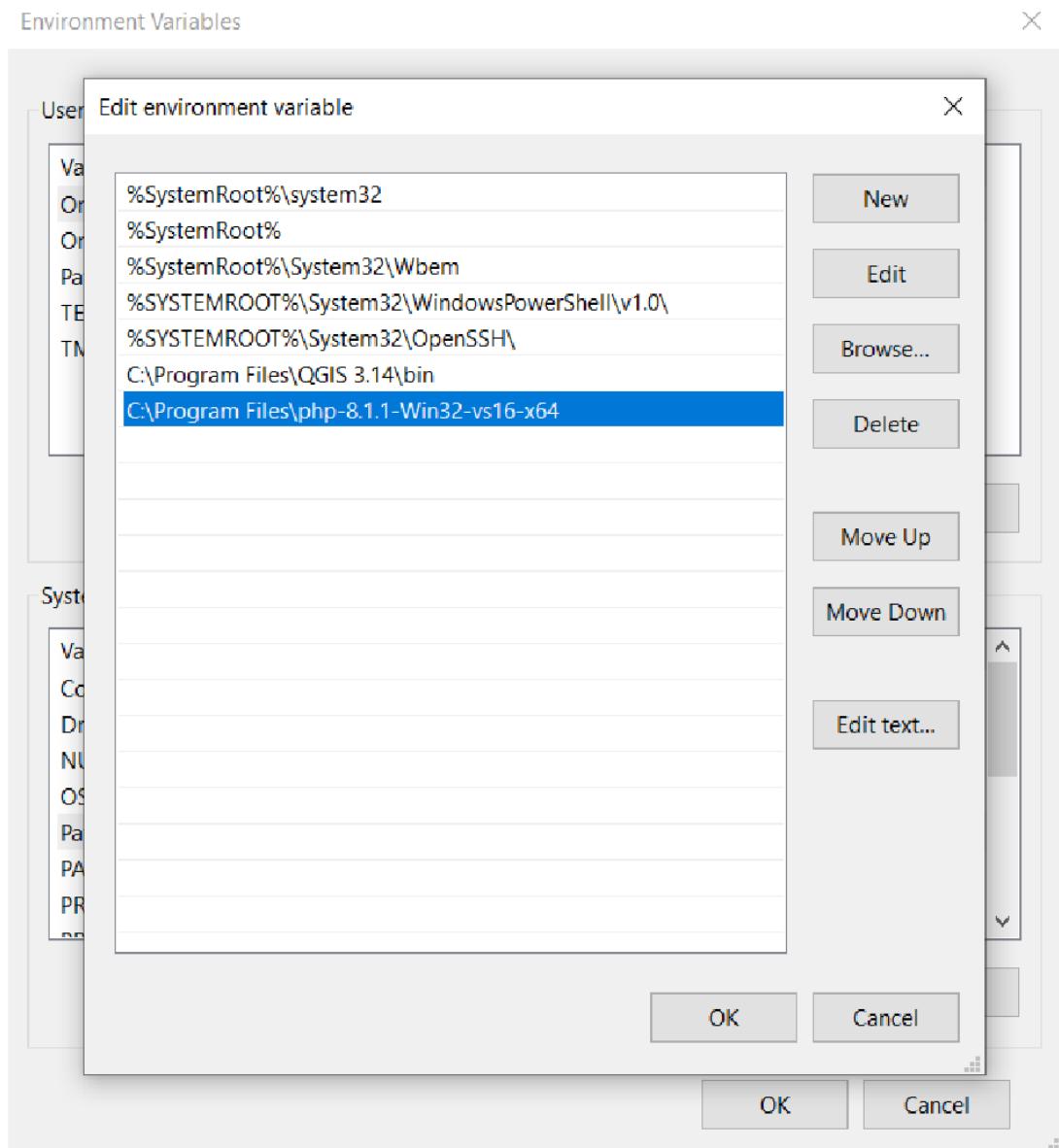
Step 12: Now go to the “System variables” Path option and double click on Path.



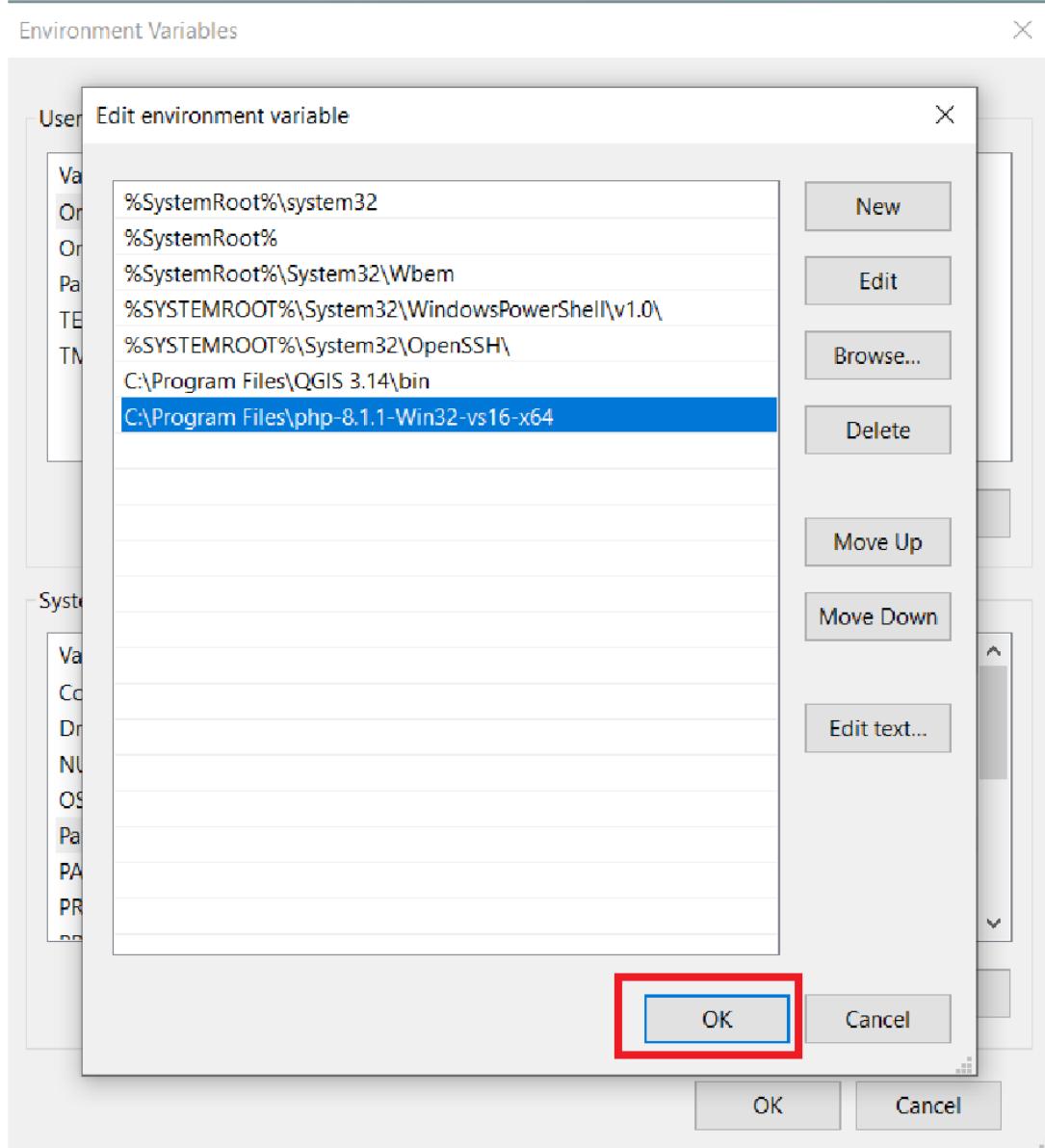
Step 13: Next screen will open and click on the “New” button.



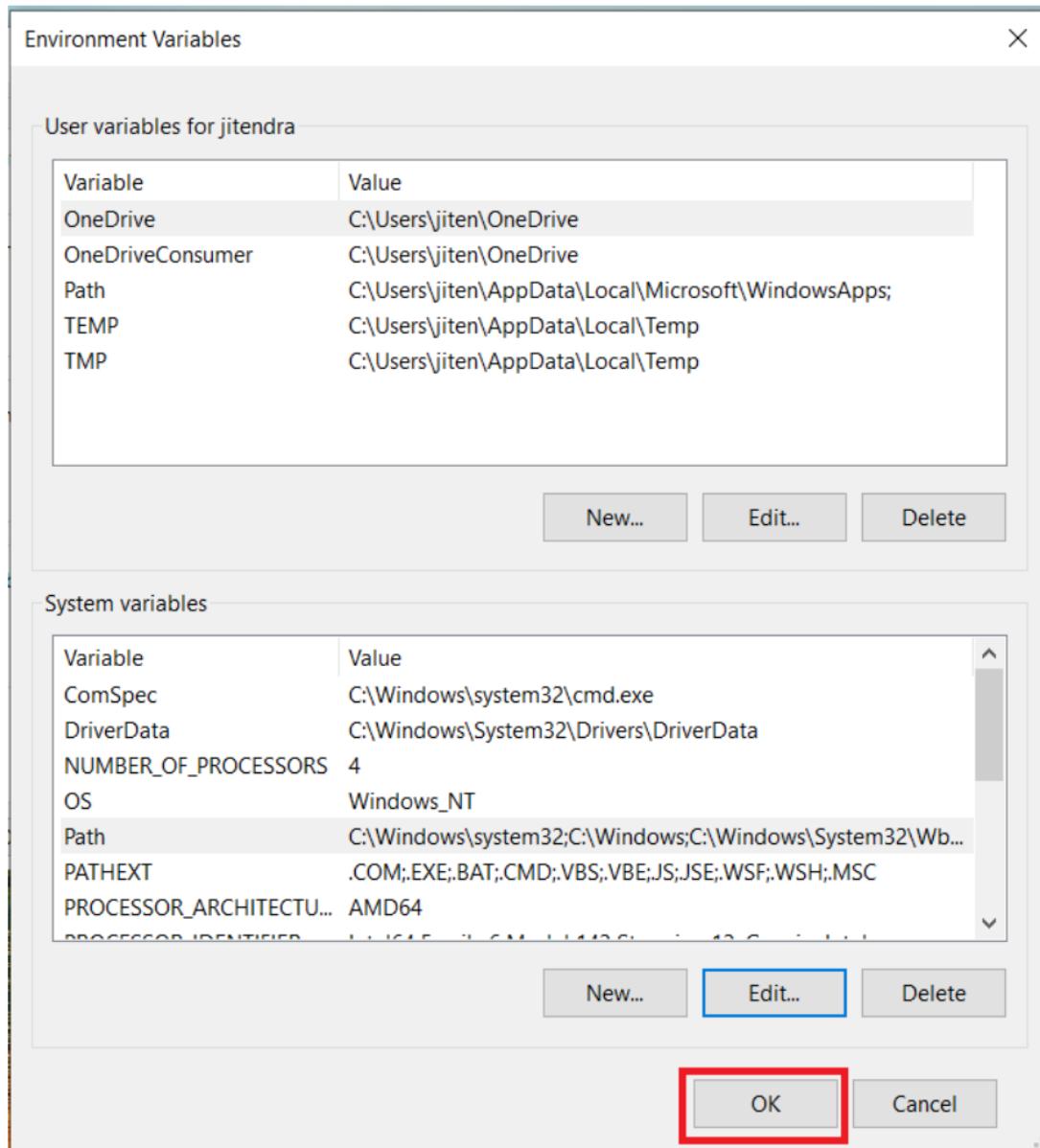
Step 14: After New Paste the address we copy from program files to new and click on *Enter* button.



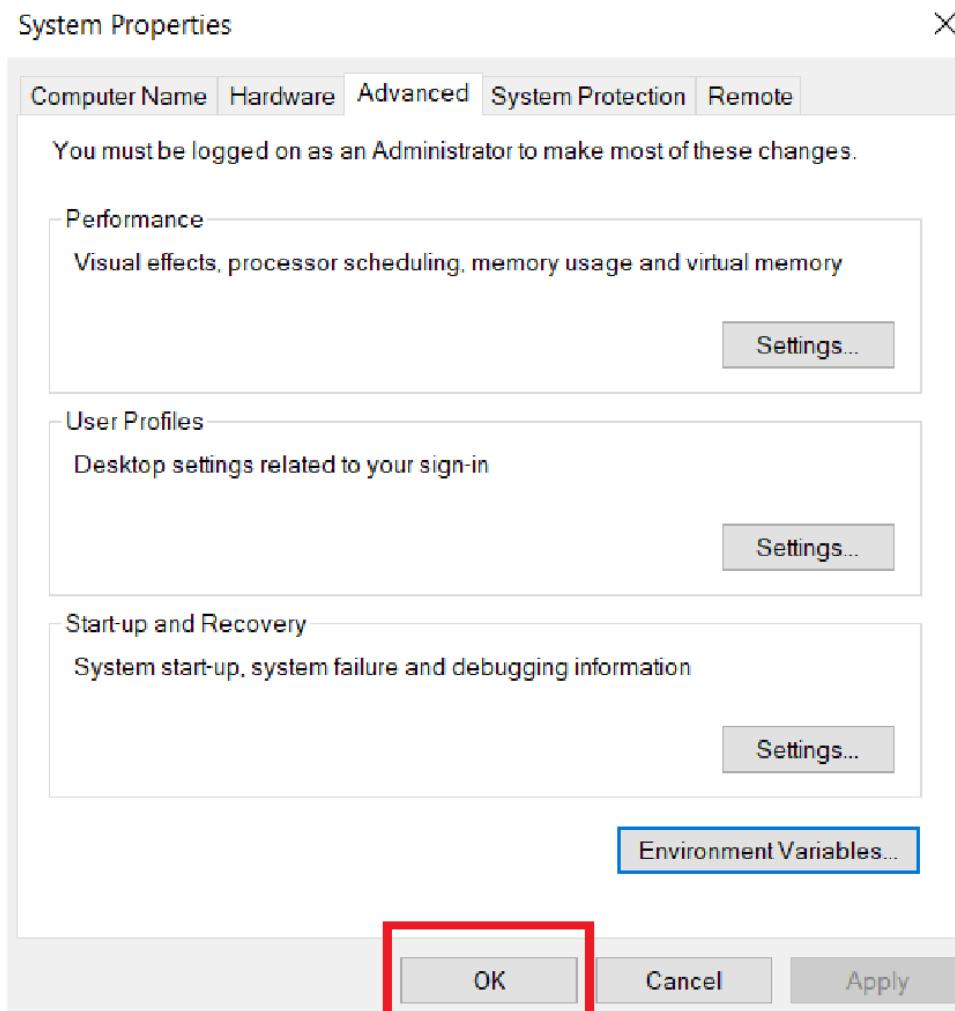
Step 15: Now Click on the **OK** button.



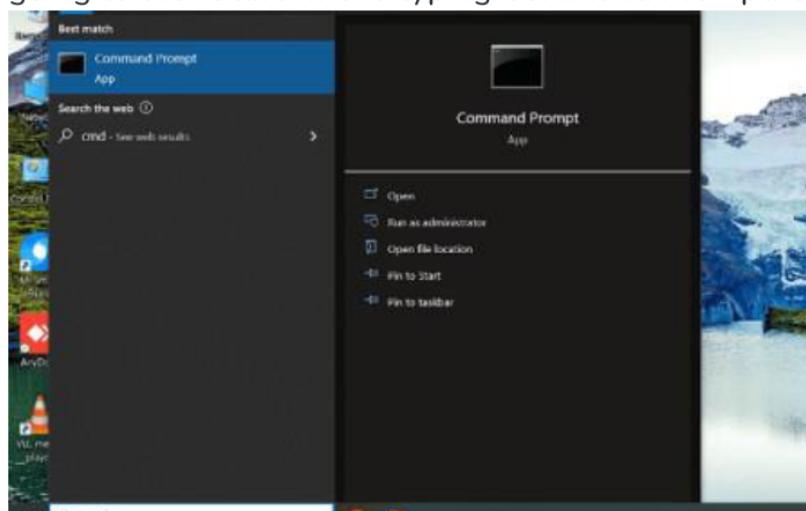
Step 16: Click on the **OK** button.

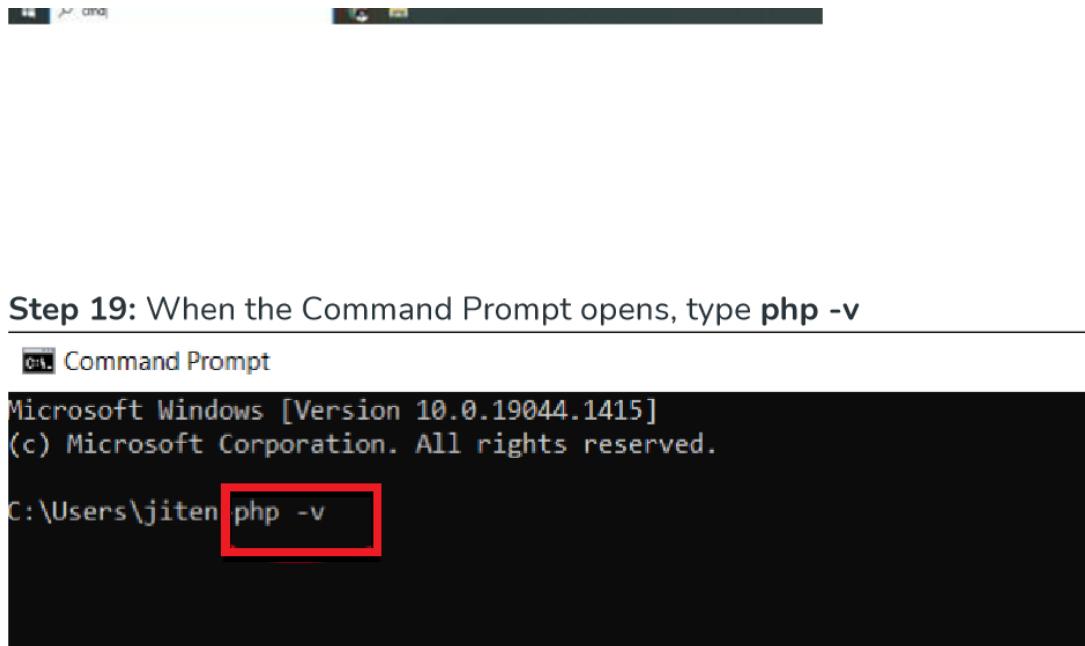


Step 17: Click on **OK** for saving changes.



Step 18: Now your PHP is installed on your computer. You may check by going to the “Start” menu typing Command Prompt. Open it.



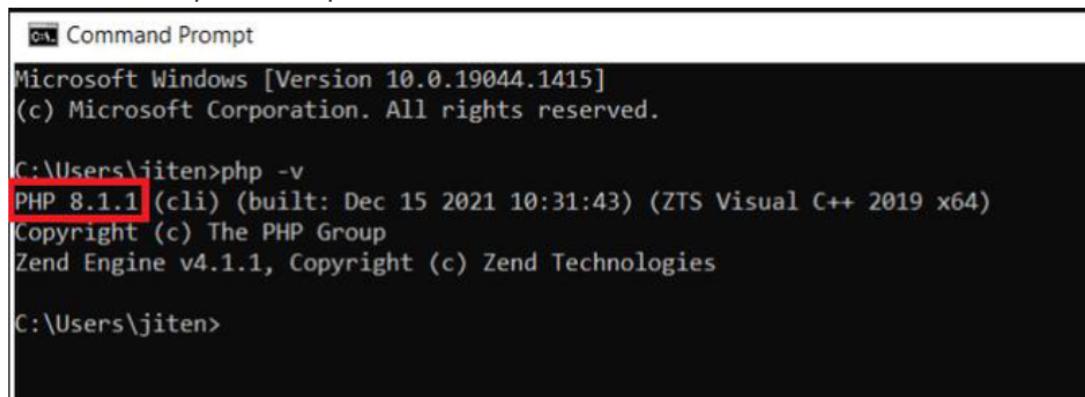


Step 19: When the Command Prompt opens, type **php -v**

```
cmd Command Prompt
Microsoft Windows [Version 10.0.19044.1415]
(c) Microsoft Corporation. All rights reserved.

C:\Users\jiten php -v
```

Step 20: Now enter the command prompt to show the version of PHP installed on your computer.



```
cmd Command Prompt
Microsoft Windows [Version 10.0.19044.1415]
(c) Microsoft Corporation. All rights reserved.

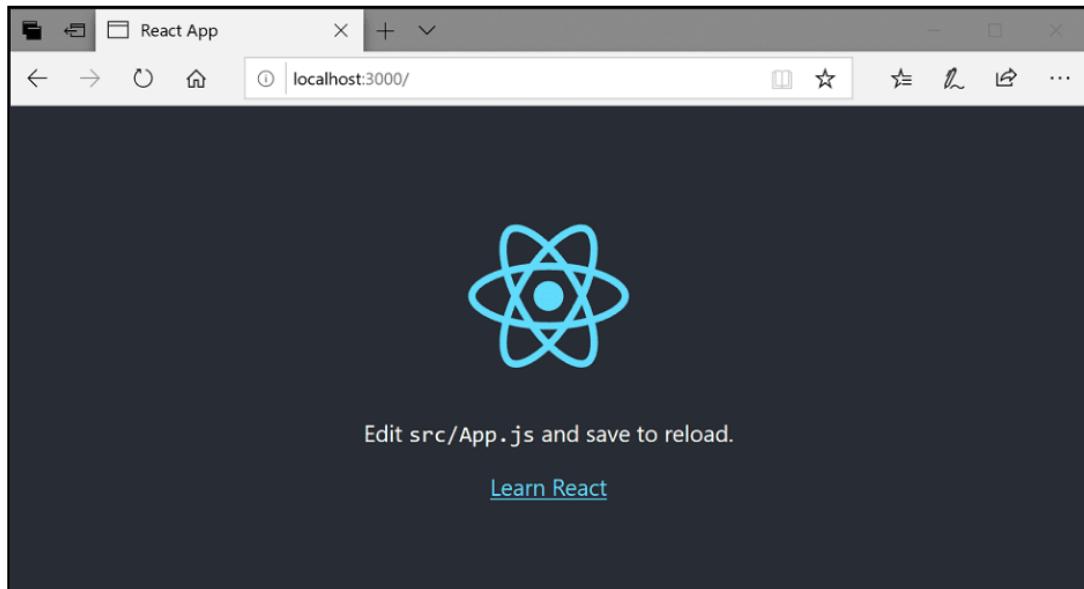
C:\Users\jiten>php -v
PHP 8.1.1 (cli) (built: Dec 15 2021 10:31:43) (ZTS Visual C++ 2019 x64)
Copyright (c) The PHP Group
Zend Engine v4.1.1, Copyright (c) Zend Technologies

C:\Users\jiten>
```

You have successfully installed PHP on your Windows 10 system.

2. Using React in Visual Studio Code

[React](#) is a popular JavaScript library developed by Facebook for building user interfaces. The Visual Studio Code editor supports React.js IntelliSense and code navigation out of the box.



Welcome to React

We'll be using the [create-react-app generator](#) for this tutorial. To use the generator as well as run the React application server, you'll need [Node.js](#) JavaScript runtime and [npm](#) (Node.js package manager) installed. npm is included with Node.js which you can download and install from [Node.js downloads](#).

Tip: To test that you have Node.js and npm correctly installed on your machine, you can type `node --version` and `npm --version` in a terminal or command prompt. You can now create a new React application by typing:

```
npx create-react-app my-app
```

where `my-app` is the name of the folder for your application. This may take a few minutes to create the React application and install its dependencies.

Note: If you've previously installed `create-react-app` globally via `npm install -g create-react-app`, we recommend you uninstall the package using `npm uninstall -g create-react-app` to ensure that `npx` always uses the latest version.

Let's quickly run our React application by navigating to the new folder and typing `npm start` to start the web server and open the application in a browser:

```
cd my-app  
npm start
```

You should see the React logo and a link to "Learn React" on <http://localhost:3000> in your browser. We'll leave the web server running while we look at the application

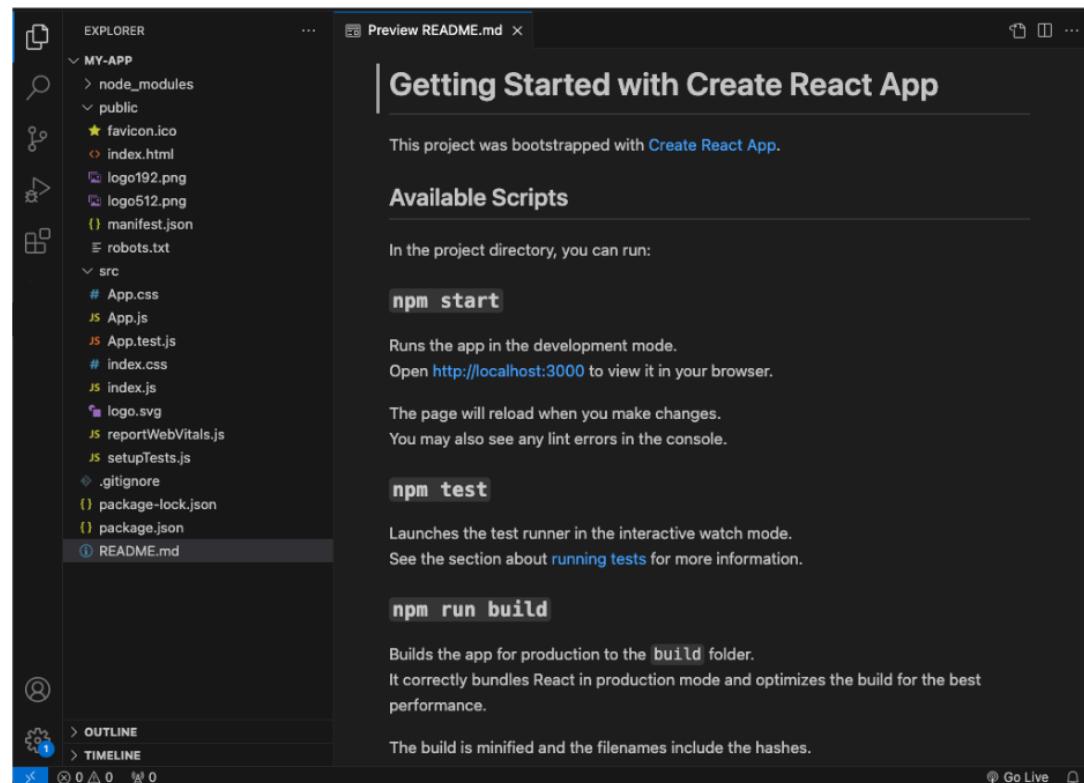
with VS Code.

To open your React application in VS Code, open another terminal or command prompt window, navigate to the `my-app` folder and type code `..`:

```
cd my-app
code .
```

[Markdown preview](#)

In the File Explorer, one file you'll see is the application `README.md` Markdown file. This has lots of great information about the application and React in general. A nice way to review the `README` is by using the VS Code [Markdown Preview](#). You can open the preview in either the current editor group (**Markdown: Open Preview** `Ctrl+Shift+V`) or in a new editor group to the side (**Markdown: Open Preview to the Side** `Ctrl+K V`). You'll get nice formatting, hyperlink navigation to headers, and syntax highlighting in code blocks.



[Syntax highlighting and bracket matching](#)

Now expand the `src` folder and select the `index.js` file. You'll notice that VS Code has syntax highlighting for the various source code elements and, if you put the cursor on a parenthesis, the matching bracket is also selected.

```
1 import React from 'react';
2 import ReactDOM from 'react-dom/client';
3 import './index.css';
4 import App from './App';
5 import reportWebVitals from './reportWebVitals';
6
7 const root = ReactDOM.createRoot(document.getElementById('root'));
8 root.render(
9   <React.StrictMode>
10   |   <App />
11   </React.StrictMode>
12 );
```

IntelliSense

As you start typing in `index.js`, you'll see smart suggestions or completions.

```
1 import React from 'react';
2 import ReactDOM from 'react-dom/client';
3 import './index.css';
4 import App from './App';
5 import reportWebVitals from './reportWebVitals';
6
7 read
8 const [e] React      (alias) namespace Reactimport Re... entById('root'));
9 root[e] ReactDOM
10  |<R[e] ReadableByteStreamController
11  |  | requestIdleCallback
12  |</[e] ReadableStreamDefaultController
13  );  |[e] RemotePlayback
14  |  | RTCIceCandidate
15 // I[e] RTCEncodedAudioFrame
16 // t[e] SpeechRecognitionAlternative
17 // o[e] RTCDTMFToneChangeEvent
18 repo[e] RTCCertificate
19     |[e] RTCEncodedVideoFrame
```

your app, pass a function
onsole.log))
<https://bit.ly/CRA-vitals>

After you select a suggestion and type `..`, you see the types and methods on the object through IntelliSense.

A screenshot of the VS Code editor showing a file named App.js. The code imports React, ReactDOM, and App from their respective files. It then uses the React.createRoot method to render an App component into a root element. A tooltip is displayed over the 'createRoot' call, providing documentation: 'Creates a new root instance. If you pass a function to the sole argument, it will be called with the current render function. This is useful for testing or for creating multiple roots in your app, pass a function to the sole argument (e.g. jest.fn())'. Below the tooltip, there is a link: 'https://bit.ly/CRA-vitals'.

```
1 import React from 'react';
2 import ReactDOM from 'react-dom/client';
3 import './index.css';
4 import App from './App';
5 import reportWebVitals from './reportWebVitals';
6
7 React.
8 const root = React.createRoot(container: Element | DocumentFragment, options?: ReactDOM.RootOptions | undefined): ReactDOM.Root
9 root.render();
10
11 // If you pass a function to the sole argument, it will be called with the current render function. This is useful for testing or for creating multiple roots in your app, pass a function to the sole argument (e.g. jest.fn())
12 // https://bit.ly/CRA-vitals
```

VS Code uses the TypeScript language service for its JavaScript code intelligence and it has a feature called [Automatic Type Acquisition](#) (ATA). ATA pulls down the npm Type Declaration files (*.d.ts) for the npm modules referenced in the package.json.

If you select a method, you'll also get parameter help:

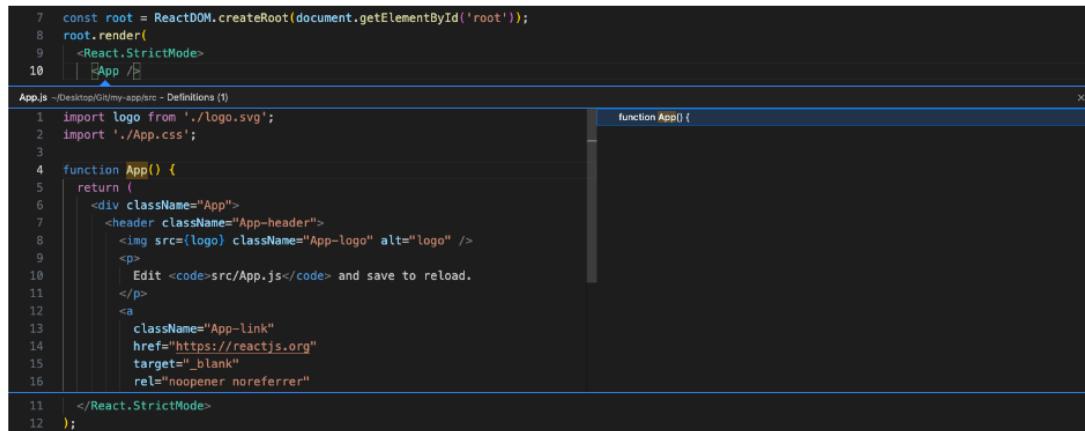
A screenshot of the VS Code editor showing the same App.js file. The cursor is over the 'root.render' call. A tooltip provides detailed information about the parameters: 'Creates a new root instance. If you pass a function to the sole argument, it will be called with the current render function. This is useful for testing or for creating multiple roots in your app, pass a function to the sole argument (e.g. jest.fn())'. Below the tooltip, there is a link: 'https://bit.ly/CRA-vitals'.

```
1 import React from 'react';
2 import ReactDOM from 'react-dom/client';
3 import './index.css'; function createRoot(container: Element | DocumentFragment, options?: ReactDOM.RootOptions | undefined): ReactDOM.Root
4 import App from './App' Replaces ReactDOM.render when the .render method is called and enables Concurrent Mode.
5 import reportWebVitals
6
7 const root = React.createRoot([document.getElementById('root')]);
8 root.render(
9   <React.StrictMode>
10     <App />
11   </React.StrictMode>
12 );
```

[Go to Definition, Peek definition](#)

Through the TypeScript language service, VS Code can also provide type definition information in the editor through **Go to Definition** (F12) or **Peek Definition**.

Put the cursor over the App, right click and select **Peek Definition**. A [Peek window](#) will open showing the App definition from App.js.



The screenshot shows the Firefox Peek window displaying the source code of `App.js`. The code defines a `function App()` that creates a React component with a logo and a link to the React website.

```
7 const root = ReactDOM.createRoot(document.getElementById('root'));
8 root.render(
9   <React.StrictMode>
10   |   <App />
11 );
12 
```

```
App.js -/Desktop/Git/my-app/src - Definitions (1)
1 import logo from './logo.svg';
2 import './App.css';
3
4 function App() {
5   return (
6     <div className="App">
7       <header className="App-header">
8         <img src={logo} className="App-logo" alt="logo" />
9         <p>
10           Edit <code>src/App.js</code> and save to reload.
11         </p>
12         <a
13           className="App-link"
14           href="https://reactjs.org"
15           target="_blank"
16           rel="noopener noreferrer"
17         >Get started </a>
18       </header>
19       <main className="App-main">
20         <h1>Hello World!</h1>
21       </main>
22     </div>
23   );
24 }
```

Press Escape to close the Peek window.

Hello World

Let's update the sample application to "Hello World!". Create a component inside `index.js` called `HelloWorld` that contains a H1 header with "Hello, world!" and replace the `<App />` tag in `root.render` with `<HelloWorld />`.

```
import React from 'react';

import ReactDOM from 'react-dom/client';

import './index.css';

import App from './App';

import reportWebVitals from './reportWebVitals';

function HelloWorld() {

  return <h1 className="greeting">Hello, world!</h1>;
}

const root = ReactDOM.createRoot(document.getElementById('root'));

root.render(

  <React.StrictMode>

    <HelloWorld />

  </React.StrictMode>
)
```

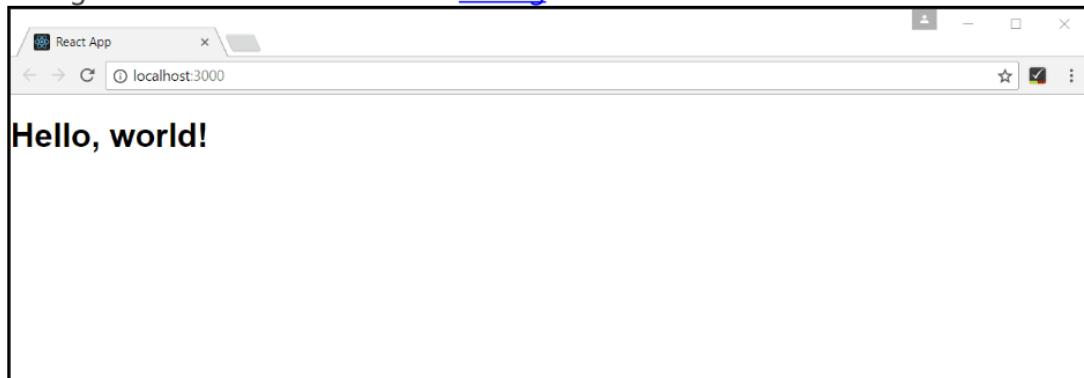
```
);
```

```
// If you want to start measuring performance in your app, pass a function
// to log results (for example: reportWebVitals(console.log))
// or send to an analytics endpoint. Learn more: https://bit.ly/CRA-vitals
reportWebVitals();
```

Once you save the `index.js` file, the running instance of the server will update the web page and you'll see "Hello World!" when you refresh your browser.

Tip: VS Code supports Auto Save, which by default saves your files after a delay.

Check the **Auto Save** option in the **File** menu to turn on Auto Save or directly configure the `files.autoSave` user [setting](#).



[Debugging React](#)

To debug the client side React code, we'll use the built-in JavaScript debugger.

Note: This tutorial assumes you have the Edge browser installed. If you want to debug using Chrome, replace the launch type with `chrome`. There is also a debugger for the [Firefox](#) browser.

[Set a breakpoint](#)

To set a breakpoint in `index.js`, click on the gutter to the left of the line numbers. This will set a breakpoint which will be visible as a red circle.

```
1 import React from 'react';
2 import ReactDOM from 'react-dom/client';
3 import './index.css';
4 import App from './App';
5 import reportWebVitals from './reportWebVitals';
6
● 7 function HelloWorld() {
8   return <h1 className="greeting">Hello, world!</h1>
9 }
10
11 const root = ReactDOM.createRoot(document.getElementById('root'));
12 root.render(
13   <React.StrictMode>
14     <HelloWorld />
15   </React.StrictMode>
16 );
17
18 // If you want to start measuring performance in your app, pass a function
19 // to log results (for example: reportWebVitals(console.log))
20 // or send to an analytics endpoint. Learn more: https://bit.ly/CRA-vitals
21 reportWebVitals();
```

[Configure the debugger](#)

We need to initially configure the [debugger](#). To do so, go to the **Run and Debug** view (Ctrl+Shift+D) and select the [create a launch.json file](#) link to create a `launch.json` debugger configuration file. Choose **Web App (Edge)** from the **Select debugger** dropdown list. This will create a `launch.json` file in a new `.vscode` folder in your project which includes a configuration to launch the website.

We need to make one change for our example: change the port of the url from 8080 to 3000. Your `launch.json` should look like this:

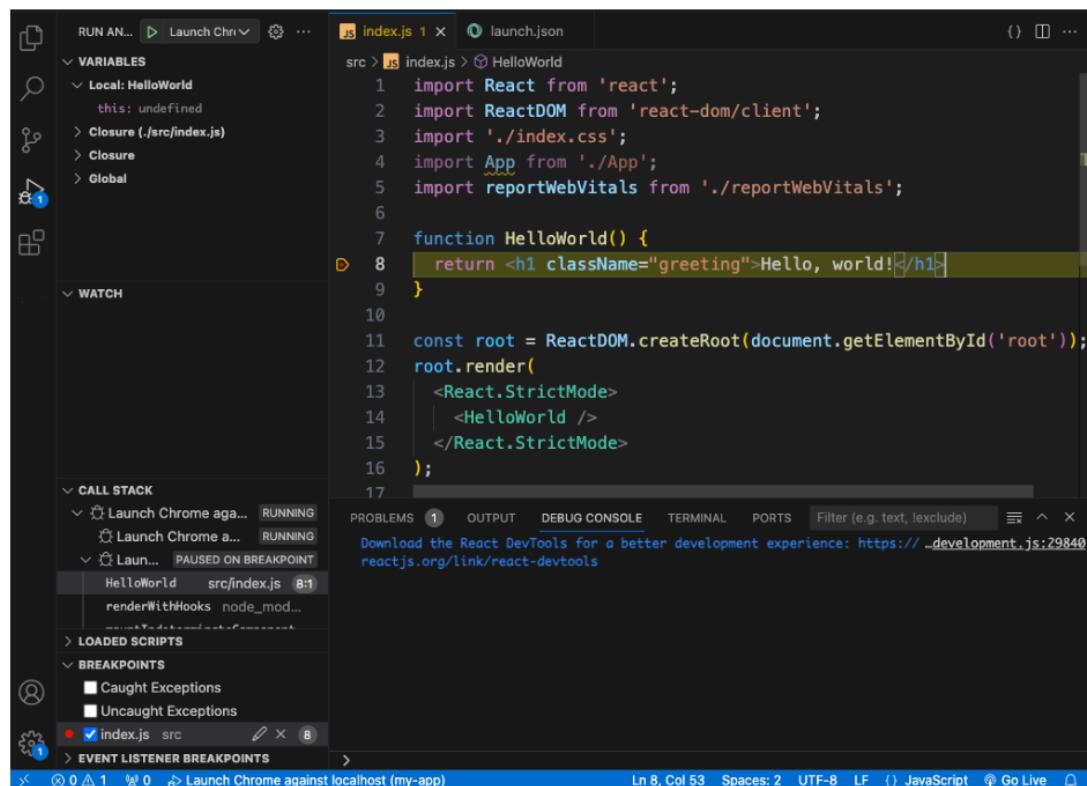
```
{
  "version": "0.2.0",
  "configurations": [
    {
      "type": "msedge",
      "request": "launch",
      "name": "Launch Edge against localhost",
      "url": "http://localhost:3000",
      "webRoot": "${workspaceFolder}"
```

```
}
```

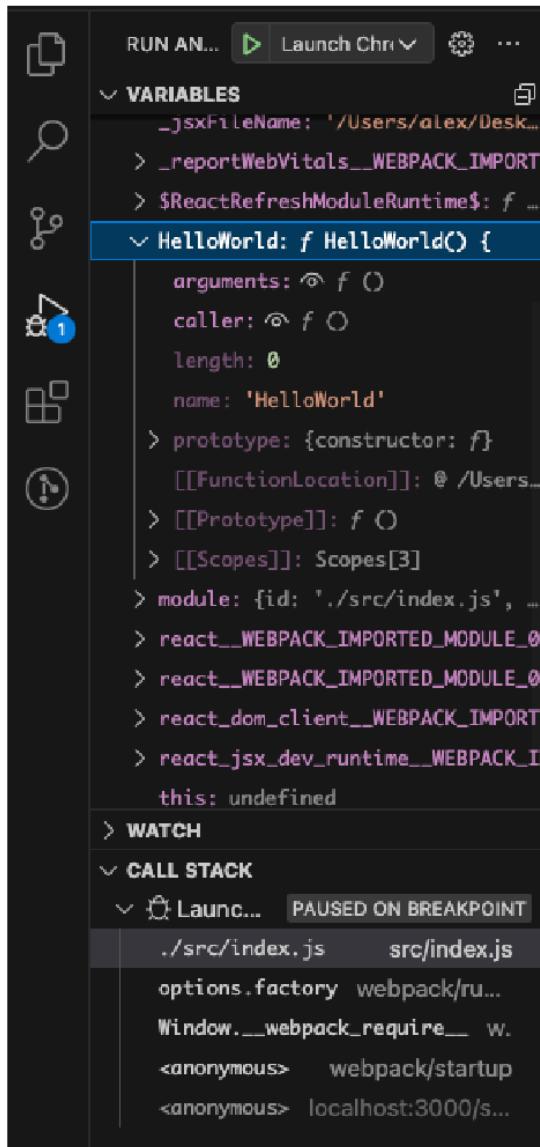
```
]
```

```
}
```

Ensure that your development server is running (`npm start`). Then press F5 or the green arrow to launch the debugger and open a new browser instance. The source code where the breakpoint is set runs on startup before the debugger was attached, so we won't hit the breakpoint until we refresh the web page. Refresh the page and you should hit your breakpoint.



You can step through your source code (F10), inspect variables such as `HelloWorld`, and see the call stack of the client side React application.



For more information about the debugger and its available options, check out our documentation on [browser debugging](#).

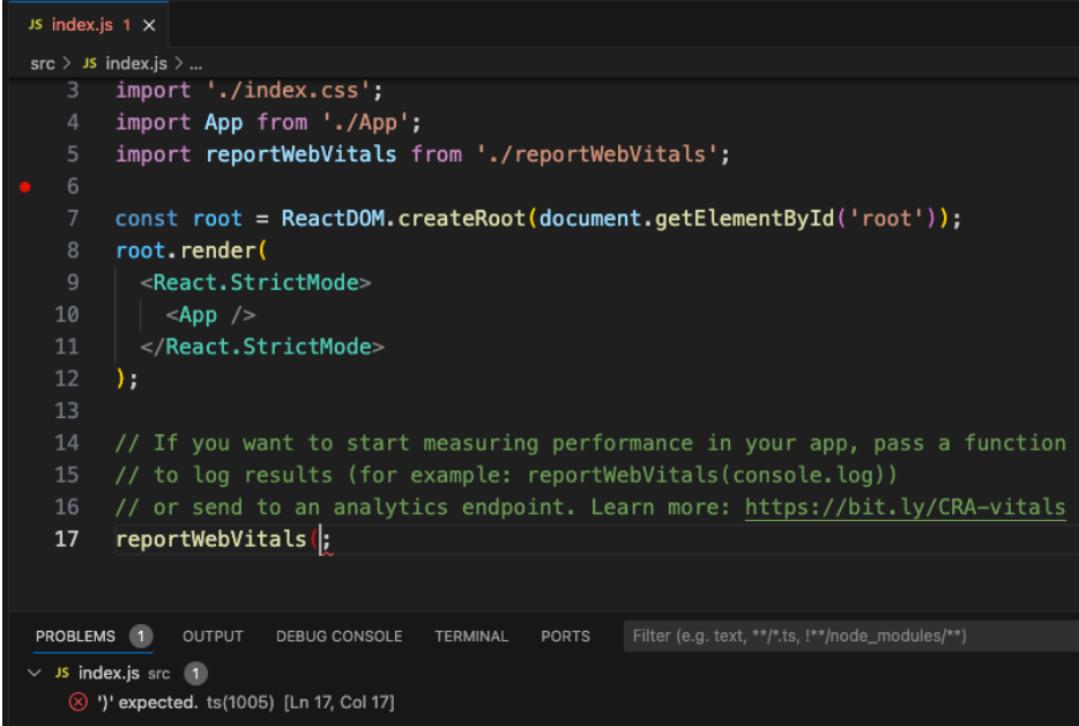
[Live editing and debugging](#)

If you are using [webpack](#) together with your React app, you can have a more efficient workflow by taking advantage of webpack's HMR mechanism which enables you to have live editing and debugging directly from VS Code. You can learn more in this [Live edit and debug your React apps directly from VS Code](#) blog post and the [webpack Hot Module Replacement documentation](#).

Linting

Linters analyze your source code and can warn you about potential problems before you run your application. The JavaScript language services included with VS Code has syntax error checking support by default, which you can see in action in the **Problems** panel (**View > Problems** **Ctrl+Shift+M**).

Try making a small error in your React source code and you'll see a red squiggle and an error in the **Problems** panel.



The screenshot shows the VS Code interface with the following details:

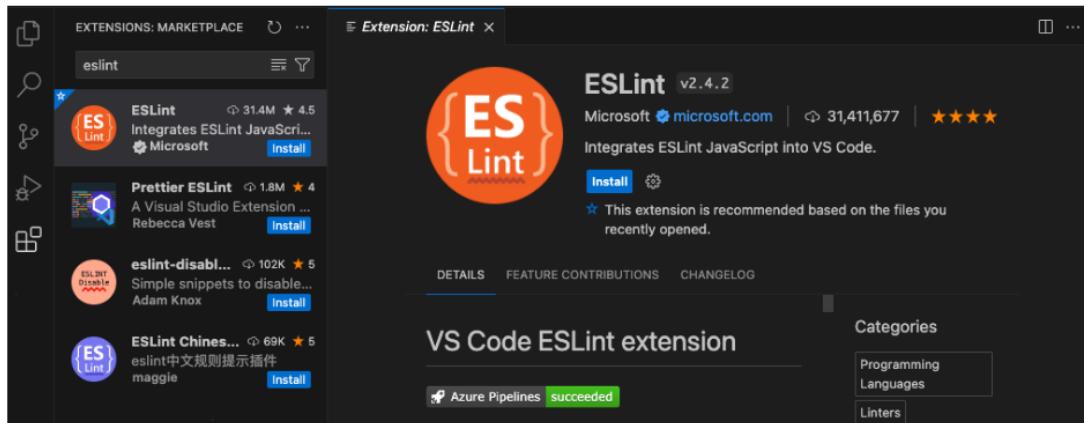
- Editor:** The code editor shows a file named `index.js` with 1 error. The code is a simple React application setup.
- Problems Panel:** The **PROBLEMS** tab is active, showing 1 error. The error message is: `✖ ')' expected. ts(1005) [Ln 17, Col 17]`.
- Bottom Status Bar:** The status bar displays the file path `src/index.js` and the error count `1`.

Linters can provide more sophisticated analysis, enforcing coding conventions and detecting anti-patterns. A popular JavaScript linter is [ESLint](#). ESLint, when combined with the ESLint VS Code [extension](#), provides a great in-product linting experience.

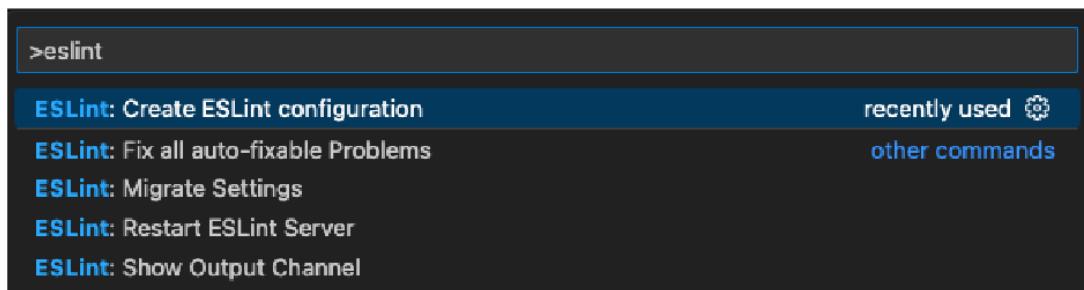
First, install the ESLint command-line tool:

```
npm install -g eslint
```

Then install the ESLint extension by going to the **Extensions** view and typing 'eslint'.



Once the ESLint extension is installed and VS Code reloaded, you'll want to create an ESLint configuration file, `.eslintrc.js`. You can create one using the extension's **ESLint: Create ESLint configuration** command from the **Command Palette** (`Ctrl+Shift+P`).



The command will prompt you to answer a series of questions in the **Terminal** panel. Take the defaults, and it will create a `.eslintrc.js` file in your project root that looks something like this:

```
module.exports = {

  env: {

    browser: true,
    es2020: true
  },
  extends: ['eslint:recommended', 'plugin:react/recommended'],
  parserOptions: {
    ecmaFeatures: {
      jsx: true
    },
  }
};
```

ecmaVersion: 11,

```
    sourceType: 'module'

  },
  plugins: ['react'],
  rules: {}

};
```

ESLint will now analyze open files and shows a warning in `index.js` about 'App' being defined but never used.

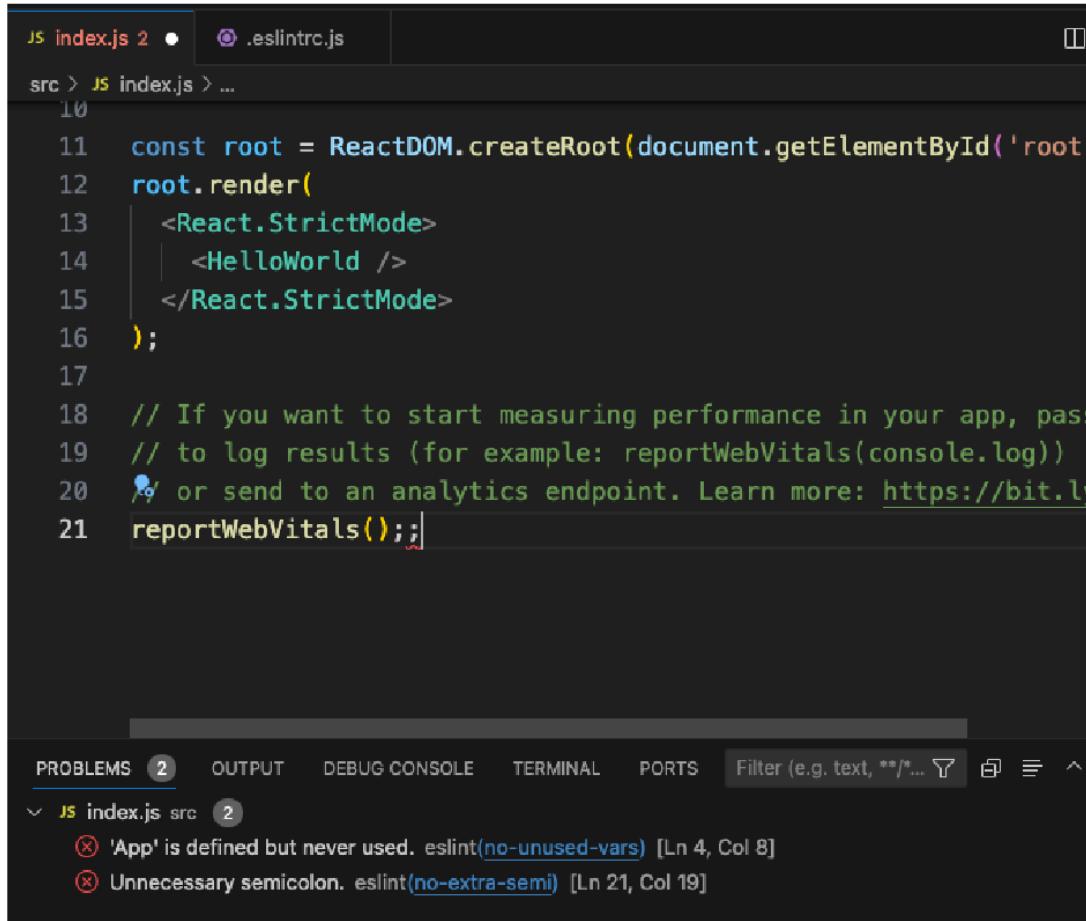
```
1 import React from 'react';
2 import ReactDOM from 'react-dom/client';
3 import './index.css';
4 import App from './App';
5 import reportWebVitals from './reportWebVitals';
6
7 function HelloWorld() {
8   return <h1 className="greeting">Hello, world!</h1>
9 }
10
11 const root = ReactDOM.createRoot(document.getElementById('root'));
12 root.render(
13   <React.StrictMode>
14     <HelloWorld />
15   </React.StrictMode>
16 );
17
```

You can modify the ESLint rules in the `.eslintrc.js` file.

Let's add an error rule for extra semi-colons

```
"rules": {  
    "no-extra-semi": "error"  
}
```

Now when you mistakenly have multiple semicolons on a line, you'll see an error (red squiggle) in the editor and error entry in the **Problems** panel.



```
JS index.js 2 • .eslintrc.js
src > JS index.js > ...
10
11  const root = ReactDOM.createRoot(document.getElementById('root'));
12  root.render(
13    <React.StrictMode>
14      <HelloWorld />
15    </React.StrictMode>
16  );
17
18 // If you want to start measuring performance in your app, pass
19 // to log results (for example: reportWebVitals(console.log))
20 // or send to an analytics endpoint. Learn more: https://bit.ly
21 reportWebVitals();
```

PROBLEMS 2 OUTPUT DEBUG CONSOLE TERMINAL PORTS Filter (e.g. text, **/*...)

- ✓ **index.js** src 2
 - ✖ 'App' is defined but never used. eslint([no-unused-vars](#)) [Ln 4, Col 8]
 - ✖ Unnecessary semicolon. eslint([no-extra-semi](#)) [Ln 21, Col 19]

Popular Starter Kits

In this tutorial, we used the `create-react-app` generator to create a simple React application. There are lots of great samples and starter kits available to help build your first React application.

VS Code React Sample

This is a [sample](#) React application, which creates a simple TODO application and includes the source code for a Node.js [Express](#) server. It also shows how to use the [Babel](#) ES6 transpiler and then use [webpack](#) to bundle the site assets.

TypeScript React

If you're curious about TypeScript and React, you can also create a TypeScript version of the `create-react-app` application by specifying that you want to use the TypeScript template:

```
npx create-react-app my-app --template typescript
```

See the details at [Adding TypeScript](#) on the [Create React App site](#).

[Angular](#)

[Angular](#) is another popular web framework. If you'd like to see an example of Angular working with VS Code, check out the [Debugging with Angular CLI](#) recipe. It will walk you through creating an Angular application and configuring the `launch.json` file for the JavaScript debugger

