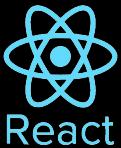




# Introduction to React

- What is **React**?
- Working of **DOM**
- Problems with JS
- Working of **React**
- JS Vs **React**
- Intro to **Components**

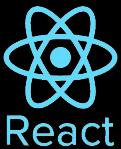




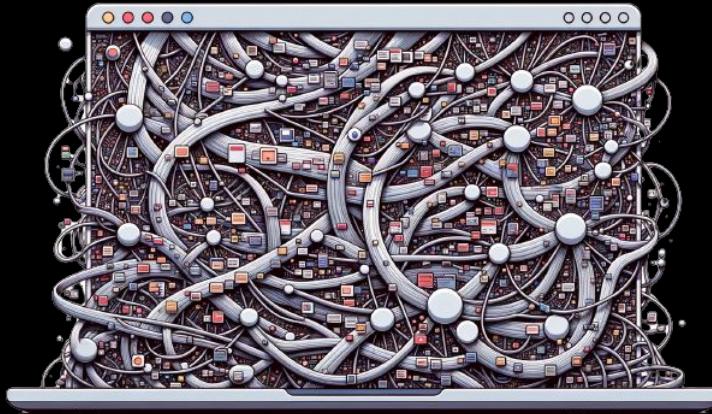
# What is React



1. JavaScript library to build Dynamic and interactive user interfaces
2. Developed at Facebook in 2011.
3. Currently most widely used JS library for front-end development.
4. Used to create single page application (page does not re-load).



# Working of DOM



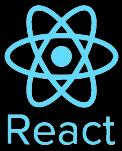
1. Browser takes **HTML** and create **DOM**.
2. **JS** helps us modify **DOM** based on **user actions** or events.
3. In **big applications**, Working with **DOM** becomes complicated.



# Problems with JavaScript

1. React has a simpler mental model
2. JS is cumbersome
3. JS is Error-prone
4. JS is Hard to maintain





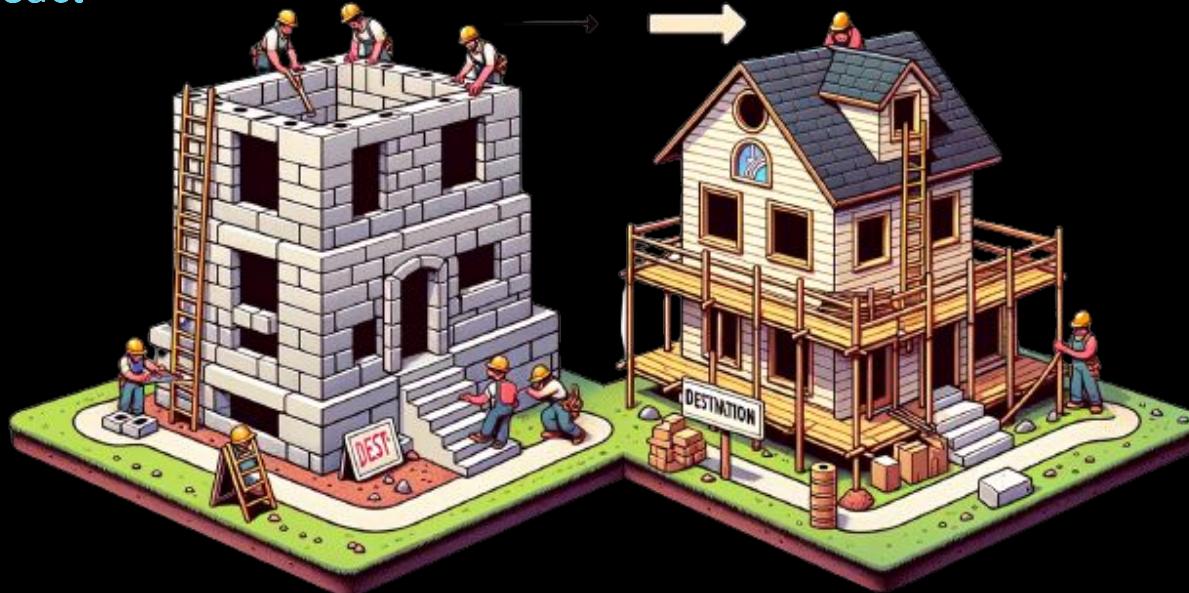
# Working of React



1. No need to worry about **querying** and **updating** DOM elements.
2. React creates a web page with **small** and **reusable** components
3. React will take care of **creating** and **updating** DOM elements.
4. IT saves a lot of time, **cheezein aasan hai, pahele se likhi hui hain**



# JS Vs React



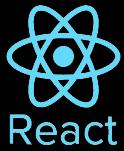
1. JS is **imperative**: You define **steps** to reach your **desired state**.
2. React is **Declarative**: You define the **target UI state** and then react figures out how to reach that state.



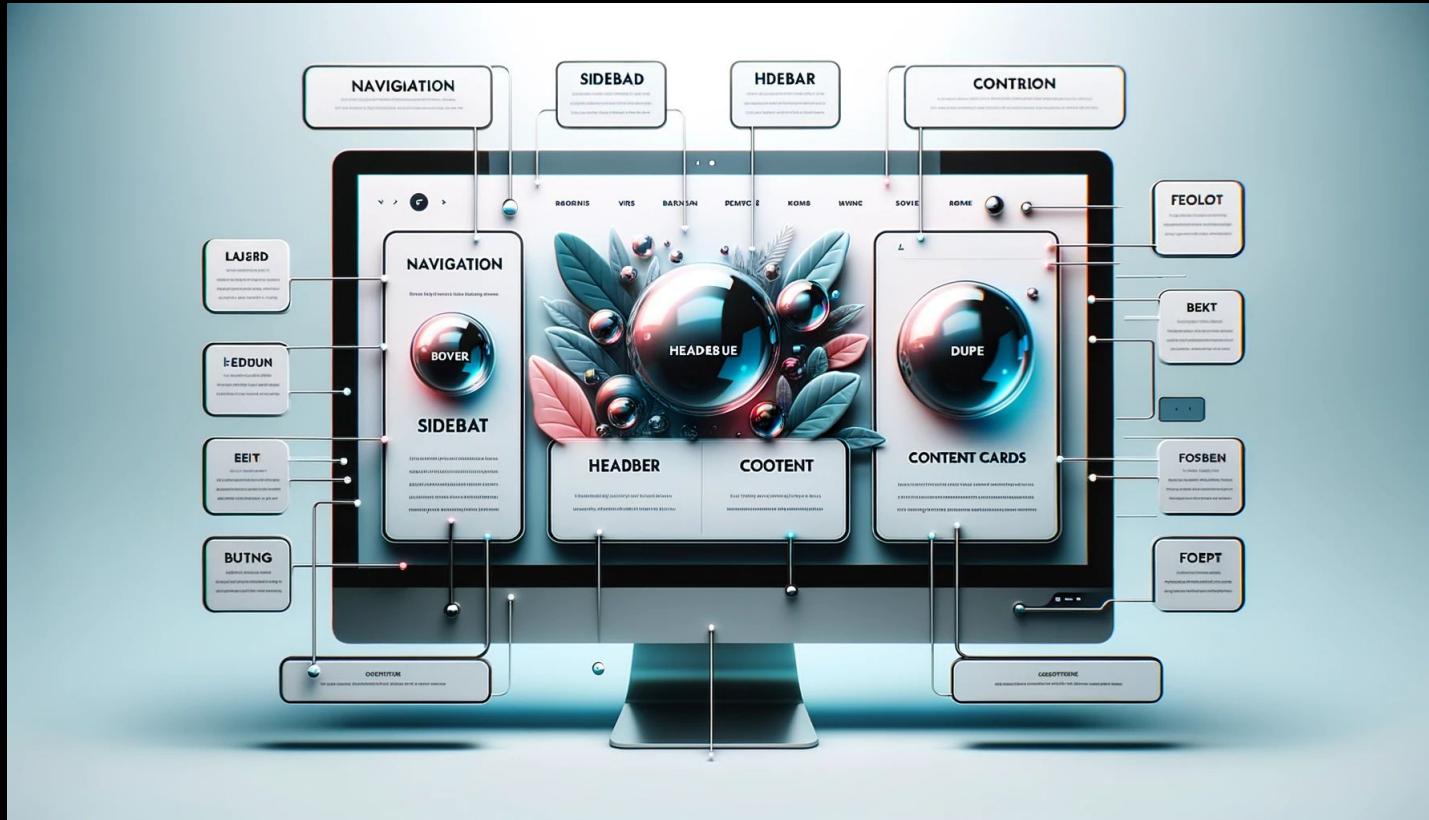
# Introduction to Components

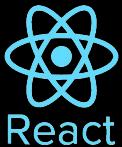


Components help us write reusable, modular and better organized code.

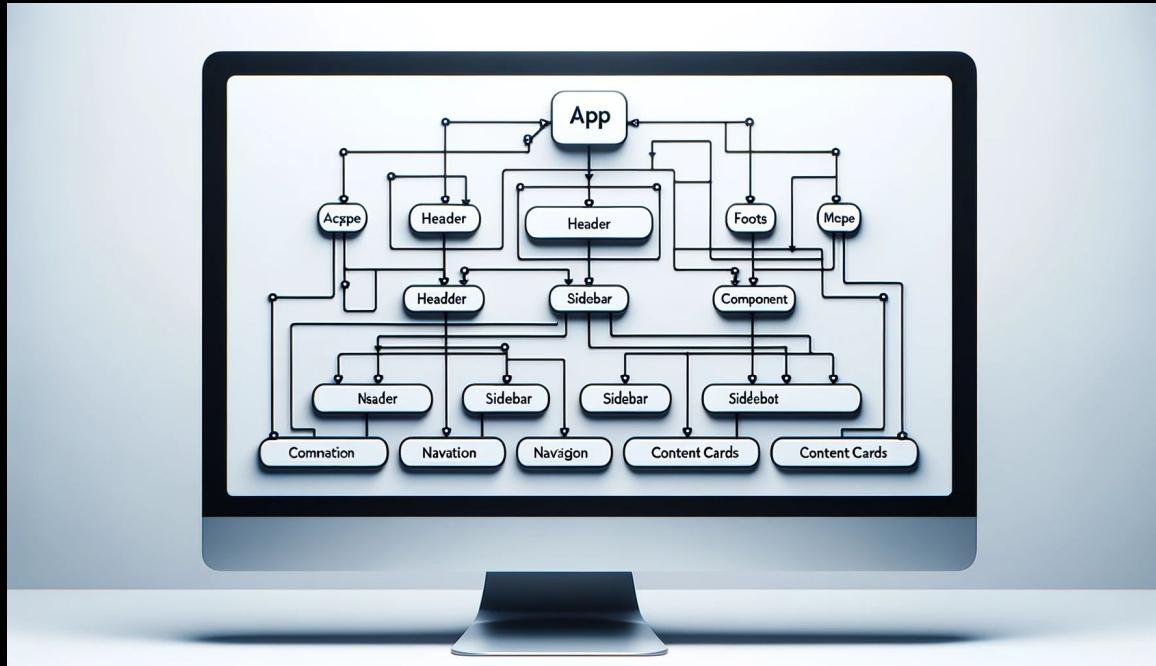


# Introduction to Components

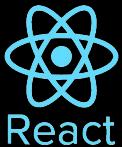




# Introduction to Components



React application is a tree of components with App Component as the root bringing everything together.



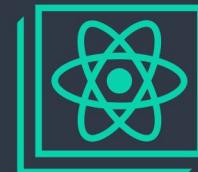
React

# Create a React App

Official tool is CRA(Create React APP)

## Quick Start

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

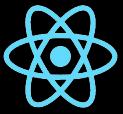


## Create React App

Set up a modern web app by running one command.

[Get Started](#)

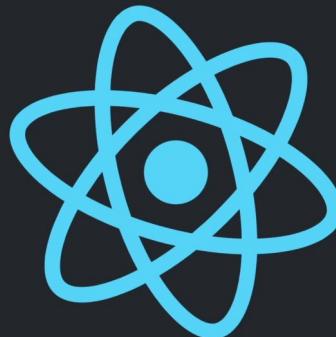
- **npm:** You need to install a package first (using npm install) before you can use it.
- **npx:** You can directly run a package without installing it, using npx <package-name>.



React

```
> node_modules
└─ public
    ★ favicon.ico
    ▷ index.html
    🖼 logo192.png
    🖼 logo512.png
    { } manifest.json
    ⏵ robots.txt
└─ src
    # App.css
    JS App.js
    JS App.test.js
    # index.css
    JS index.js
    🖼 logo.svg
    JS reportWebVitals.js
    JS setupTests.js
    ♦ .gitignore
    { } package-lock.json
    { } package.json
    ⓘ README.md
```

# Project Structure

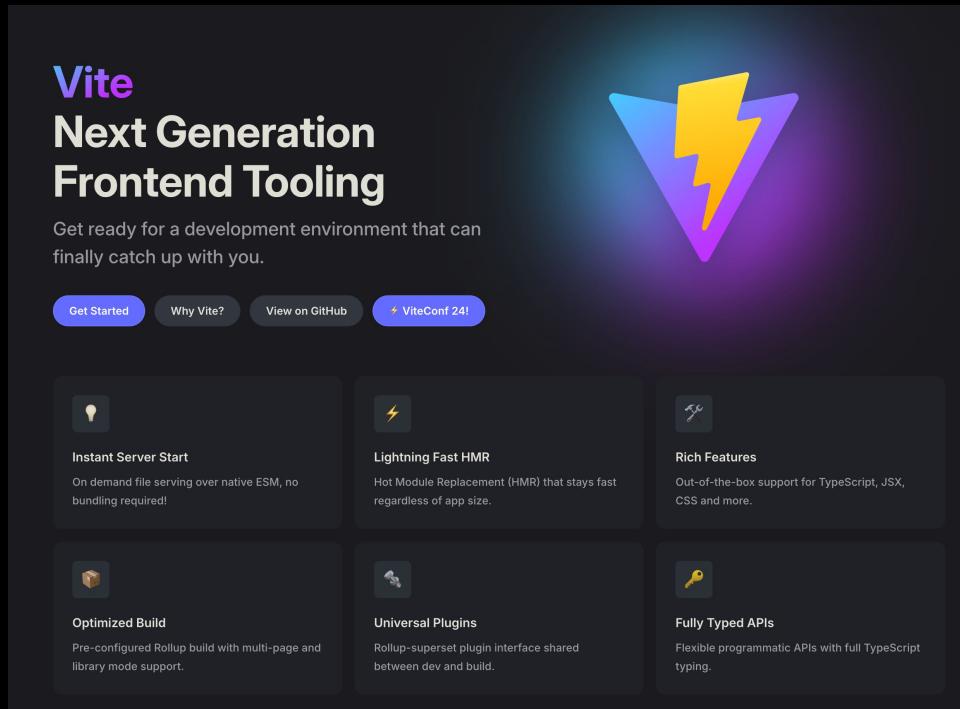


Edit `src/App.js` and save to reload.

[Learn React](#)



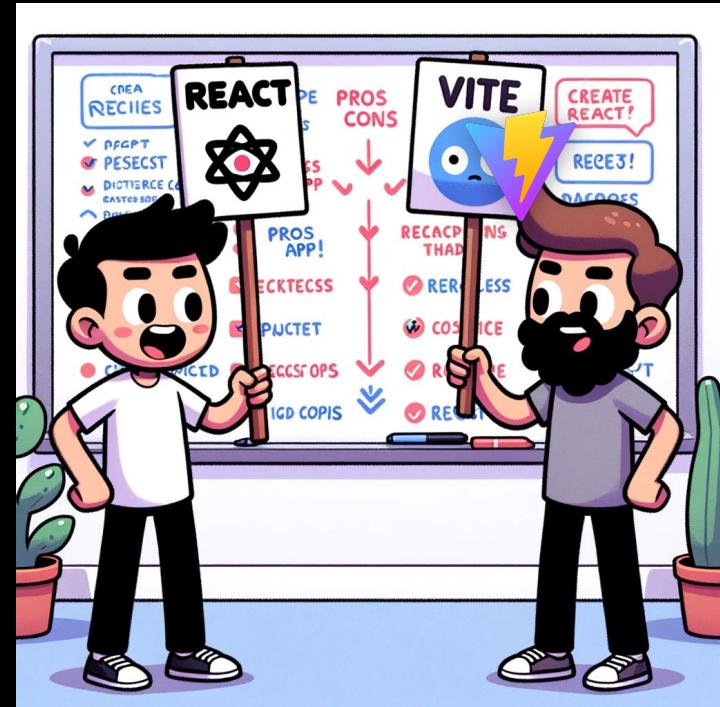
# Create a React App



The screenshot shows the Vite homepage. At the top, it says "Vite Next Generation Frontend Tooling". Below that is a large yellow lightning bolt icon. A sub-headline reads "Get ready for a development environment that can finally catch up with you." There are four main features listed: "Instant Server Start", "Lightning Fast HMR", "Rich Features", and "Optimized Build". Each feature has a small icon and a brief description.

- Instant Server Start**  
On demand file serving over native ESM, no bundling required!
- Lightning Fast HMR**  
Hot Module Replacement (HMR) that stays fast regardless of app size.
- Rich Features**  
Out-of-the-box support for TypeScript, JSX, CSS and more.
- Optimized Build**  
Pre-configured Rollup build with multi-page and library mode support.
- Universal Plugins**  
Rollup-superset plugin interface shared between dev and build.
- Fully Typed APIs**  
Flexible programmatic APIs with full TypeScript typing.

Buttons at the bottom include "Get Started", "Why Vite?", "View on GitHub", and "ViteConf 24!"



1. Vite is a modern tool to create React Project.
2. Vite produces Quick and Small bundle size.



React

# Create a React App



## Scaffolding Your First Vite Project

### Compatibility Note

Vite requires [Node.js](#) version 18+ or 20+. However, some templates require a higher Node.js version to work, please upgrade if your package manager warns about it.

NPM   Yarn   PNPM   Bun

\$ npm create vite@latest

bash

1. Vite is a modern tool to create React Project.
2. Vite produces Quick and Small bundle size.



React

# Create a React App



```
Scaffolding project in /Users/prashantjain/workspace/Test Project/react/test2...
```

```
Done. Now run:
```

```
cd test2  
npm install  
npm run dev
```

- prashantjain@Prashants-Mac-mini react % cd test2
- prashantjain@Prashants-Mac-mini test2 % npm install

```
added 215 packages, and audited 216 packages in 26s
```

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

```
found 0 vulnerabilities
```

- ✖ prashantjain@Prashants-Mac-mini test2 % npm run dev

```
> test2@0.0.0 dev  
> vite
```



## Vite + React

count is 0

Edit src/App.jsx and save to test HMR

Click on the Vite and React logos to learn more



# Project Structure

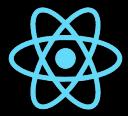
1. `node_modules/` has all the installed node packages
2. `public/` Directory: Contains static files that don't change.
3. `src/` Directory: Main folder for the React code.
  1. `components/`: Reusable parts of the UI, like buttons or headers.
  2. `assets/`: Images, fonts, and other static files.
  3. `styles/`: CSS or stylesheets.
4. `package.json` contains information about this project like name, version, dependencies on other react packages.
5. `vite.config.js` contains vite config.

EXPLORER

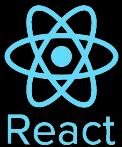
...

LEARNING-REACT

- > node\_modules
- > public
- > src
  - > assets
  - # App.css
  - ⚛ App.jsx
  - # index.css
  - ⚛ main.jsx
- ∅ .eslintrc.cjs
- ∅ .gitignore
- ↳ index.html
- { } package-lock.json
- { } package.json
- ⓘ README.md
- JS vite.config.js

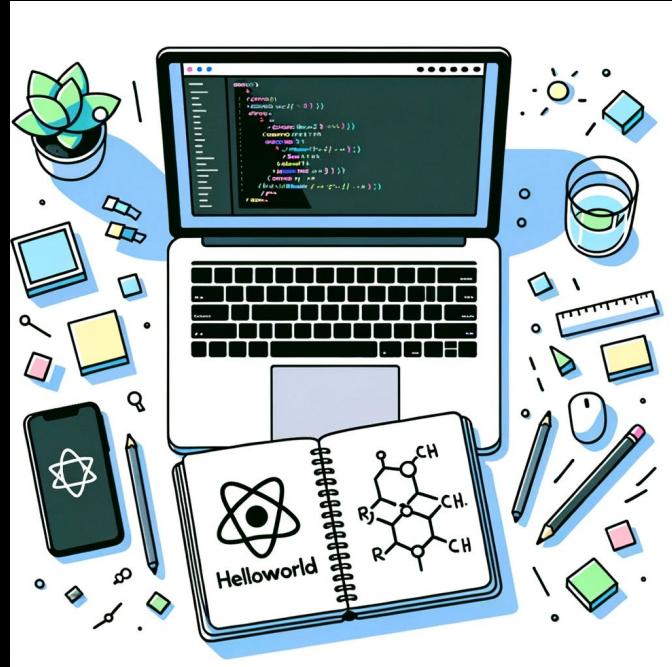


React



# First React Component

- File Extensions
- What is JSX?
- Class vs Function Components
- Exporting component
- Other important Points
- Dynamic Components
- Reusable Components





# File Extensions



## .JS

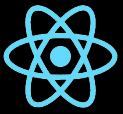
- Stands for **JavaScript**
- Contains **regular JavaScript** code
- Used for **general logic** and components

## .JSX

- Stands for **JavaScript XML**
- Combines **JavaScript** with **HTML-like tags**
- Makes it easier to design **UI components**



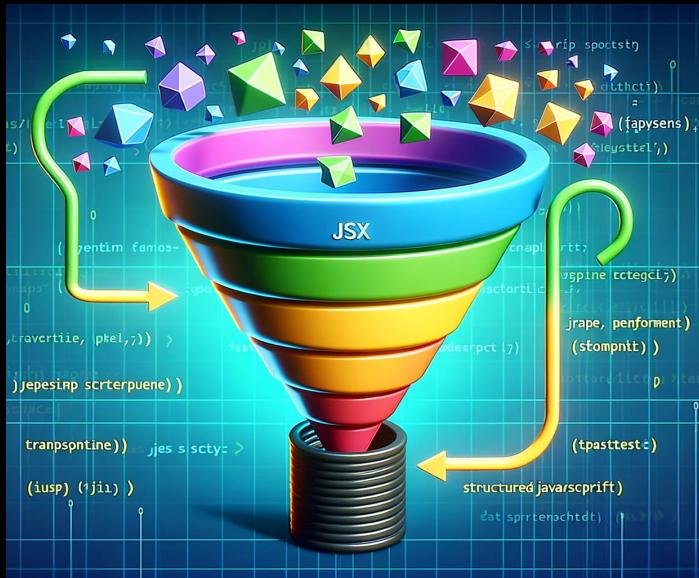
Technically, there **is no difference** between **.js** and **.jsx** files in terms of functionality. The separation is **purely a convention** to differentiate between files that predominantly contain **JSX** from those that are purely **JavaScript**.

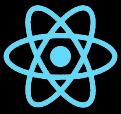


React

## What is JSX?

1. **Definition:** **JSX** determines how the UI will look wherever the component is used.
2. **Not HTML:** Though it **resembles HTML**, you're actually writing **JSX**, which stands for **JavaScript XML**.
3. **Conversion:** **JSX** gets converted to regular **JavaScript**.
4. **Babeljs.io/repl** is a tool that allows you to see how **JSX** is transformed into **JavaScript**.

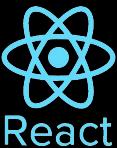




React

# JSX Example

```
function App() {
  return (
    <div className="App">
      <header className="App-header">
        <img src={logo} className="App-logo" alt="logo" />
        <p>
          | Edit <code>src/App.js</code> and save to reload.
        </p>
        <a
          |   className="App-link"
          |   href="https://reactjs.org"
          |   target="_blank"
          |   rel="noopener noreferrer"
          |
          |   Learn React
        </a>
      </header>
    </div>
  );
}
```



# Class vs Function Components

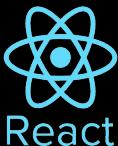
## Class Components

- Stateful: Can manage state.
- Lifecycle: Access to lifecycle methods.
- Verbose: More boilerplate code.
- Not Preferred anymore.

## Functional Components

- Initially **stateless**.
- Can use **Hooks** for state and effects.
- Simpler and more concise.
- More Popular.





React

## Class Components

```
import React, { Component } from 'react';

class Heading extends Component {
  render() {
    return <h1>{this.props.title}</h1>;
  }
}

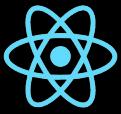
export default Heading;
```

## Function Components

```
import React from 'react';

const Heading = (props) => {
  return <h1>{props.title}</h1>;
};

export default Heading;
```



React

# Using CSS

```
est1 > src > # KGButton.css > ...
.custom-button {
  background-color: #4CAF50;
  color: white;
  padding: 10px 20px;
  border: none;
  border-radius: 5px;
  font-size: 16px;
  cursor: pointer;
  transition: background-color 0.3s ease;
}

.custom-button:hover {
  background-color: #45a049;
}
```

```
import React from 'react';
import './KGButton.css';

const KGButton = () => {
  return (
    <button className="custom-button">
      Click Me!
    </button>
  );
};

export default KGButton;
```