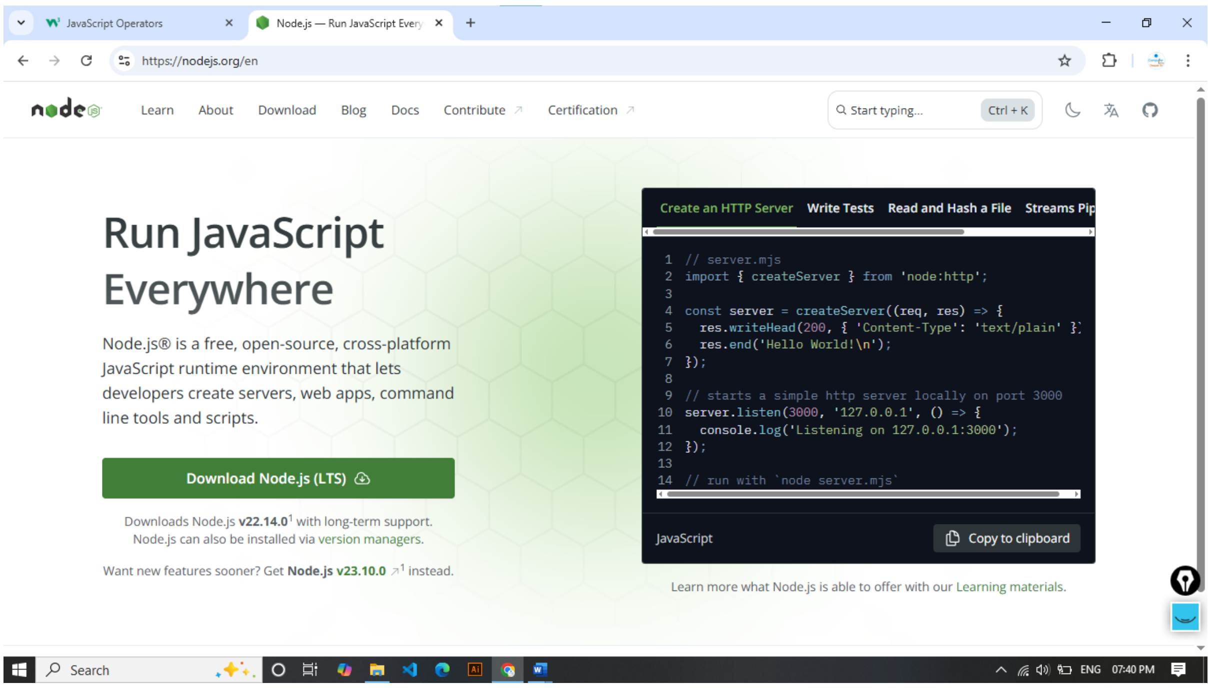
Introduction to React.js

- React.js is a JavaScript library used for building user interfaces (UIs) and single-page applications.
- Created by **Jordan Walke** at **Facebook.**
- Most popular JavaScript library for frontend development.

How to Download and Install Nodejs



Run JavaScript Everywhere

Node.js® is a free, open-source, cross-platform JavaScript runtime environment that lets developers create servers, web apps, command line tools and scripts.

Download Node.js (LTS)

Downloads Node.js **v22.14.0**[1] with long-term support.
Node.js can also be installed via version managers.

Want new features sooner? Get **Node.js v23.10.0**[1] instead.

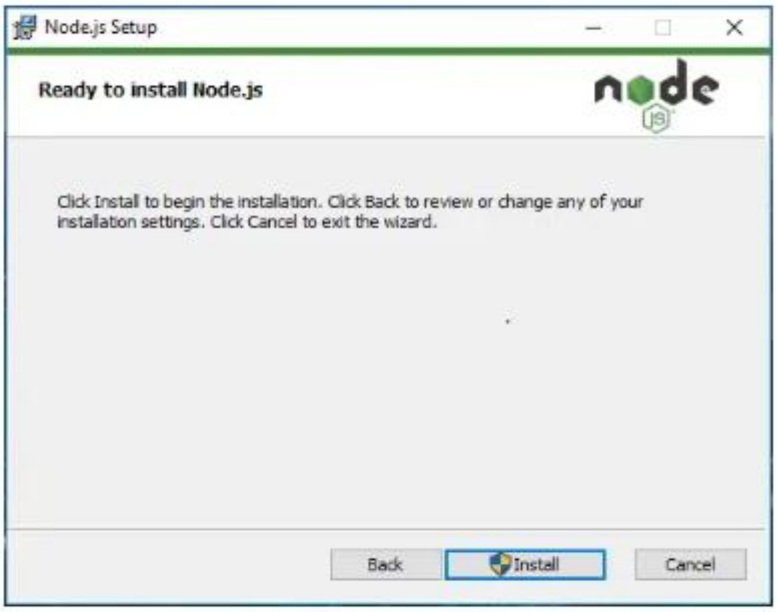
```
// server.mjs
import { createServer } from 'node:http';

const server = createServer((req, res) => {
  res.writeHead(200, { 'Content-Type': 'text/plain' }
  res.end('Hello World!\n');
});

// starts a simple http server locally on port 3000
server.listen(3000, '127.0.0.1', () => {
  console.log('Listening on 127.0.0.1:3000');
});

// run with `node server.mjs`
```

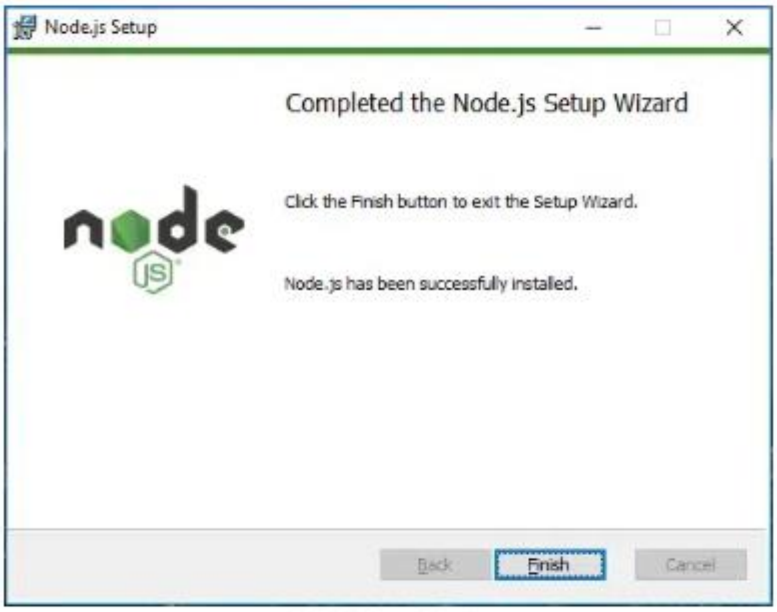
Learn more what Node.js is able to offer with our Learning materials.



Ready to install Node.js

Click Install to begin the installation. Click Back to review or change any of your installation settings. Click Cancel to exit the wizard.

Finish the setup


Node.js Setup

Completed the Node.js Setup Wizard

Click the Finish button to exit the Setup Wizard.

Node.js has been successfully installed.

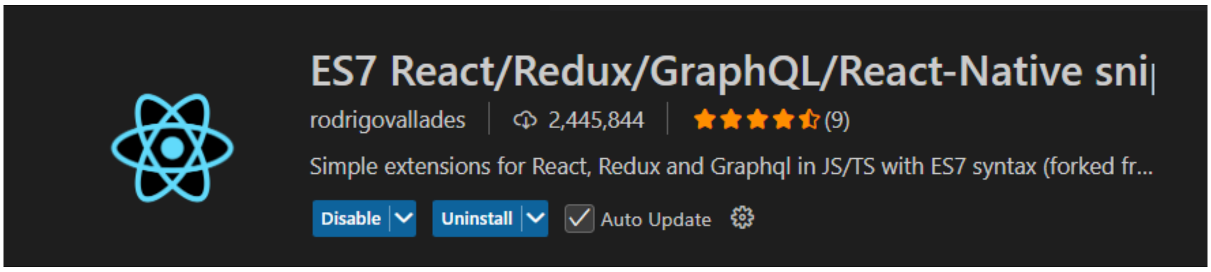
Nodejs Installation

Verify the Installation

- Type **node -v** and press Enter to check the Node.js version.
- Type **npm -v** and press Enter to check the npm version.
- Both commands should return version numbers, confirming successful installation.

Vscode Setup :

Download Extensions : **ES7 React/Redux/GraphQL/React-Native snippets**


ES7 React/Redux/GraphQL/React-Native sni

rodrigovallades | 2,445,844 | (9)

Simple extensions for React, Redux and Graphql in JS/TS with ES7 syntax (forked fr...

learn before React

HTML

CSS

JAVASCRIPT

Create React App:

Create react app : npm create vite@latest

- Project name
- React
- Javascript
- cd <ProjectName>
- npm install
- npm run dev

Install Bun : **Bun is an npm-compatible package manager.**

npm i -g bun

create react app using bun

bun create vite

- Project name
- React
- Javascript
- cd <ProjectName>
- bun install
- bun dev

Error in Vscode

```
bun : File C:\Users\HP\AppData\Roaming\npm\bun.ps1 cannot be loaded because running scripts is disabled on this system. For more information, see

about_Execution_Policies at https:/go.microsoft.com/fwlink/?LinkID=135170.

At line:1 char:1

+ bun create vite

+ ~

    + CategoryInfo          : SecurityError: (:) [], PSSecurityException

    + FullyQualifiedErrorId : UnauthorizedAccess fix this
```

Open windows powershell

1. Get-ExecutionPolicy
2. Set-ExecutionPolicy RemoteSigned -Scope CurrentUser
3. Get-ExecutionPolicy

Project Structure

- **node_modules**
    - This is the folder which contains all the **necessary libraries & dependencies by React.js.**
    - You can ignore this folder completely.
- **public**
    - This folder contains all **static files** like images, videos, fonts, etc.
- **src**
    - This **folder contains all source files** (The source directory—here your React components, JavaScript files, and CSS are stored).
    - **App.jsx**
        - The main React component that acts as the root component.
        - There is also App.css with it, which contains styles for this component.
    - **main.jsx**
        - This is the **entry point to our React.js project**, which renders the App component.

Naming

- **camelCase**
  - It is used for variables, functions/methods, properties inside objects, file names, etc.
  - Capitalization of each word **except the first** is done.
- **PascalCase**
  - It is used for component names, class names, types, etc.
  - Capitalization of **each word** is done.
- **snake_case**
  - It is not common in JavaScript but is used heavily in Python.
  - Each word is separated by “_” and is in **small letters**.
- **kebab-case**
  - It is common for file names, CSS classes, IDs, etc.
  - Each word is separated by **hyphen (-)**.

JSX & Rendering Elements :

What is JSX?

- JSX is a syntax extension for JavaScript, similar to HTML.
- It allows writing HTML elements inside JavaScript.

Components in React :

Components in React are reusable and independent pieces of code that render specific parts of a user interface. They act as building blocks, allowing developers to divide complex UIs into manageable and testable units

Types of Components

- Functional Components
- Class Components

Functional Component Example

```
Welcome() {
        return <h1>Welcome to React!</h1>;
}
export default Welcome;
```

React Fragments :

React Fragments are a feature that allow grouping multiple elements without adding an extra node to the DOM.

```
function MyComponent() {
return (
         <>
                 <h1>Hello</h1>
                <p>World</p>
        </>
);
}
```

Props in React

What are Props?

- Props (short for properties) allow passing data between components.

```
function Greeting(props) {
    return <h1>Hello, {props.name}!</h1>;
   }

 function App() {
    return <Greeting name="PIET" />;
   }
```

Handling Events in React

Event Handling in React

- Events in React are handled similarly to DOM events but follow a camelCase convention.
- Use the onClick, onChange, onSubmit, etc., attributes.

```
function ButtonClick() {
        function handleClick() {
                alert("Button Clicked!");
        }
   return <button onClick={handleClick}>Click Me</button>;
 }
```

State in React

What is State?

- State is used to manage component data dynamically.

Example (Using useState Hook)

```
import { useState } from 'react';
function Counter() {
const [count, setCount] = useState(0);
 return (
<div>
<p>Count: {count}</p>
<button onClick={() => setCount(count + 1)}>Increment</button>
</div>
 );
}
```

Conditional Rendering in React

What is Conditional Rendering?

- Rendering components or elements based on conditions.
- Use if-else, ternary operators, or logical && operators.

```
function Greeting(props) {
       return props.isLoggedIn ? <h1>Welcome Back!</h1> : <h1>Please Sign In</h1>;
 }
```

Lists and Keys in React

Rendering Lists

- Use map() to render arrays dynamically.
- Use a unique key prop for better performance.

```
const items = ['Apple', 'Banana', 'Cherry'];
function ItemList() {
    return (
    <ul>
    {items.map((item, index) => (
    <li key={index}>{item}</li>
    ))}
    </ul>
    );
    }
```

Forms in React

Controlled Components

- Forms in React use **state** to control input values.
- onChange event updates state as the user types.

Example: Controlled Input

```
    import { useState } from "react";
    function Form() {
            const [name, setName] = useState("");
            return (
                    <div>
                    <input  type="text" value={name}
                   onChange={(e) => setName(e.target.value)}  />
            <p>Hello, {name}!</p>
             </div>
    );
    }
```

React Hooks

Common Hooks

- useState(): Manage state.
- useEffect(): Handle side effects.
- useContext(): Share state across components.

Example: useEffect Hook

```
import { useState, useEffect } from "react";
function Timer() {
  const [time, setTime] = useState(0);
  useEffect(() => {
    const interval = setInterval(() => setTime((prev) => prev + 1), 1000);
    return () => clearInterval(interval); // Cleanup
  }, []);

  return <p>Time: {time}s</p>;
}
```

React Router (Navigation in React)

What is React Router?

- React Router is used to handle navigation in a React app.
- It allows single-page applications (SPA) to have multiple views.

Installation

```
npm install react-router-dom
```

Basic Example

```
import { BrowserRouter as Router, Routes, Route, Link } from "react-router-dom";

function Home() {
  return <h2>Home Page</h2>;
}

function About() {
  return <h2>About Page</h2>;
}

function App() {
  return (
    <Router>
      <nav>
        <Link to="/">Home</Link> | <Link to="/about">About</Link>
      </nav>
      <Routes>
        <Route path="/" element={<Home />} />
        <Route path="/about" element={<About />} />
      </Routes>
    </Router>
  );
}

export default App;
```

Context API (State Management in React)

What is Context API?

- A built-in way to manage state without props drilling.
- Provides **global state management**.

Steps to Use Context API

1. Create a Context
2. Provide Context
3. Consume Context

Example: Using Context API

```
import { createContext, useContext, useState } from "react";

const UserContext = createContext();

function UserProvider({ children }) {
  const [user, setUser] = useState("Deepak");
  return <UserContext.Provider value={user}>{children}</UserContext.Provider>;
}

function DisplayUser() {
  const user = useContext(UserContext);
  return <h1>User: {user}</h1>;
}

function App() {
  return (
    <UserProvider>
      <DisplayUser />
    </UserProvider>
  );
}

export default App;
```

Fetching API Data in React

Using Fetch API

```
import { useEffect, useState } from "react";

function FetchData() {
  const [data, setData] = useState([]);

  useEffect(() => {
   fetch("https://jsonplaceholder.typicode.com/posts")
    .then((response) => response.json())
    .then((json) => setData(json));
  }, []);

  return (
   <div>
    <h2>Posts</h2>
    {data.slice(0, 5).map((post) => (
     <p key={post.id}>{post.title}</p>
    ))}
   </div>
  );
}

export default FetchData;
```

Axios for API Requests

Why use Axios?

- Simplifies HTTP requests.
- Handles errors better than Fetch API.

Installation

```
npm install axios
```

Example: Fetching API Data using Axios

```
import axios from "axios";
import { useEffect, useState } from "react";

function FetchUsers() {
  const [users, setUsers] = useState([]);

  useEffect(() => {
    axios.get("https://jsonplaceholder.typicode.com/users")
      .then((response) => setUsers(response.data))
      .catch((error) => console.error("Error fetching data:", error));
  }, []);

  return (
    <div>
      <h2>User List</h2>
      {users.map((user) => (
        <p key={user.id}>{user.name}</p>
      ))}
    </div>
  );
}

export default FetchUsers;
```