**Create React App**

https://create-react-app.dev/

We don’t need to install or configure tools like webpack or Babel. They are preconfigured and hidden so that you can focus on the code. We’ll need to have Node >= 8.10 on our local development machine (but it’s not required on the server).

npx

npx create-react-app my-app

npm

npm init react-app my-app

yarn

yarn create react-app my-app

cd my-app

npm start

npm test

npm run build

**Selecting a template:-**

You can now optionally start a new app from a template by appending --template [template-name] to the creation command.

npx create-react-app my-app --template [template-name]  
  
[Show All Templates](https://www.npmjs.com/search?q=cra-template-*)

**Supported Browsers:-**

By default, the generated project supports all modern browsers. Support for Internet Explorer 9, 10, and 11 requires polyfills.

Install react-app-polyfill:-

npm install react-app-polyfill

yarn add react-app-polyfill

// This must be the first line in src/index.js

import 'react-app-polyfill/ie9';

// This must be the first line in src/index.js ‘

import 'react-app-polyfill/ie11';

**Best Editor:-**

Visual Studio Code and WebStorm. Debugging feature is currently only supported by these.

<https://code.visualstudio.com/>

<https://marketplace.visualstudio.com/items?itemName=msjsdiag.debugger-for-chrome>

<https://www.jetbrains.com/webstorm/>

<https://chrome.google.com/webstore/detail/jetbrains-ide-support/hmhgeddbohgjknpmjagkdomcpobmllji>

**Adding a Sass Stylesheet:-**

npm install node-sass –save

yarn add node-sass

Now We can rename src/App.css to src/App.scss and update src/App.js to import src/App.scss. This file and any other file will be automatically compiled if imported with the extension .scss or .sass.

To share variables between Sass files, you can use Sass imports. For example, src/App.scss and other component style files could include @import "./shared.scss"; with variable definitions.

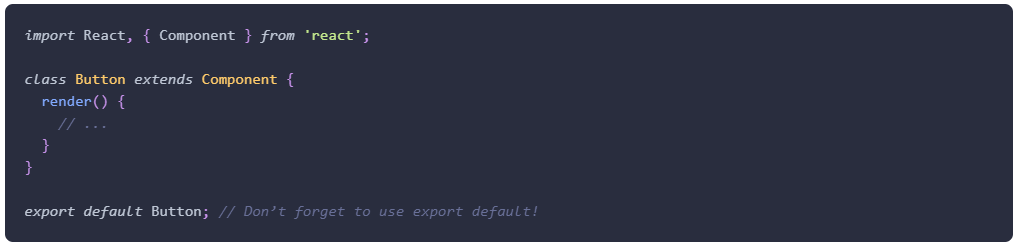
**Adding Routing :-**

Create React App doesn't prescribe a specific routing solution, but React Router is the most popular one. Note that you may need to configure your production server to support client-side routing before deploying your app.

npm install --save react-router-dom

yarn add react-router-dom

**Importing a Component:-**

****

**Adding Bootstrap:-**

**React Bootstrap** is the most popular option that strives for complete parity with Bootstrap. **reactstrap** is also a good choice for projects looking for smaller builds at the expense of some features.

<https://react-bootstrap.netlify.app/getting-started/introduction>

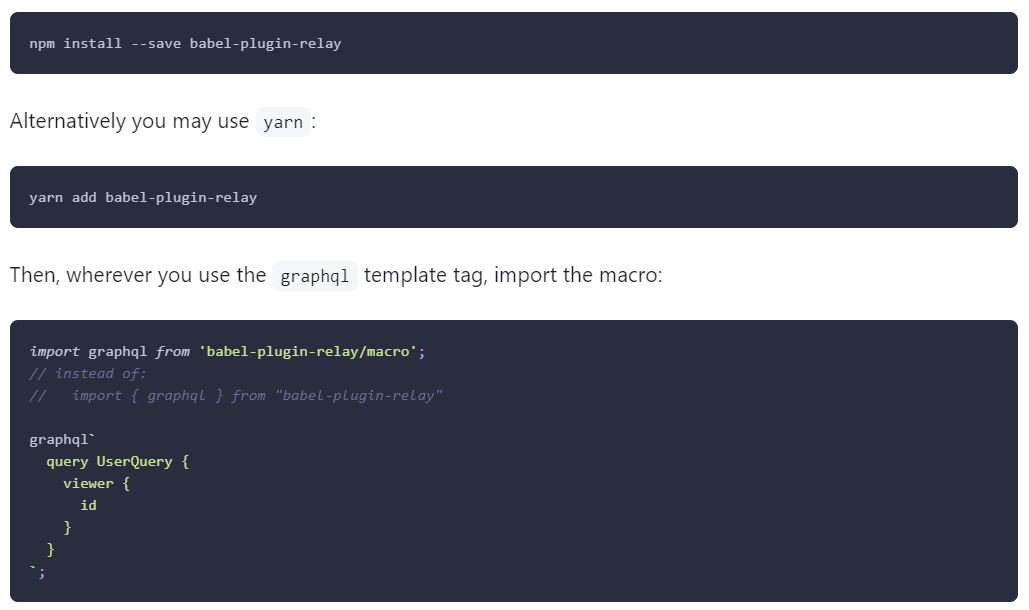
<https://reactstrap.github.io/>

**Adding TypeScript:-**  
npx create-react-app my-app --template typescript

yarn create react-app my-app --template typescript

**Adding Relay:-**

Relay is a framework for building data-driven React applications powered by GraphQL. The current release of Relay works with Create React App projects out of the box using Babel Macros. Set up your project as shown in the Relay documentation, then make sure you have a version of the babel plugin providing the macro.



**Running Tests:-**

Create React App uses **Jest** as its test runner. Jest is a Node-based runner. This means that the tests always run in a Node environment and not in a real browser.

**Title and Meta Tags:-**

You can find the source HTML file in the public folder of the generated project. You may edit the <title> tag in it to change the title from “React App” to anything else.

If you need to dynamically update the page title based on the content, you can use the browser document.title API. For more complex scenarios when you want to change the title from React components, you can use **React Helmet**, a third party library.

<https://create-react-app.dev/docs/title-and-meta-tags/>

<https://github.com/nfl/react-helmet>