React Training

Presented By

Bibhuranjan Mohanty

SESSION-02

Objectives

- Creating First React App
- Project structure and folder organization
- Writing your first React component
- JSX – an Introduction
- Rendering of Elements and Components in React

React is used for building most of the user interfaces today

Creating First React App

Different Ways to Create React App

1. Using Create React App (CRA)

❑ Recommended for beginners and quick prototypes.

❑ It is the simplest way to start a new React project without manual configuration.

❑ **Pros:** Easy setup, pre-configured with best practices, includes build scripts, hot reloading, and linting.

❑ **Command**

```
npx create-react-app my-app
cd my-app
npm start
```

❑ Create React App Documentation

Node Package Manager (npm)
vs.
Node Package Execute(npx)

npm will install the package & then executes It

Whereas

npx directly executes & then uninstalls that package

```
npm init react-app    projectName
```

```
npx create-react-app  projectName
```

Different Ways to Create React App

2. Using Vite

- ❑ For faster builds and a more modern development experience.
- ❑ Vite is a build tool that is much faster than CRA, providing a more efficient developer experience.
- ❑ **Pros:** Fast startup, lightweight, optimized for development with instant hot module replacement.
- ❑**Command**

```
npm create vite@latest my-app --template react
cd my-app
npm install
npm run dev
```

- ❑Vite Documentation

VITE

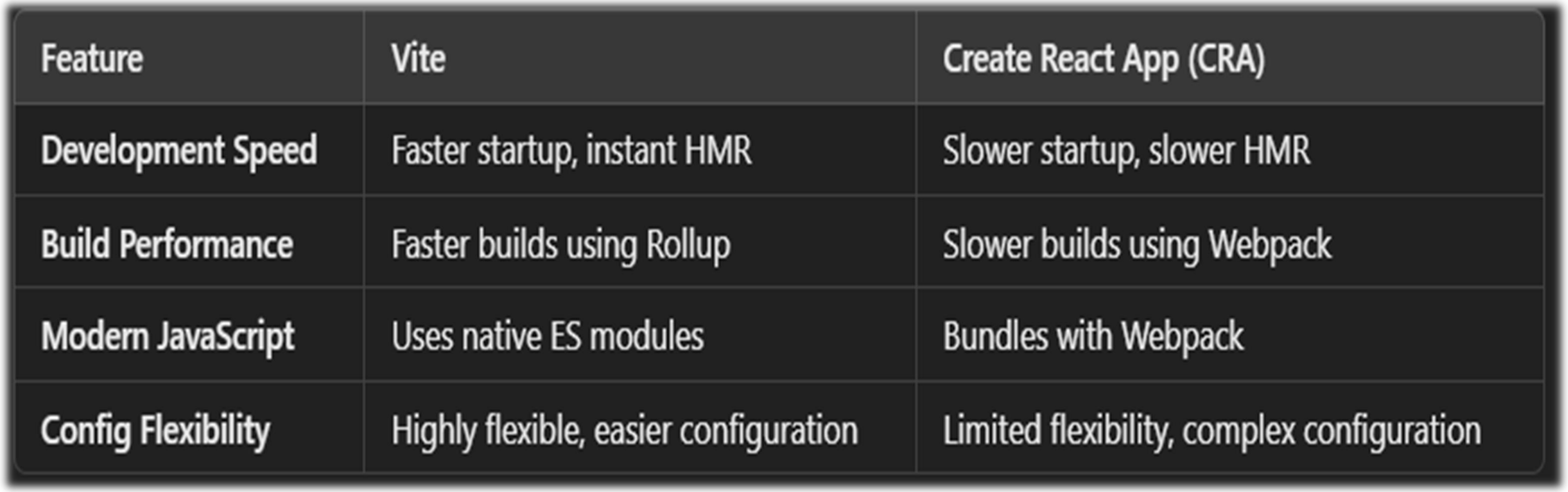
- ❑ Vite is a modern front-end build tool designed for fast and efficient development.

- ❑ Created by Evan You (the creator of Vue.js), it's optimized for modern frameworks like React, Vue, and Svelte.

Features of Vite:

- ❑ **Instant Server Start:** Extremely fast development server using native ES modules.
- ❑ **Hot Module Replacement (HMR):** Instant updates without full-page reload.
- ❑ **Fast Builds:** Leverages Rollup for fast and optimized production builds.
- ❑ **Out-of-the-Box Support:** Works seamlessly with modern JavaScript frameworks (React, Vue, Svelte).

Advantages of Vite over Create React App (CRA):

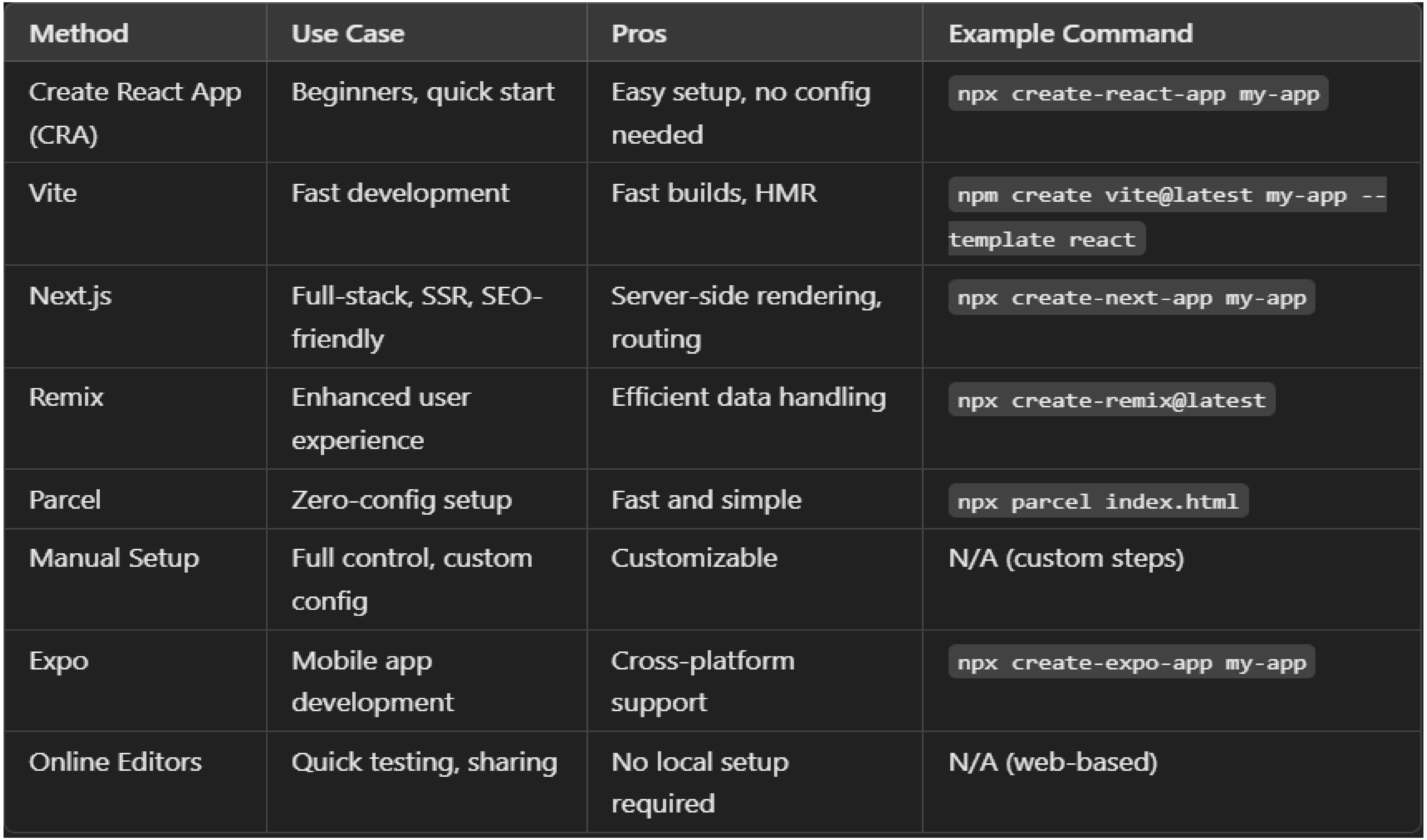


Feature	Vite	Create React App (CRA)
Development Speed	Faster startup, instant HMR	Slower startup, slower HMR
Build Performance	Faster builds using Rollup	Slower builds using Webpack
Modern JavaScript	Uses native ES modules	Bundles with Webpack
Config Flexibility	Highly flexible, easier configuration	Limited flexibility, complex configuration

Why Choose Vite?

❑ Ideal for modern projects requiring **fast development and build times**.

❑ Uses **native ES module imports**, reducing bundling complexity.



Method	Use Case	Pros	Example Command
Create React App (CRA)	Beginners, quick start	Easy setup, no config needed	`npx create-react-app my-app`
Vite	Fast development	Fast builds, HMR	`npm create vite@latest my-app --template react`
Next.js	Full-stack, SSR, SEO-friendly	Server-side rendering, routing	`npx create-next-app my-app`
Remix	Enhanced user experience	Efficient data handling	`npx create-remix@latest`
Parcel	Zero-config setup	Fast and simple	`npx parcel index.html`
Manual Setup	Full control, custom config	Customizable	N/A (custom steps)
Expo	Mobile app development	Cross-platform support	`npx create-expo-app my-app`
Online Editors	Quick testing, sharing	No local setup required	N/A (web-based)

What is create-react-app ?

Create-react-app

Command-line interface (CLI)

helps to create

prerequisite environment

for a react project.

How do we create a react application or react Project?

Earlier

```
npm install create-react-app
```

```
Create-react-app
```

Now

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

Or

```
npm init react-app my-app
```

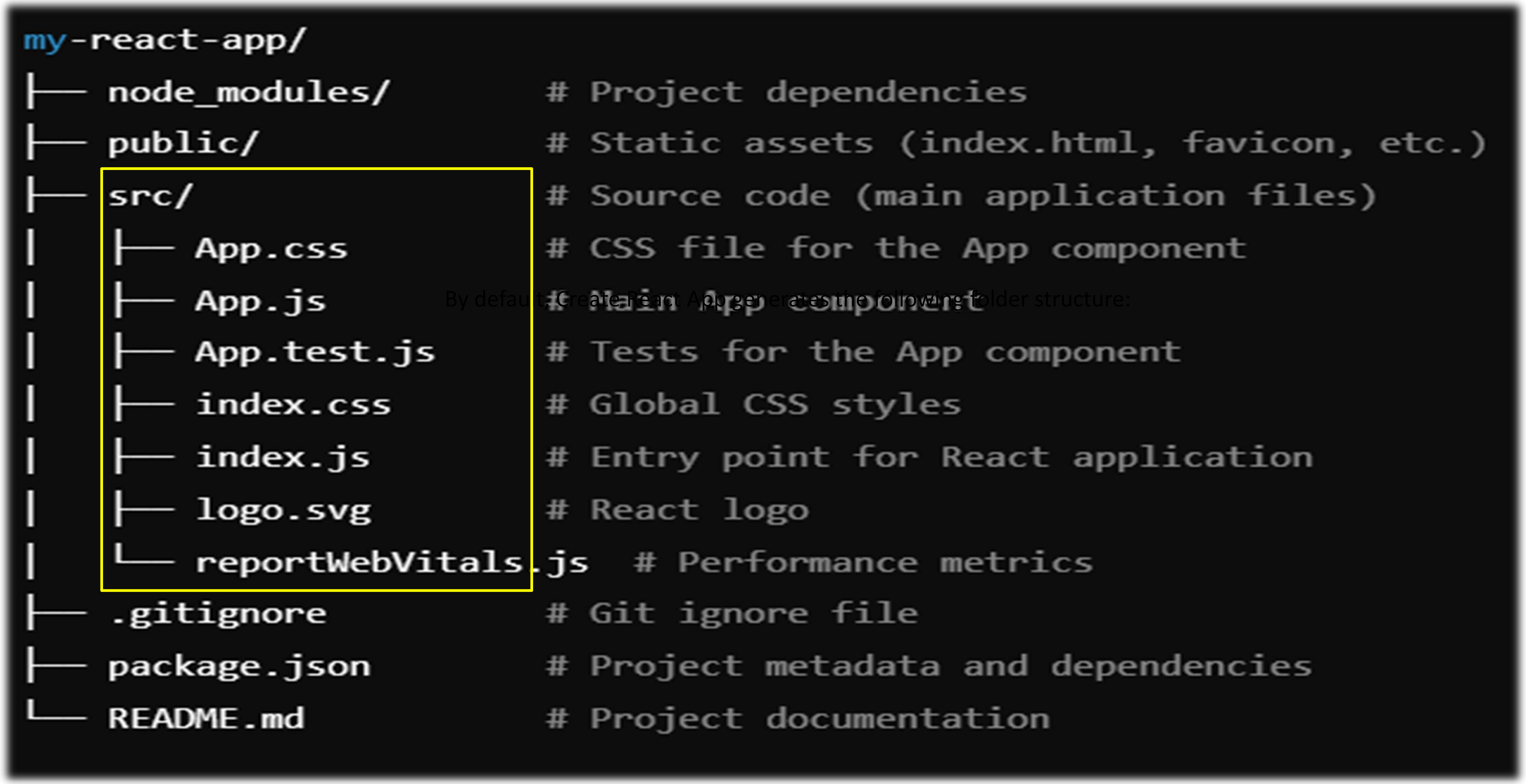
Or

```
npm create vite@latest proj --template react
```

Project structure and folder organization

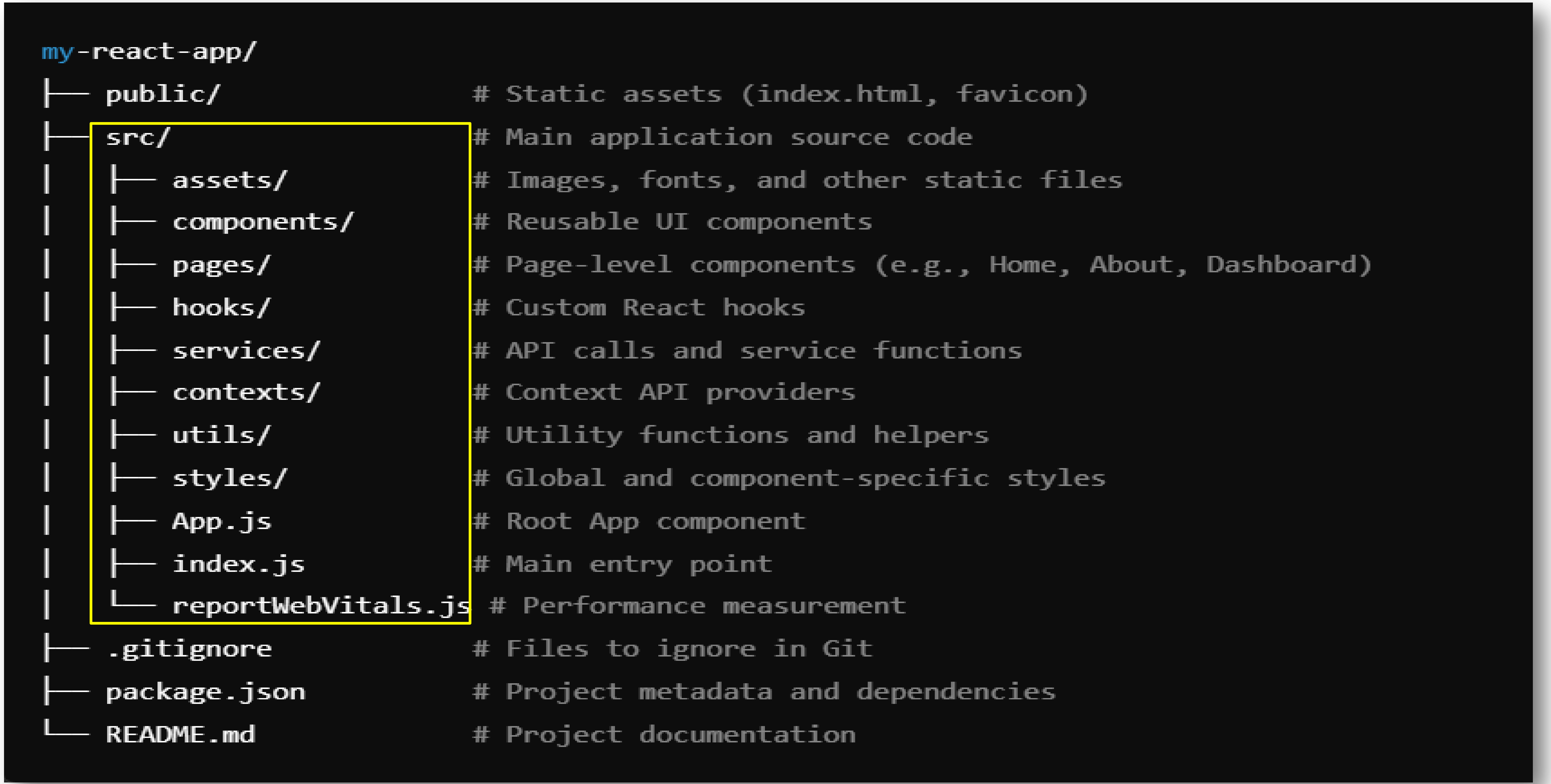
Default Project Structure

By default, Create React App generates the following folder structure:



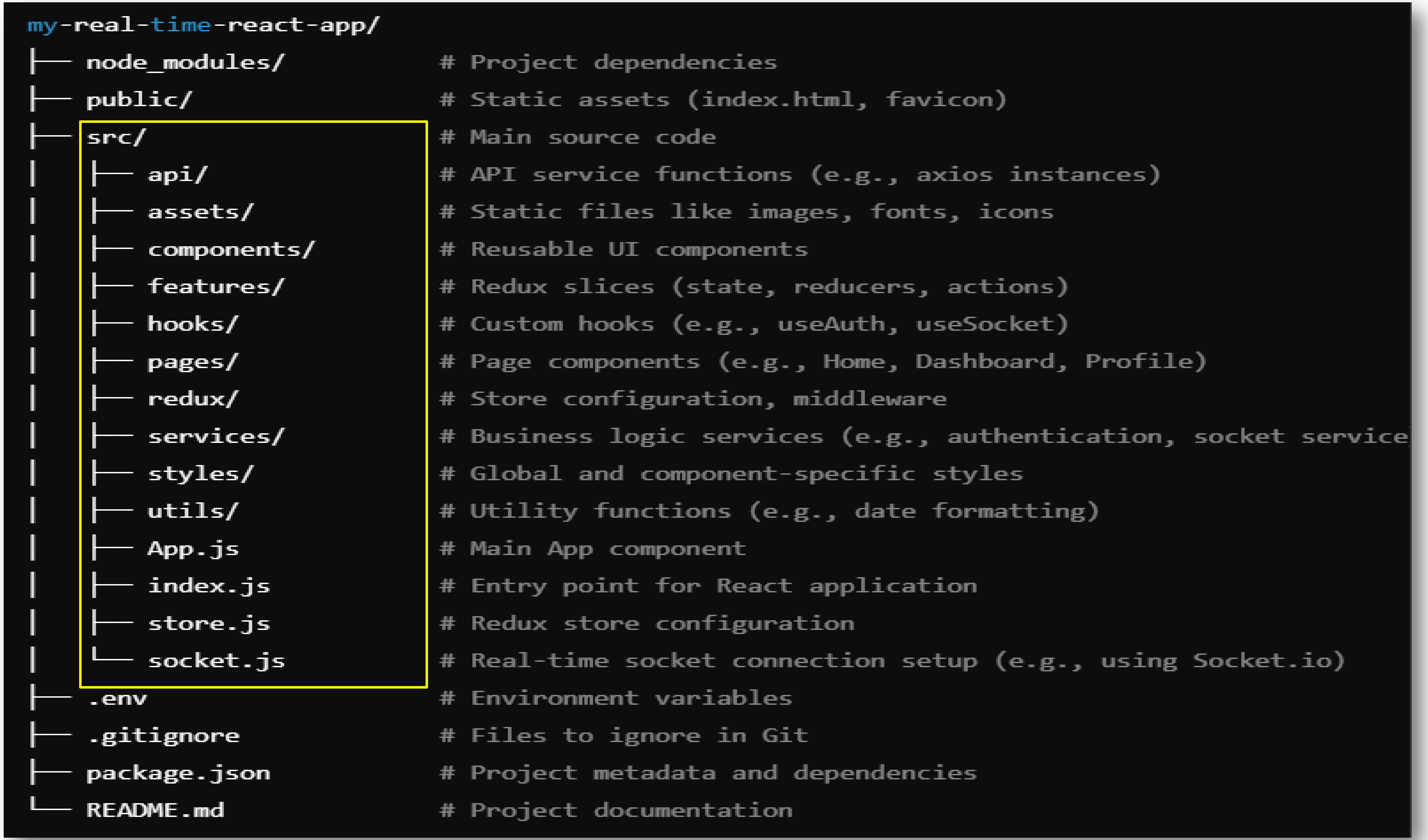
```
my-react-app/
├── node_modules/         # Project dependencies
├── public/               # Static assets (index.html, favicon, etc.)
├── src/                  # Source code (main application files)
│   ├── App.css           # CSS file for the App component
│   ├── App.js            # Main App component
│   ├── App.test.js       # Tests for the App component
│   ├── index.css         # Global CSS styles
│   ├── index.js          # Entry point for React application
│   ├── logo.svg          # React logo
│   └── reportWebVitals.js  # Performance metrics
├── .gitignore            # Git ignore file
├── package.json          # Project metadata and dependencies
└── README.md             # Project documentation
```

Enhanced Folder Structure for Better Organization



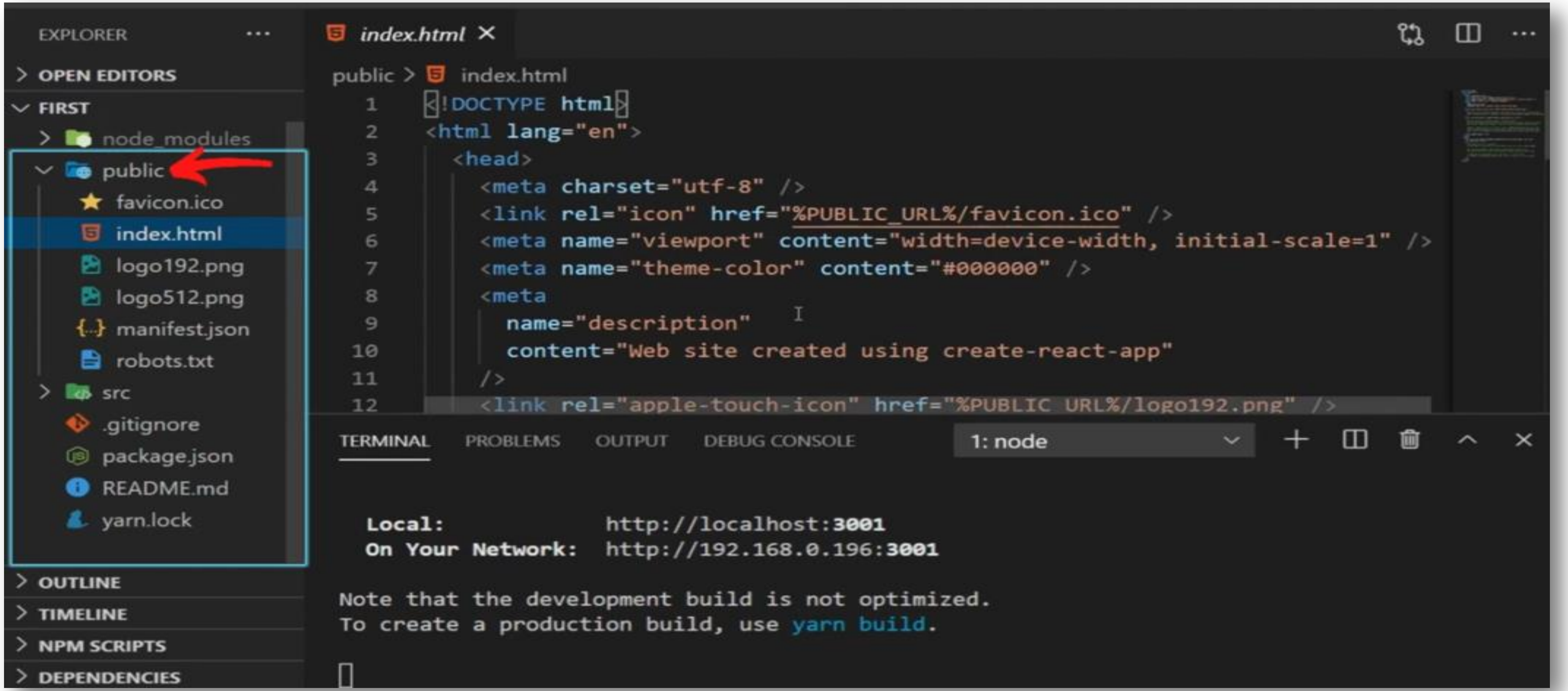
```
my-react-app/
├── public/              # Static assets (index.html, favicon)
├── src/                 # Main application source code
│   ├── assets/          # Images, fonts, and other static files
│   ├── components/      # Reusable UI components
│   ├── pages/           # Page-level components (e.g., Home, About, Dashboard)
│   ├── hooks/           # Custom React hooks
│   ├── services/        # API calls and service functions
│   ├── contexts/        # Context API providers
│   ├── utils/           # Utility functions and helpers
│   ├── styles/          # Global and component-specific styles
│   ├── App.js           # Root App component
│   ├── index.js         # Main entry point
│   └── reportWebVitals.js # Performance measurement
├── .gitignore           # Files to ignore in Git
├── package.json         # Project metadata and dependencies
└── README.md            # Project documentation
```

Detailed folder structure and organization for a real-time React app using Redux



```
my-real-time-react-app/
├── node_modules/          # Project dependencies
├── public/                # Static assets (index.html, favicon)
├── src/                   # Main source code
│   ├── api/               # API service functions (e.g., axios instances)
│   ├── assets/            # Static files like images, fonts, icons
│   ├── components/        # Reusable UI components
│   ├── features/          # Redux slices (state, reducers, actions)
│   ├── hooks/             # Custom hooks (e.g., useAuth, useSocket)
│   ├── pages/             # Page components (e.g., Home, Dashboard, Profile)
│   ├── redux/             # Store configuration, middleware
│   ├── services/          # Business logic services (e.g., authentication, socket service
│   ├── styles/            # Global and component-specific styles
│   ├── utils/             # Utility functions (e.g., date formatting)
│   ├── App.js             # Main App component
│   ├── index.js           # Entry point for React application
│   ├── store.js           # Redux store configuration
│   └── socket.js          # Real-time socket connection setup (e.g., using Socket.io)
├── .env                   # Environment variables
├── .gitignore             # Files to ignore in Git
├── package.json           # Project metadata and dependencies
└── README.md              # Project documentation
```

Where does the index.html reside in the default react project?



```html
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="utf-8" />
    <link rel="icon" href="%PUBLIC_URL%/favicon.ico" />
    <meta name="viewport" content="width=device-width, initial-scale=1" />
    <meta name="theme-color" content="#000000" />
    <meta
      name="description"
      content="Web site created using create-react-app"
    />
    <link rel="apple-touch-icon" href="%PUBLIC_URL%/logo192.png" />
```

```
  Local:            http://localhost:3001
  On Your Network:  http://192.168.0.196:3001

Note that the development build is not optimized.
To create a production build, use yarn build.
```

Writing your first React component.

Component.

Component is a reusable & independent piece of code .

Creating a Component :

```
function Welcome(){

  return (<h1>Welcome to React World</h1>) ;

}
```

Rendering a Component :

```
<Welcome/>                <Welcome><Welcome/>
```

Import - Export

Import:

```
import   Welcome  from    "./Welcome.jsx";
```

Default Export :

```
export default Welcome ;
```

Named Export :

```
export default Welcome ;
```

```
Import {Title} from ""./Welcome.jsx"
```

Named exports are useful when you want to export multiple values and import them with their specific names, while default exports are handy for exporting a single value and giving it a custom name when importing .The choice between the two depends on the structure and requirements of your codebase.

JSX – an Introduction

Writing Markup in JSX

1. Return a single root element
2. Close all the tags
3. camelCase most of the things

React Fragment

Fragment let you group a list of children without adding extra nodes to the DOM.

JSX with Curly Braces

```
function App( ){

  return (<p>5 * 5  = {  2 * 2 }</p>) ;

}
```

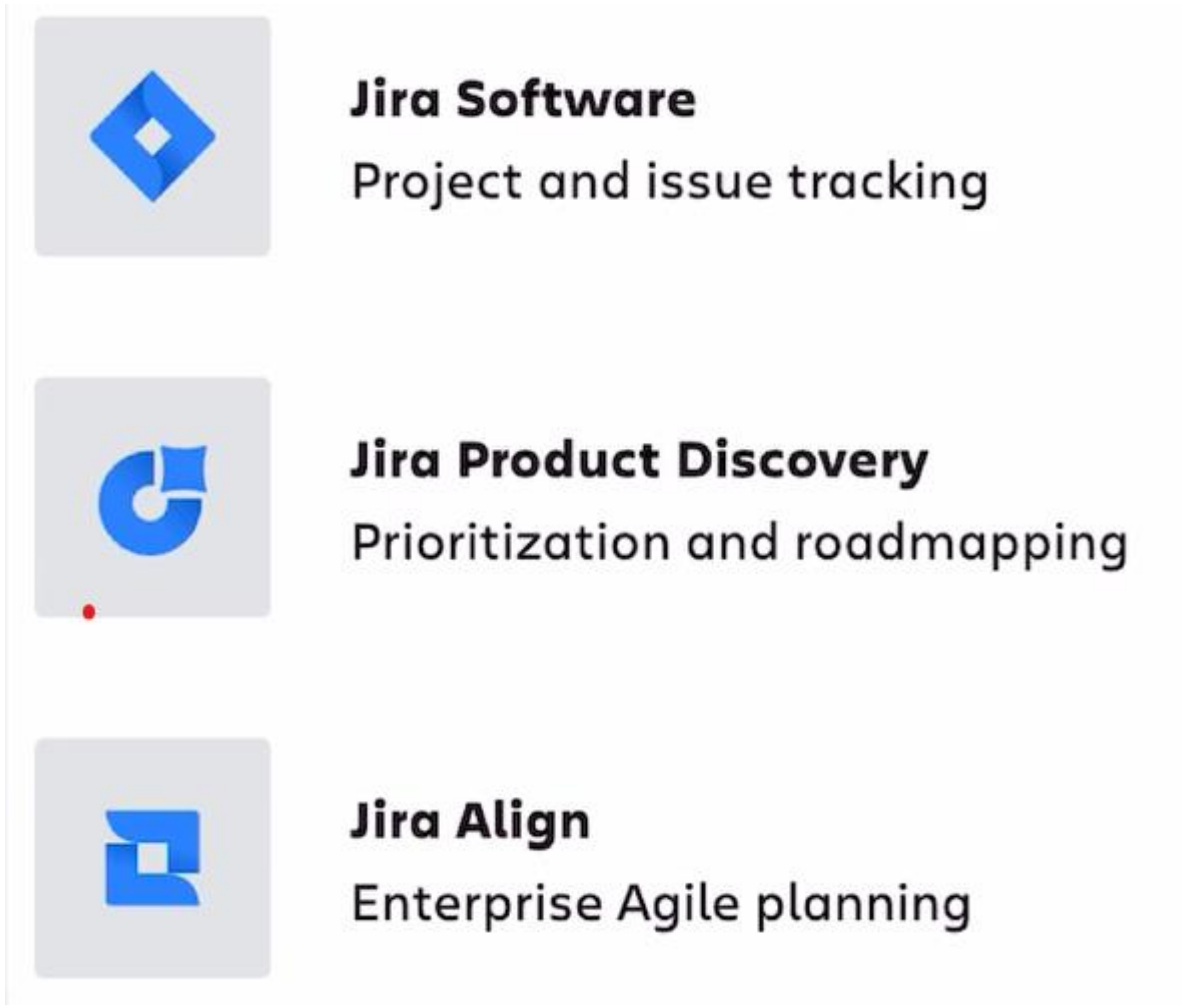
```
function App( ){

let  msg =“hello world !”;

  return (<p>message says {msg}</p>) ;

}
```

Structuring Componets



Jira Software
Project and issue tracking

Jira Product Discovery
Prioritization and roadmapping

Jira Align
Enterprise Agile planning

Case Study

Let’s Check Our Understanding So far !

Quiz

1. What's a special feature of JSX, not found in XML or XHTML?

A) JSX can use JavaScript expressions

B) JSX can return multiple elements

C) HTML attributes and event references do not have to be written in camelCase while writing JSX

D) All of the above

Quiz

2. When interpreted by the system, JSX elements are transformed into ______.

A) React.createElement() calls

B) Objects

C) appendChild() calls

D) None of the above

Quiz

3. True or False: Multi-line JSX has to be surrounded by parentheses.

A) True

B) False

Quiz

4. What symbol should be used to denote JavaScript expressions within JSX?

A) @

B) $

C) {}

D) #

Quiz

5. True or False: Comments in JSX are written using <!-- This is a comment --> notation.

A) True

B) False

Quiz

6. True or False: Calling **React.createElement()** triggers a DOM operation.

A) True

B) False

Quiz

7. True or False: React elements are simple objects.

A) True

B) False

Quiz

8. What is the name of the node that a React element needs to render on?

A) main node

B) root node

C) Render node

D) All of the above

Quiz

9. How many root nodes will you typically have in a React application?

A) 2

B) 0

C) 3

D) 1

Quiz

9. How many root nodes will you typically have in a React application?

A) 2

B) 0

C) 3

D) 1

Quiz

10. True or False: The name of a React component should start with a capital letter.

A) True

B) False

11. True or False: React components can contain other components.

A) True

B) False

Q & A

Thank You