



# iX

**Embrace Opportunity**

# Software Engineering Day 3

Introduction to ReactJS

iX



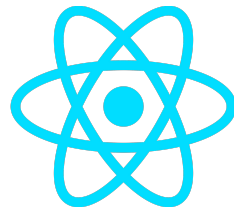
# Today's Overview

- Introduction to React.js, JSX, and Components.
  - Generate React application, extensions for react components and generation.
  - Style components
- 
- Cheat Sheet:
    - <https://byrondev121.github.io/ix-vue-press/>
    - <https://byrondev121.github.io/ix-vue-press/day-3>

# React JS

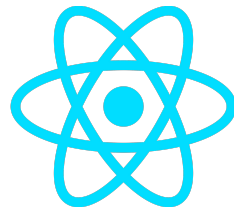
Frontend web framework

# What is react?



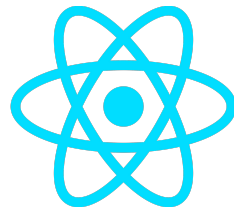
- **React JS is a library for building user interfaces**
- React runs on the client as a SPA(single page application), but can be used to build full stack applications by communicating with a server/API (e.g. MERN stack, next.js, etc.)
- React is often referred to as a front-end “framework” because it is capable and directly compared to a framework such as Angular or Vue.

# Why would you use ReactJS



- Reusable components with own state
- JSX - Dynamic markup language
- Interactive UIs with the virtual DOM
- Performance and testing
- Very popular in industry

# Getting started



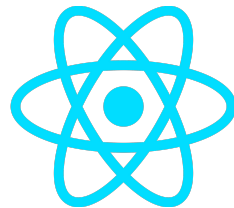
- Node
  - <https://nodejs.org/en/download>
- Create React App
  - <https://github.com/facebook/create-react-app>
  - To create a new react project run the terminal command below (it will create a folder called ix-blog-app that you can then open in vsCode)

```
npx create-react-app [my-app]
```

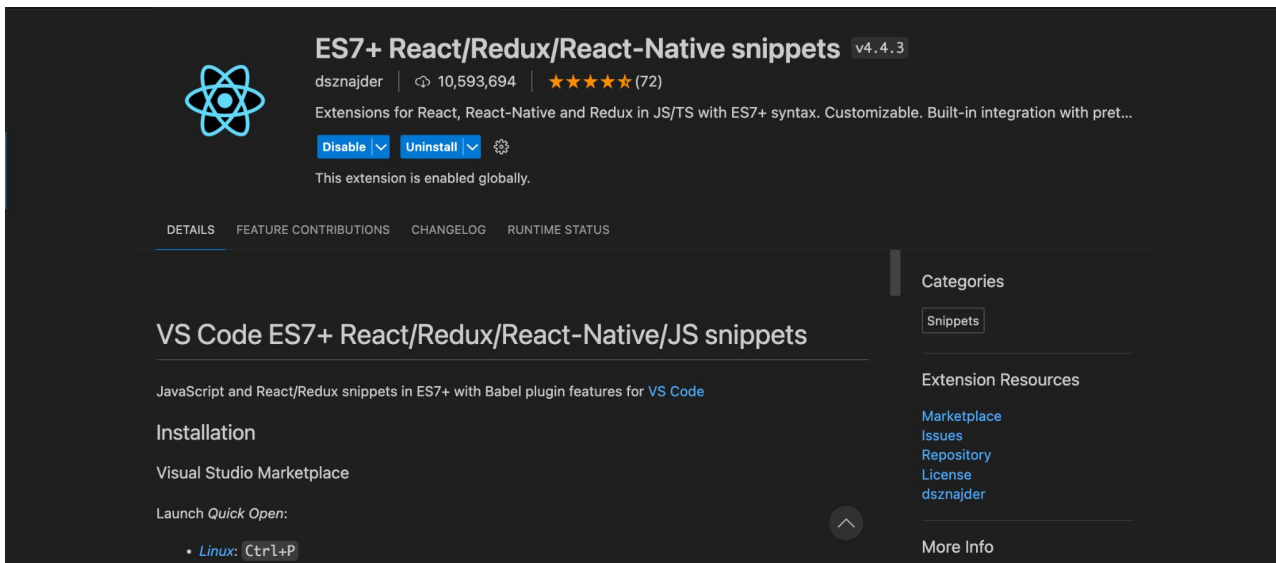
sh

- Install chrome extension for react dev tools
  - [React Developer Tools for Chrome](#)

# Getting started

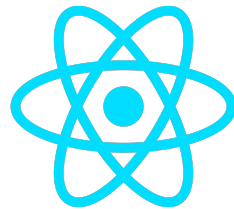


- VSCode Extensions
  - ES7+ React/Redux/React-Native snippets

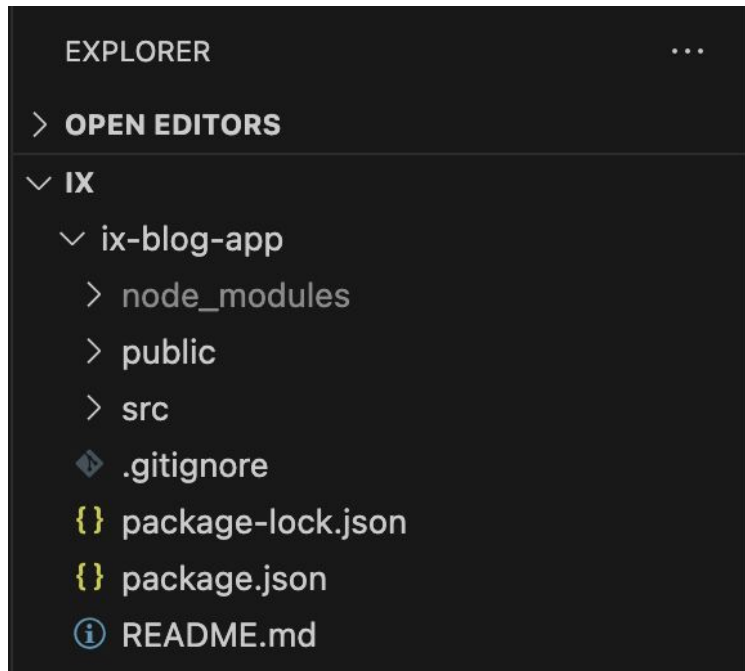
A screenshot of the Visual Studio Code extension marketplace page for the 'ES7+ React/Redux/React-Native snippets' extension by dsnajder. The page has a dark theme. At the top, the extension name is displayed in white, followed by the version 'v4.4.3'. Below this, the author 'dsnajder' is listed, along with a download count of '10,593,694' and a star rating of '5 stars (72)'. A description states: 'Extensions for React, React-Native and Redux in JS/TS with ES7+ syntax. Customizable. Built-in integration with pret...'. There are buttons for 'Disable' and 'Uninstall', and a gear icon for settings. A status message says 'This extension is enabled globally.' Below this is a tab bar with 'DETAILS', 'FEATURE CONTRIBUTIONS', 'CHANGELOG', and 'RUNTIME STATUS'. The 'DETAILS' tab is active. The main content area has the title 'VS Code ES7+ React/Redux/React-Native/JS snippets' and a description: 'JavaScript and React/Redux snippets in ES7+ with Babel plugin features for VS Code'. There is an 'Installation' section with the text 'Visual Studio Marketplace' and 'Launch Quick Open:'. At the bottom, it shows 'Linux: Ctrl+P'. On the right side, there are sections for 'Categories' (with 'Snippets' selected), 'Extension Resources' (with links to Marketplace, Issues, Repository, License, and dsnajder), and 'More Info'.



# Anatomy of React App



- node\_modules
  - This is where all the external libraries that we use are.
  - When you install a new library using npm install (or npx) then it will put it there.
- src
  - This is our code :)



# Anatomy of React App

- public
  - This is the entry-point (index.html).
  - This is the single page that is rendered in our SPA.
  - One element in the body:

```
<div id="root"><div>
```

html

- src/index.js
  - This is the entry-point for react
  - ReactDOM is imported & Instantiates ReactDOM.createRoot() with <div id="root">/><div>
  - Executes render passing in App.js

```
ix-blog-app > public > <> index.html > ...
Byron de Villiers, 7 hours ago | 1 author (Byron de Villiers)
1  <!DOCTYPE html>
2  <html lang="en">
3    <head>
4      <meta charset="utf-8" />
5      <link rel="icon" href="%PUBLIC_URL%/favicon.ico" />
6      <meta name="viewport" content="width=device-width, initial-scale=1" />
7      <meta name="theme-color" content="#000000" />
8      <meta
9        name="description"
10       content="Web site created using create-react-app"
11     />
12     <link rel="apple-touch-icon" href="%PUBLIC_URL%/logo192.png" />
13     <link rel="manifest" href="%PUBLIC_URL%/manifest.json" />
14     <title>React App</title>
15   </head>
16   <body>
17     <noscript>You need to enable JavaScript to run this app.</noscript>
18     <div id="root"></div>
19   </body>
20 </html>
```

```
import React from 'react';
import ReactDOM from 'react-dom/client';
import App from './App';
import './index.css';

const root = ReactDOM.createRoot(document.getElementById('root'));
root.render(
  <React.StrictMode>
    <App />
  </React.StrictMode>
);
```

js

# Anatomy of React App

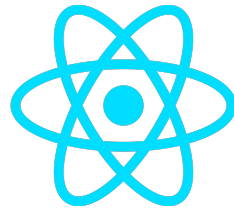
- .gitignore
  - Defines all files to be ignored by git
- package.json
  - Manifest file of any Node.js project and contains the metadata of the project.
  - Defines project name, version, scripts, author, etc.
  - Most importantly defines all dependencies and dev-dependencies for node project.

```
ix-blog-app > {} package.json > ...
Byron de Villiers, 2 hours ago | 1 author (Byron de Villiers)

1  {
2    "name": "ix-blog-app",
3    "version": "0.1.0",
4    "private": true,
5    "dependencies": {
6      "@testing-library/jest-dom": "^5.17.0",
7      "@testing-library/react": "^13.4.0",
8      "@testing-library/user-event": "^13.5.0",
9      "react": "^18.2.0",
10     "react-dom": "^18.2.0",
11     "react-scripts": "5.0.1",
12     "web-vitals": "^2.1.4"
13   },
14   "scripts": {
15     "start": "react-scripts start",
16     "build": "react-scripts build",
17     "test": "react-scripts test",
18     "eject": "react-scripts eject"
19   },
20   "eslintConfig": {
21     "extends": [
22       "react-app",
23       "react-app/jest"
24     ]
25   },
26   "browserslist": {
27     "production": [
28       ">0.2%",
29       "not dead",
30       "not op_mini all"
31     ],
32     "development": [
33       "last 1 chrome version",
34       "last 1 firefox version",
35       "last 1 safari version"
36     ]
37   }
38 }
```

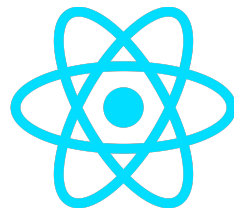
Byron de Villiers, 2 hours ago • Initialize project using Create React App

# Anatomy of React App

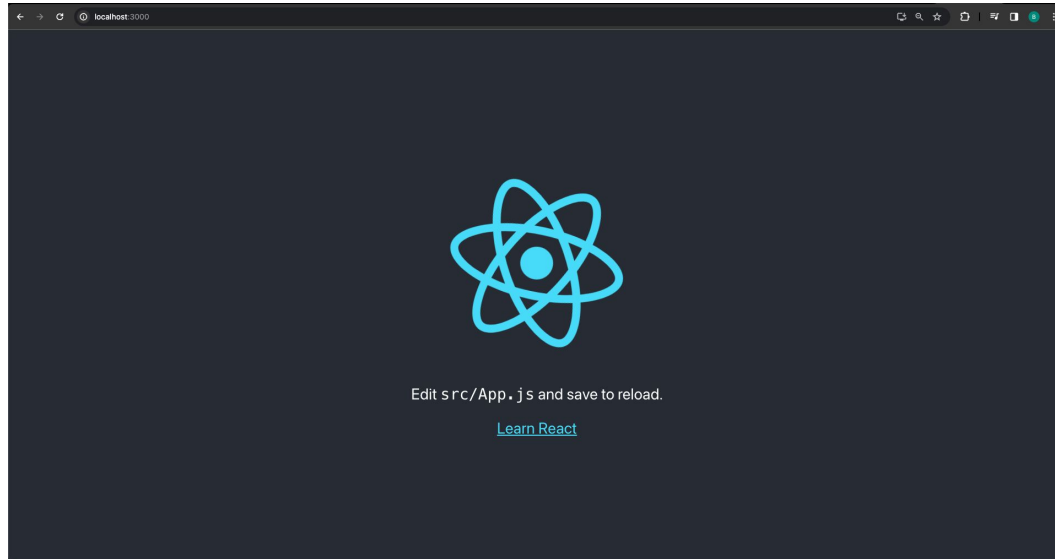


- Package-lock.json
  - package-lock.json is automatically generated for any operations where npm modifies either the node\_modules tree, or package.json. It describes the exact tree that was generated, such that subsequent installs are able to generate identical trees, regardless of intermediate dependency updates.
- README.md
  - A README is a text (markdown) file that introduces and explains a project. It contains information that is commonly required to understand what the project is about.

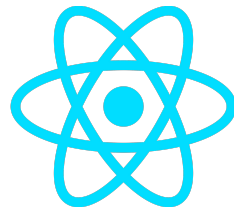
# Run local dev server



```
cd [my-app]  
npm start
```



# Add Bootstrap



- Install bootstrap using npm

```
npm install bootstrap
```

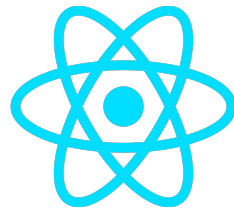
sh

- src/App.js import 'bootstrap/dist/css/bootstrap.min.css';

```
import "bootstrap/dist/css/bootstrap.min.css";
```

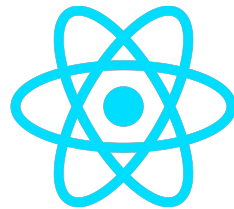
js

# What is JSX (Javascript XML)?



- JSX is a syntax extension for JavaScript that lets you write HTML-like markup inside a JavaScript file. Although there are other ways to write components, most React developers prefer the conciseness of JSX, and most codebases use it.
- JSX allows us to write HTML elements in JavaScript and place them in the DOM without any `createElement()` and/or `appendChild()` methods.
- Learn more [here](#)

# JSX - Syntax



- Defining JSX Elements / components

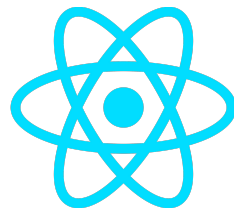
```
function MyButton() {  
  return <button style={{ color: "blue", shadowSize: 2 }}>I'm a button</button>;  
}
```

- Syntax - JSX elements must have self-closing tags if there are no children.

```
export default function MyApp() {  
  return (  
    <div>  
      <h1>Welcome to my app</h1>  
      <MyButton />  
    </div>  
  );  
}
```



# JSX - Syntax



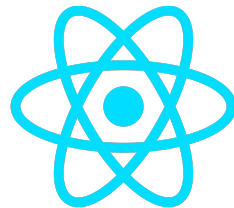
- The HTML code must be wrapped in ONE top level element.

```
const Headings = (  
  <div>  
    <h1>Heading</h1>  
    <h4>Subheading</h4>  
  </div>  
);
```

- Alternatively, you can use a "fragment" to wrap multiple lines. `<></>`

```
const Headings = (  
  <>  
    <h1>Heading</h1>  
    <h4>Subheading</h4>  
  </>  
);
```

# JSX - Under the hood



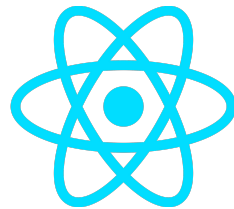
- Fundamentally, JSX just provides syntactic sugar for the `React.createElement(component, props, ...children)` function.
- The JSX code:

```
<MyButton />
```

- Compiles to:

```
React.createElement(button, {style={{color:"blue",shadowSize:2}}}, "I'm a button");
```

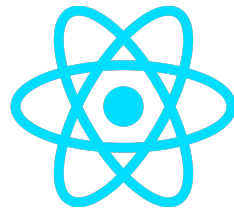
# JSX - Styling



- Inline style syntax. CSS properties are camel case using inline styling

```
const Heading = <h1 style={{ fontSize: "32px;" }}>Hello World</h1>;
```

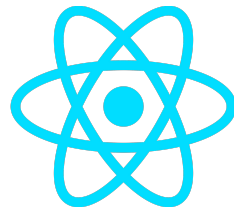
# JSX - Styling



- Attribute class = className. The class attribute is a much used attribute in HTML, but since JSX is rendered as JavaScript, and the class keyword is a reserved word in JavaScript, you are not allowed to use it in JSX. JSX solved this by using className instead. When JSX is rendered, it translates className attributes into class attributes.

```
import "./index.css";  
  
const Heading = <h1 className="heading">Hello World</h1>;
```

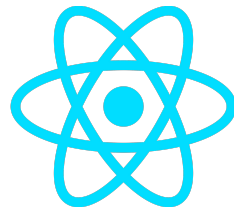
# JSX - Displaying data



- JSX lets you put markup into JavaScript. Curly braces let you “escape back” into JavaScript so that you can embed some variable from your code and display it to the user.

```
export default function MyApp() {  
  const user = {  
    image: ""  
    firstName: "Byron",  
    lastName: "de Villiers",  
    age: 31,  
    email: "byron@mail.com",  
  };  
  return (  
    <div>  
      <h1>Welcome</h1>  
      <h5>{user.firstName}</h5>  
    </div>  
  );  
}
```

# JSX - Conditional rendering



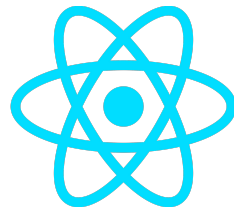
- JSX lets you put markup into JavaScript. Curly braces let you “escape back” into JavaScript so that you can embed some variable from your code and display it to the user.

```
import LoginPage from './pages/Login';
import HomePage from './pages/Home';

export default function MyApp() {
  const user = {
    image: "",
    firstName: "Byron",
    lastName: "de Villiers",
    age: 31,
    email: "byron@mail.com",
    authenticated: false;
  };

  if(user.authenticated){
    return (
      <Home/>
    );
  } else{
    return (
      <Login/>
    );
  }
}
```

# JSX - Conditional rendering



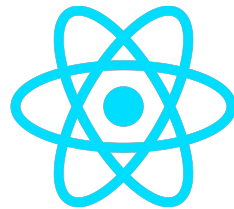
- We can also use a ternary operator.

```
import LoginPage from './pages/Login';
import HomePage from './pages/Home';

export default function MyApp() {
  const user = {
    image: "",
    firstName: "Byron",
    lastName: "de Villiers",
    age: 31,
    email: "byron@mail.com",
    authenticated: false;
  };

  return (
    <div>
      {
        user.isAuthenticated ?
          <HomePage /> :
          <LoginPage />
      }
    </div>
  )
}
```

# JSX - Rendering Lists

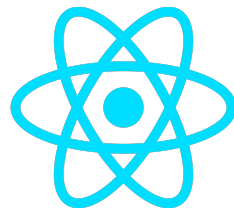


- You will rely on JavaScript features like array `map()` function to render lists of components.

```
export default function MyApp() {  
  const blogs = [  
    { id: 1, title: 'Javascript Fundamentals', },  
    { id: 1, title: 'HTML and CSS', },  
    { id: 1, title: 'iX Boot Camp', },  
  ];  
  
  return (  
    <ul>  
      {  
        blogs.map(x=>{  
          return (  
            <li key={product.id}>  
              {product.title}  
            </li>  
          )  
        })  
      }  
    </ul>  
  )  
}
```



# JSX - Responding to events



- You can respond to events by declaring event handler functions inside your components.
- Notice how `onClick={handleClick}` has no parentheses at the end! Do not call the event handler function: you only need to pass it down. React will call your event handler when the user clicks the button.

```
function MyButton() {  
  function handleClick() {  
    alert("You clicked me!");  
  }  
  
  return <button onClick={handleClick}>Click me</button>;  
}
```

# React UI components

- When using React, think of your UI as a combination of separate UI components

## THE BLOG

### Recent Blog Posts

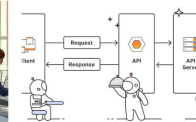


Byron de Villiers • 2024-04-16

#### My Third Blog Post test

Lorem Ipsum is simply dummy text of the printing and typesetting industry. Lorem Ipsum has been the ...

[Version Control](#) [Software Engineering Principles](#) [Data Science](#)



Byron de Villiers • 2024-05-08

#### New test blog

This is a test...

[Artificial Intelligence](#)



Byron de Villiers • 2024-05-08

#### test

test...

[Data Science](#)  
[Software Engineering Principles](#)  
[Project Management](#)

## THE BLOG

### Recent Blog Posts

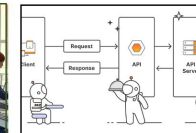


Byron de Villiers • 2024-04-16

#### My Third Blog Post test

Lorem Ipsum is simply dummy text of the printing and typesetting industry. Lorem Ipsum has been the ...

[Version Control](#) [Software Engineering Principles](#) [Data Science](#)



Byron de Villiers • 2024-05-08

#### New test blog

This is a test...

[Artificial Intelligence](#)



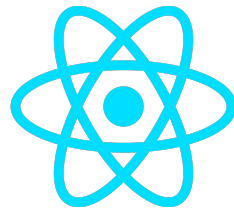
Byron de Villiers • 2024-05-08

#### test

test...

[Data Science](#)  
[Software Engineering Principles](#)  
[Project Management](#)

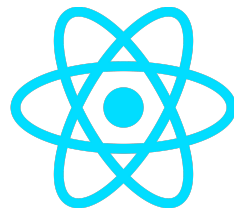
# Components: Functions vs. Classes



- A functional component is just a plain JavaScript pure function that accepts props as an argument and returns a React element(JSX).
- This is the most common way of creating components in React.

```
export const Header = () => {  
  return (  
    <div>  
      <h1>Hello World</h1>  
    </div>  
  )  
};
```

# Components: Functions vs. Classes



- A class component requires you to extend from `React.Component` and create a `render` function that returns a React element.
- Class components are less commonly used than pure components.

```
export default class Header extends React.Component {  
  render(){  
    return (  
      <div>  
        <h1>Hello World</h1>  
      </div>  
    )  
  }  
}
```

# Let's Build the Expected Home Page

iX Software Engineering Blog

Home Categories Blogs About 

## THE BLOG

### Recent Blog Posts



Dev Test • 2024-04-22

#### My First Blog Post

Lorem Ipsum is simply dummy text of the printing and typesetting industry. Lorem Ipsum has been the ...

Web Development Software Engineering



Dev Test • 2024-04-22

#### My Second Blog Post

Lorem Ipsum is simply dummy text of the printing and typesetting industry. Lorem Ipsum has been the ...

Web Development Software Engineering



Dev Test • 2024-04-22

#### My Third Blog Post

Lorem Ipsum is simply dummy text of the printing and typesetting industry. Lorem Ipsum has been the ...

Web Development Software Engineering

### Categories

#### Project Management

orem Ipsum is simply dummy text of the printing and typesetting industry. Lorem Ipsum has been the ...

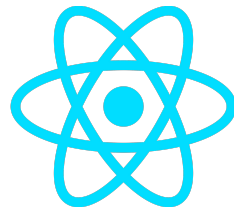
#### Web Development

orem Ipsum is simply dummy text of the printing and typesetting industry. Lorem Ipsum has been the ...

#### Software Engineering

orem Ipsum is simply dummy text of the printing and typesetting industry. Lorem Ipsum has been the ...

# Setup Components

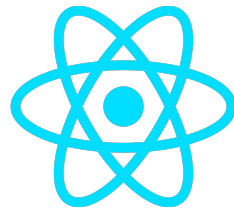


- To keep our code base clean we can move the HomePage component to its own file.
- In *frontend/src/* create a *components* directory in the components dir create a dir for the NavBar Component and add a file called *index.js*.

```

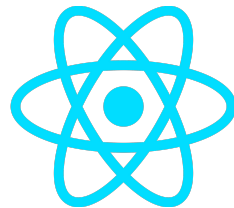
  frontend
  ├── build
  ├── node_modules
  ├── public
  └── src
      ├── components
      │   └── NavBar
      │       └── index.js
      ├── App.css
      ├── App.jsx
      ├── App.test.js
      ├── dummy-data.json
      ├── index.css
      ├── index.jsx
      ├── logo.svg
      ├── reportWebVitals.js
      ├── setupTests.js
      ├── package-lock.json
      ├── package.json
      └── README.md
```

# Navbar



- Open `frontend/src/components/Navbar/index.js`
- Generate react functional component
  - `rfc` 'tab'
- See code blocks in [vue-press](#)

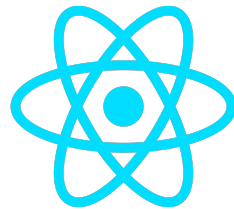
# Heading



- Create a file called index.js under src/components/Heading
  - E.g. src/components/Heading/index.js
- Generate react functional component
  - `rfc` 'tab'
- See code blocks in [vue-press](#)

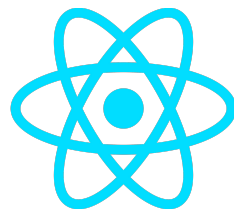


# SubHeading



- Create a file called index.js under src/components/SubHeading
  - E.g. src/components/SubHeading/index.js
- Generate react functional component
  - `rfc` 'tab'
- See code blocks in [vue-press](#)

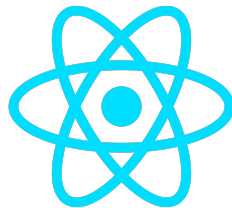
# Blog Grid



- Create a file called index.js under src/components/BlogGrid
  - E.g. src/components/BlogGrid/index.js
- Generate react functional component
  - `rfc` 'tab'
- See code blocks in [vue-press](https://vuepress.vuejs.org/)

**Note:** For now we will importing dummy data for the component, later in the course we'll learn about an implement CRUD:

```
// Importing dummy data
const data = require("../dummy-data.json");
let blogPosts = data.blogPosts;
const categories = data.categories;
```

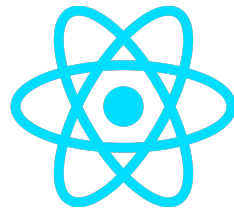


# Blog Item

- Create a file called index.js under src/components/BlogItem
  - E.g. src/components/BlogItem/index.js
- Generate react functional component
  - `rfc` 'tab'
- See code blocks in [vue-press](#)

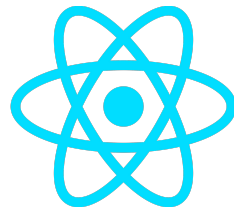
**Note:** Data is being passed into this component from the parent component. These data is called "props" (properties). We will learn more about properties tomorrow.

# Blog Item Text



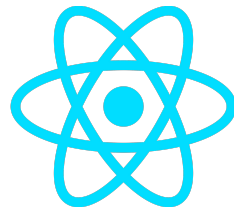
- Create a file called index.js under src/components/BlogItemText
  - E.g. src/components/BlogItemText/index.js
- Generate react functional component
  - `rfc` 'tab'
- See code blocks in [vue-press](#)

# Categories



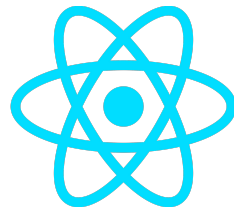
- Create a file called index.js under src/components/Categories
  - E.g. src/components/Categories/index.js
- Generate react functional component
  - `rfc` 'tab'
- See code blocks in [vue-press](#)

# Category List



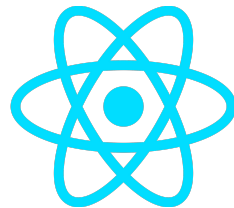
- Create a file called index.js under src/components/CategoryList
  - E.g. src/components/CategoryList/index.js
- Generate react functional component
  - `rfc` 'tab'
- See code blocks in [vue-press](#)

# Category List



- Create a file called index.js under src/components/CategoryList
  - E.g. src/components/CategoryList/index.js
- Generate react functional component
  - `rfc` 'tab'
- See code blocks in [vue-press](#)

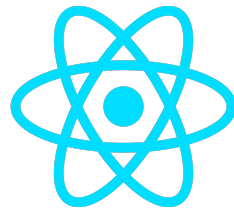
# HomePage



- Create a file called index.js under src/components/Home
  - E.g. src/components/Home/index.js
- Generate react functional component
  - `rfc` 'tab'
- See code blocks in [vue-press](#)



# Update App JS



- In `src/App.jsx`
- Update the App component to return the new HomePage component

```
import HomePage from "../components/HomePage";

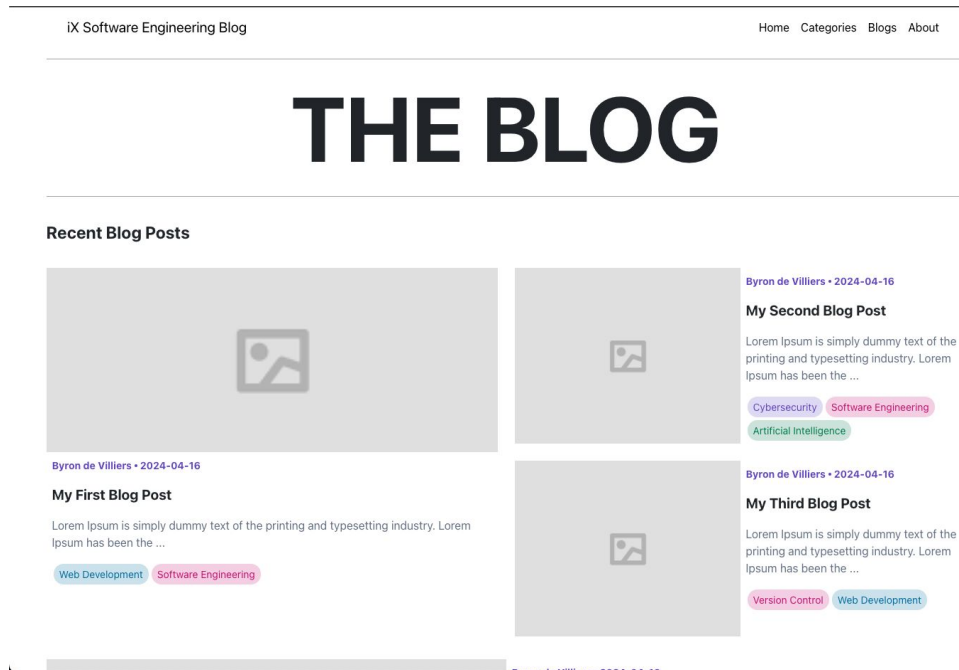
function App() {
  return (
    <div className="App">
      <HomePage />
    </div>
  );
}
```

# Homework

Apply what we have learned

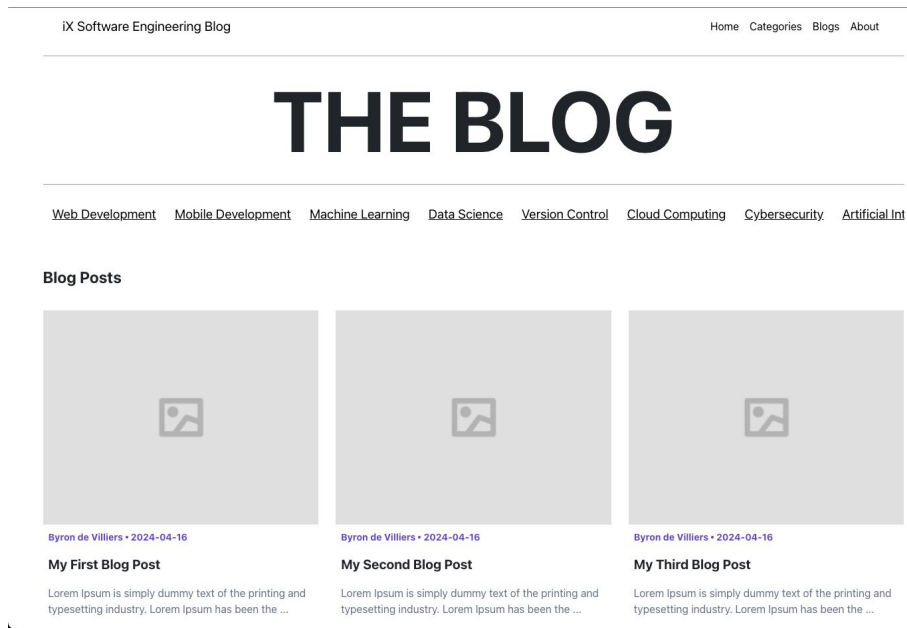
# Homework

- Complete the home page component and subcomponents.



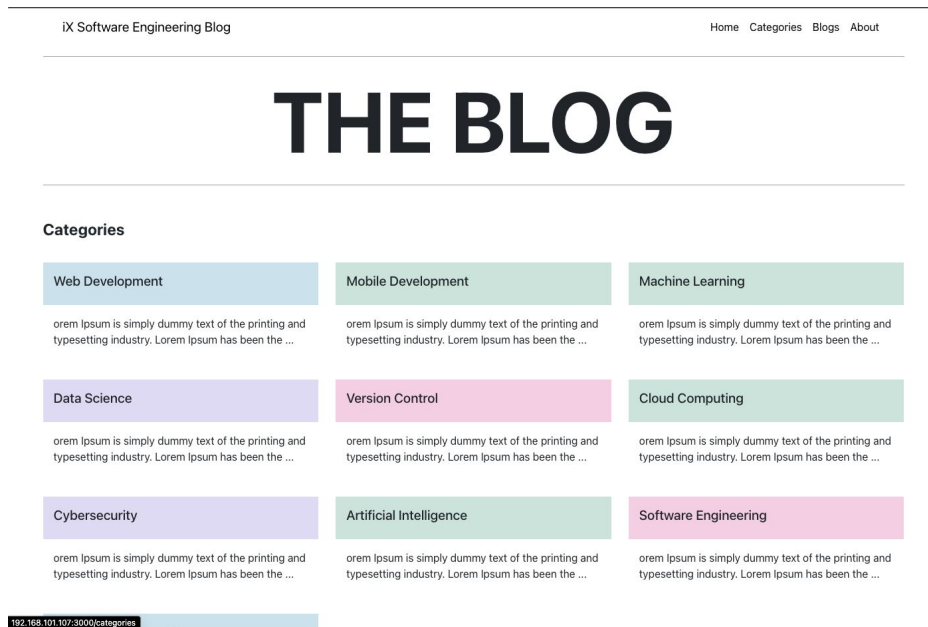
# Homework

- Create and complete the blog page component and subcomponents.



# Homework

- Create and complete the category page component and subcomponents.



# Homework

- Each page component can be displayed by adding the component under the return function in src/App.js

```
import HomePage from "../components/HomePage";
import BlogsPage from "../components/BlogsPage";
import CategoriesPage from "../components/CategoriesPage";

function App() {
  return (
    <div className="App">
      <HomePage />
      <BlogsPage />
      <CategoriesPage />
    </div>
  );
}
```

# Next Class

Understanding State Management  
and Props in React

