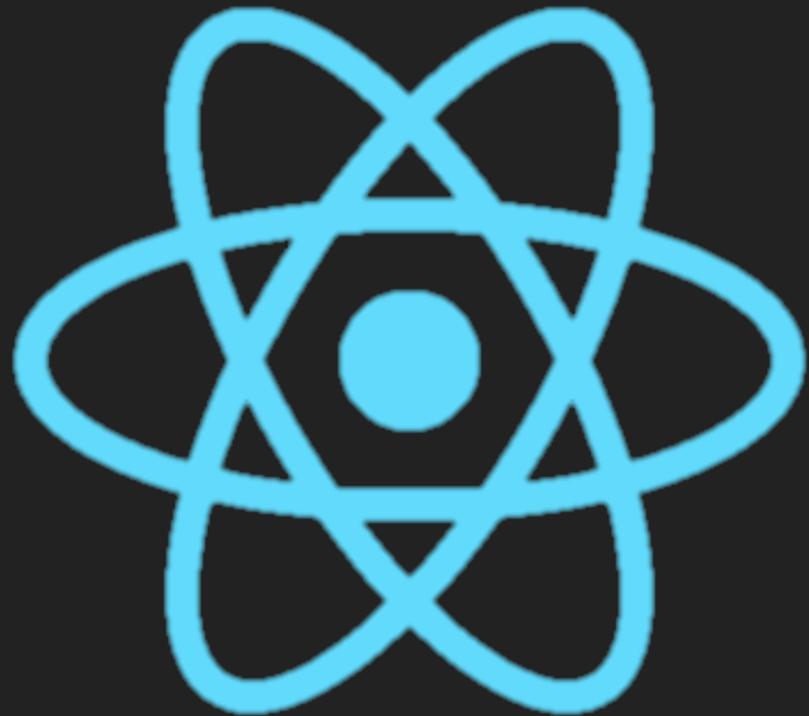


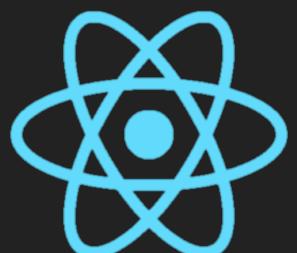
Introduction to Reactjs

By Joey Chua



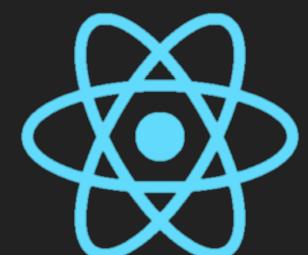
About Me

- Joey Chua
- Data Scientist @ Vertex
- <https://www.linkedin.com/in/joeychuasm/>
- Course Materials @
<https://github.com/joeychuavertex/techladies-react-intro>

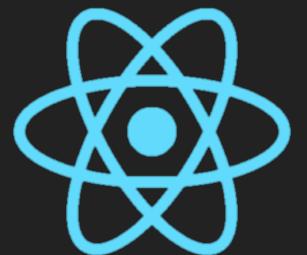


Overview of React

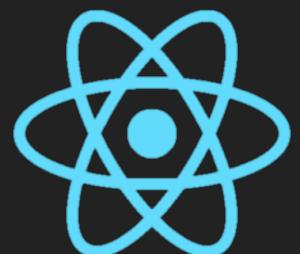
- What is React
- Why React
- Using React
- Create a simple app



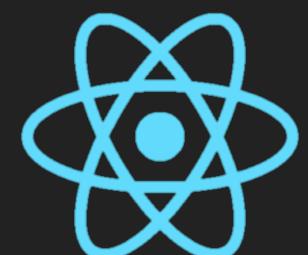
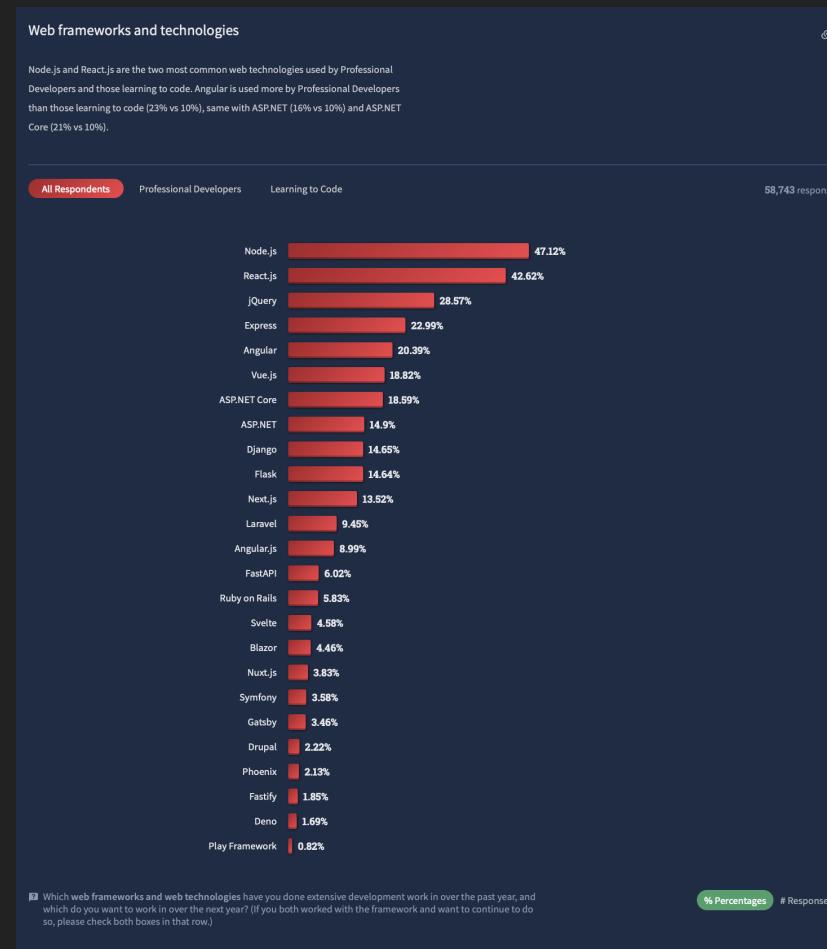
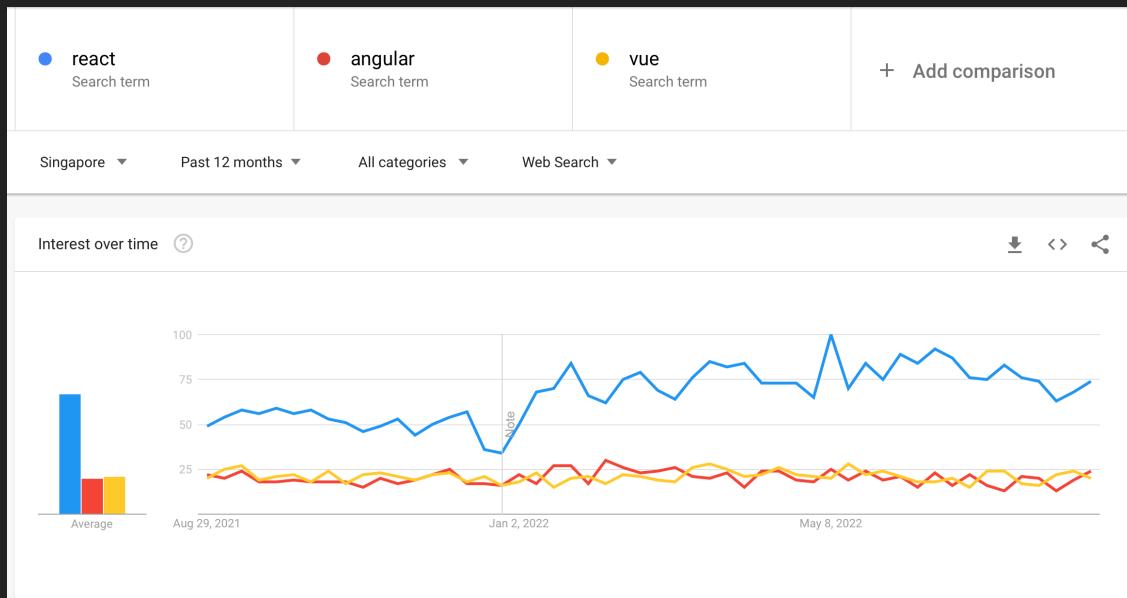
Why React?



**It is easy to understand, has a
large community, and is one of the
most popular library**

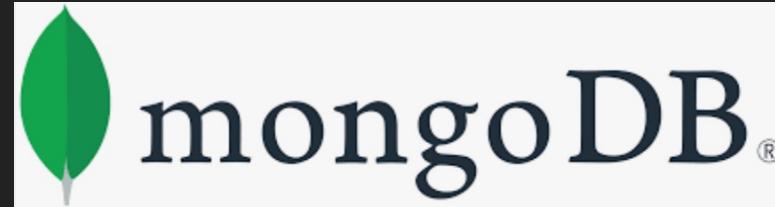


Trends



Stack

M



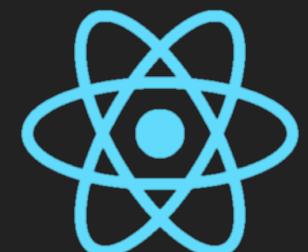
E



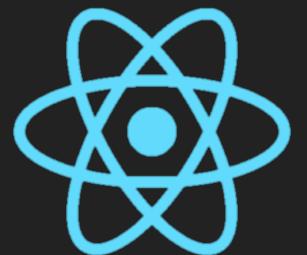
R



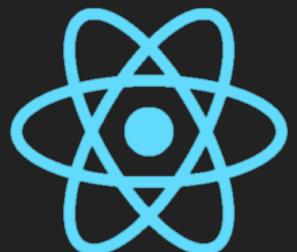
N



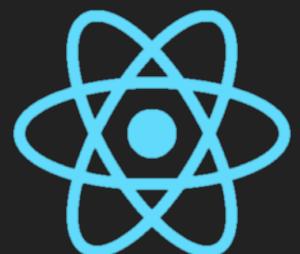
What is React?



**React is a JavaScript library
for building user interfaces**

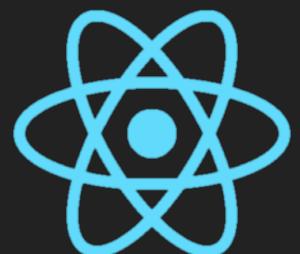


**React is developed by
Facebook**



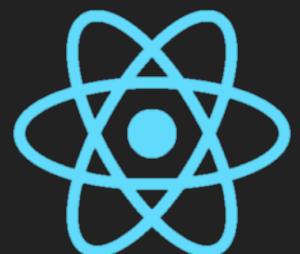
Declarative

Each line of code declares what each element of the app is.



Component-based

Build reusable UI elements and Javascript functions that accept inputs



Header Component

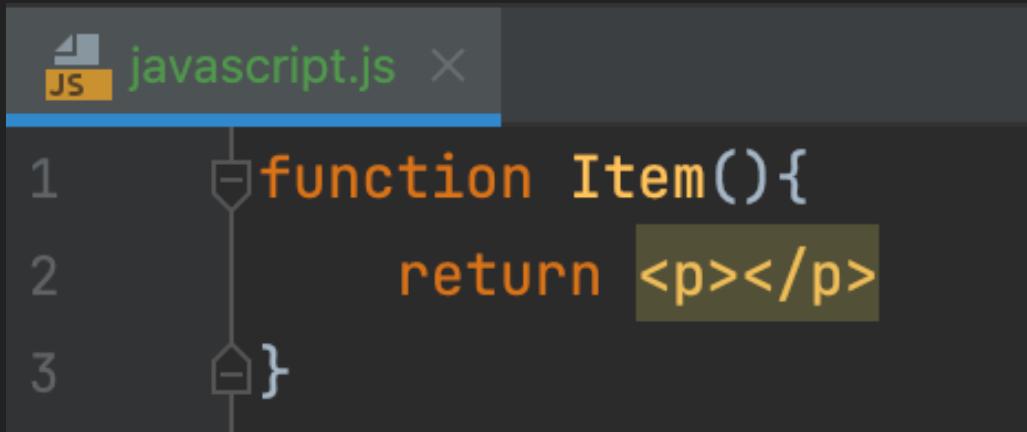
Content Component

Sidebar
Component

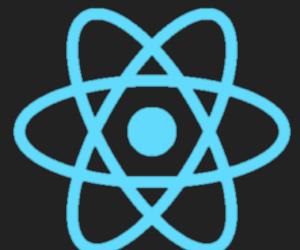
Content Component

Content Component

JSX is used to describe how the **interface** should look like

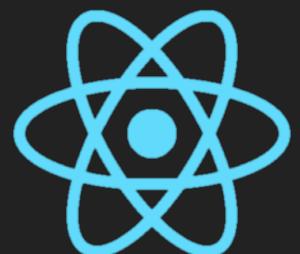


```
1  function Item(){  
2      return <p></p>  
3  }
```

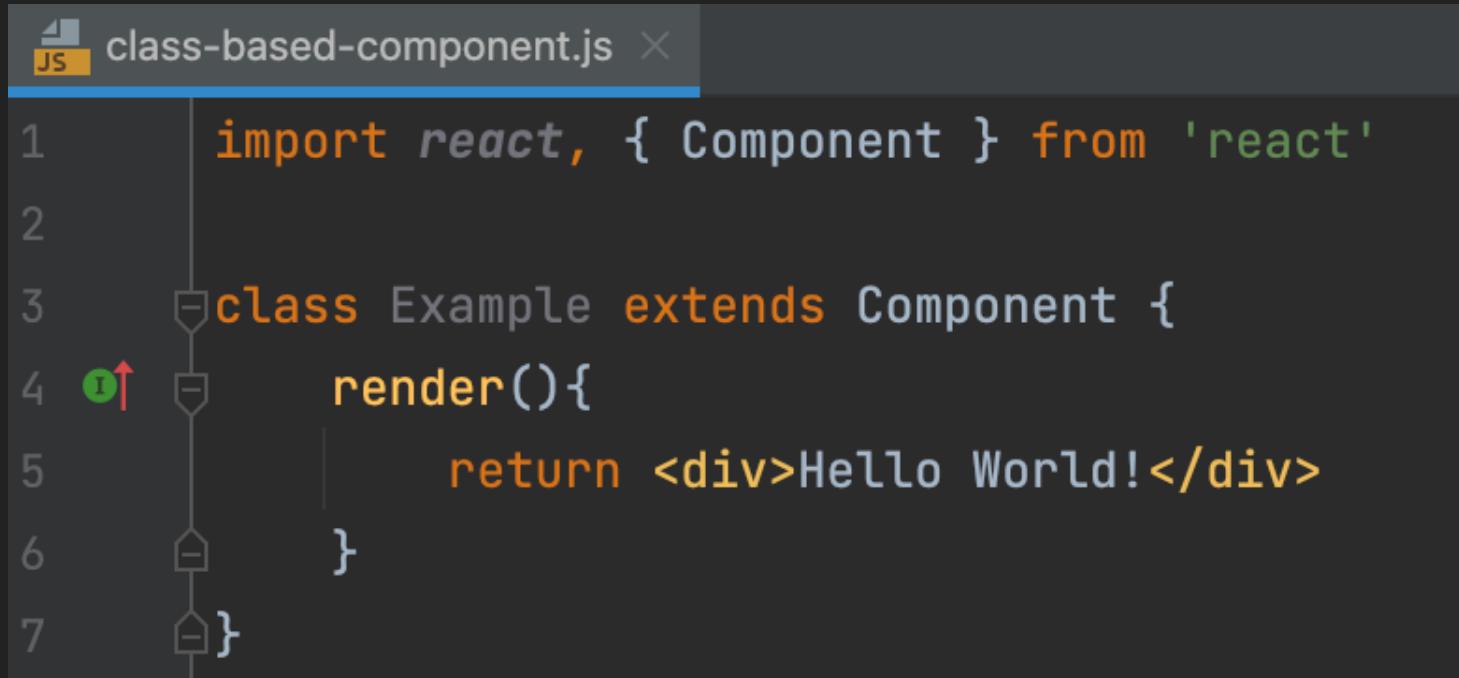


Two Types of Components

1. Class-based
2. Functional



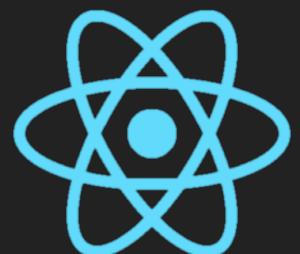
Class-based



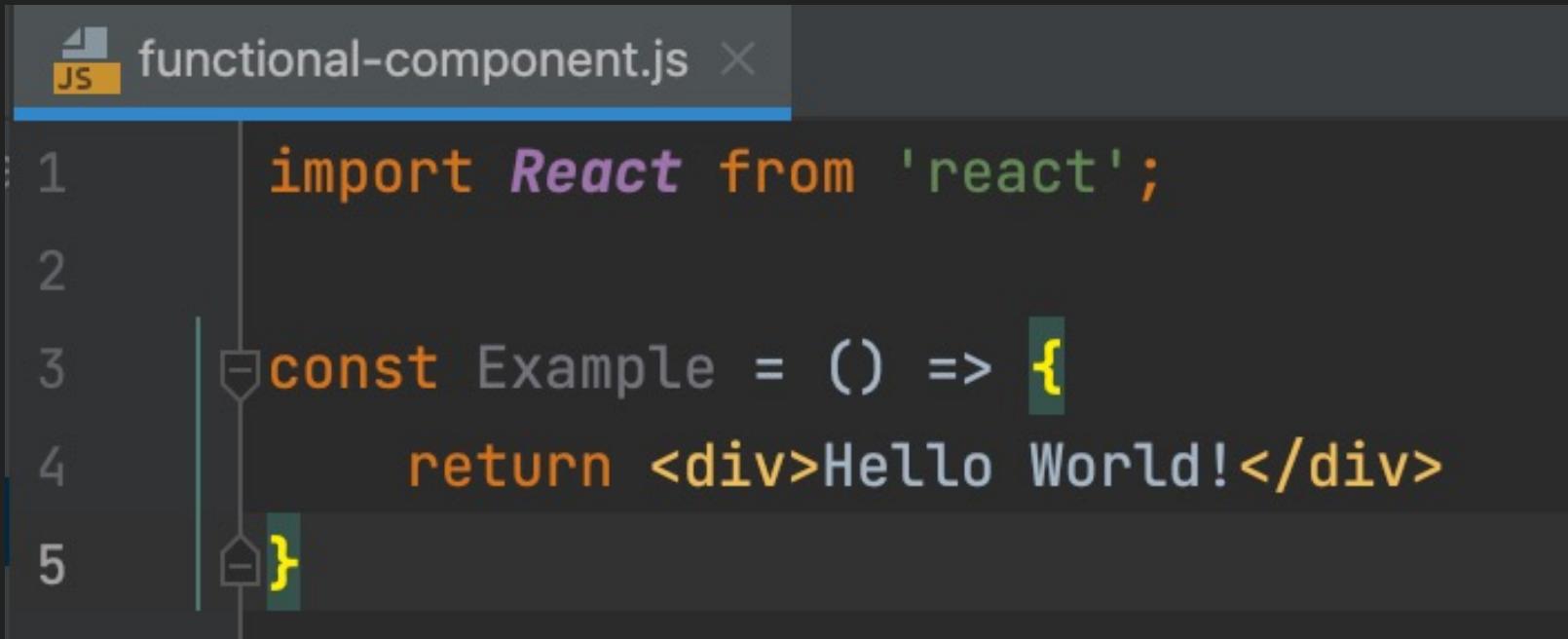
A screenshot of a code editor window titled "class-based-component.js". The code is a simple class component:

```
1 import react, { Component } from 'react'
2
3 class Example extends Component {
4     render(){
5         return <div>Hello World!</div>
6     }
7 }
```

The code editor uses syntax highlighting and includes a vertical brace matching indicator on the left margin, showing the correspondence between opening and closing braces.



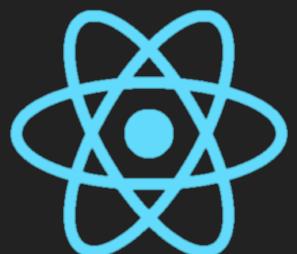
Functional



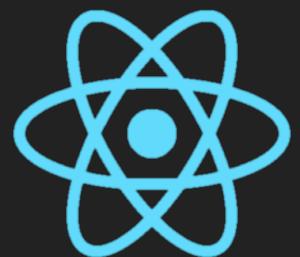
A screenshot of a code editor showing a file named "functional-component.js". The code contains a single functional component definition:

```
1 import React from 'react';
2
3 const Example = () => {
4     return <div>Hello World!</div>
5 }
```

The code editor has a dark theme with syntax highlighting. The file tab shows "functional-component.js" with a JS icon. The code is numbered 1 through 5. The opening brace of the function and the closing brace are highlighted in green.



How to use React?

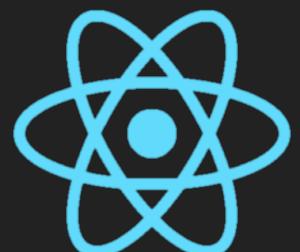


React Environment Setup

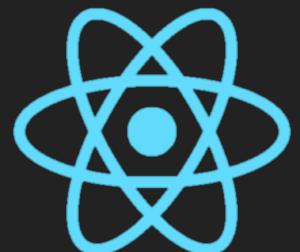
1. Manually set up using webpack & babel
2. Use **create-react-app** command

Terminal: Local × + ▾

```
joeychua@JoeyChuas-MacBook-Pro my-app % npx create-react-app my-app
```



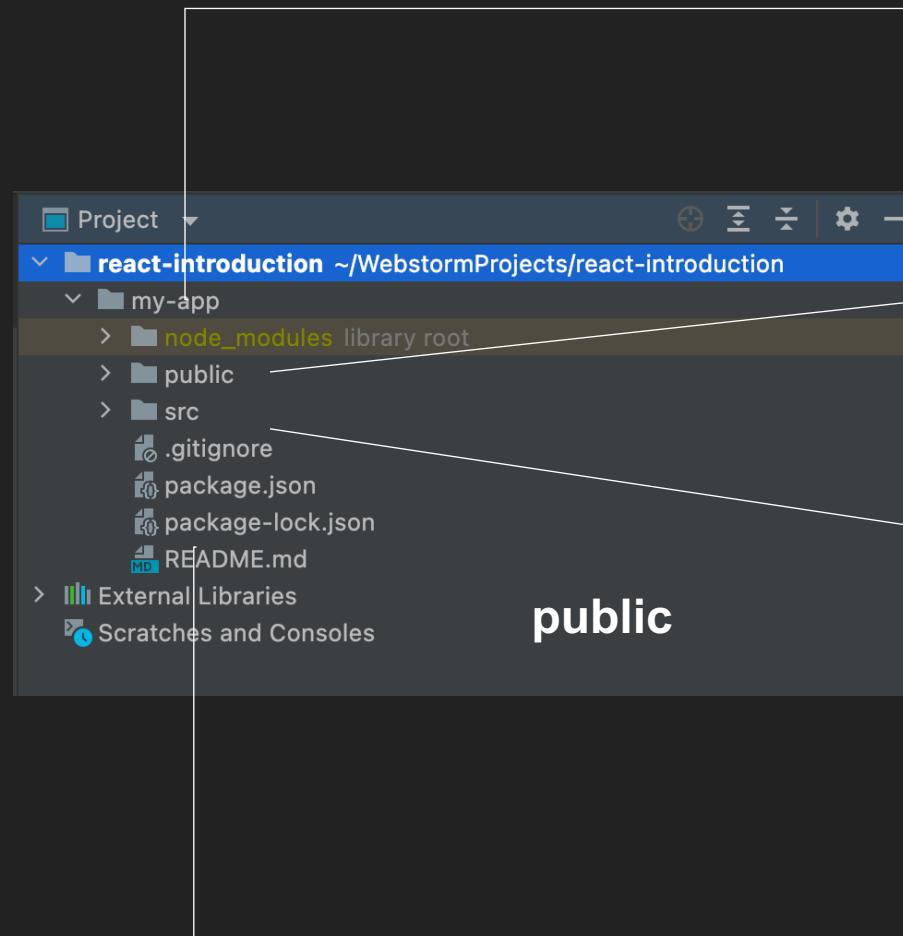
Create-react-app is a simple
tool or command that
generates required **files** &
folders to start the application



**Node is a Javascript runtime
that allows us to execute JS
code on the server**



Files and Folders



→ **node_modules:** Source code for dependencies. Do not touch.

→ **public:** The directory where your static files are stored

→ **src: index.js, app.js etc.**

→ **package.json:** The main file that defines the dependencies and other settings for the project

Index.html

React app is mounted here

```
index.html ×
23      Unlike "/favicon.ico" or "favicon.ico", "%PUBLIC_URL%/favicon.ico" will
24      work correctly both with client-side routing and a non-root public URL.
25      Learn how to configure a non-root public URL by running `npm run build`.
26      -->
27      <title>React App</title>
28      </head>
29      <body>
30          <noscript>You need to enable JavaScript to run this app.</noscript>
31          <div id="root"></div>
32          <!--
33              This HTML file is a template.
34              If you open it directly in the browser, you will see an empty page.
35
36              You can add webfonts, meta tags, or analytics to this file.
37              The build step will place the bundled scripts into the <body> tag.
38
39              To begin the development, run `npm start` or `yarn start`.
40              To create a production bundle, use `npm run build` or `yarn build`.
41          -->
42          </body>
43      </html>
```

Root

Index.js

We have **ReactDOM** to render the application with **ID root**

```
JS index.js ×  
1 import React from 'react';  
2 import ReactDOM from 'react-dom/client';  
3 import './index.css';  
4 import App from './App';  
5 import reportWebVitals from './reportWebVitals';  
6  
7 const root = ReactDOM.createRoot(document.getElementById('root'));  
8 root.render(  
9   <React.StrictMode>  
10     <App />  
11   </React.StrictMode>  
12 );  
13  
14 // If you want to start measuring performance in your app, pass a function  
15 // to log results (for example: reportWebVitals(console.log))  
16 // or send to an analytics endpoint. Learn more: https://bit.ly/CRA-vitals  
17 reportWebVitals();
```

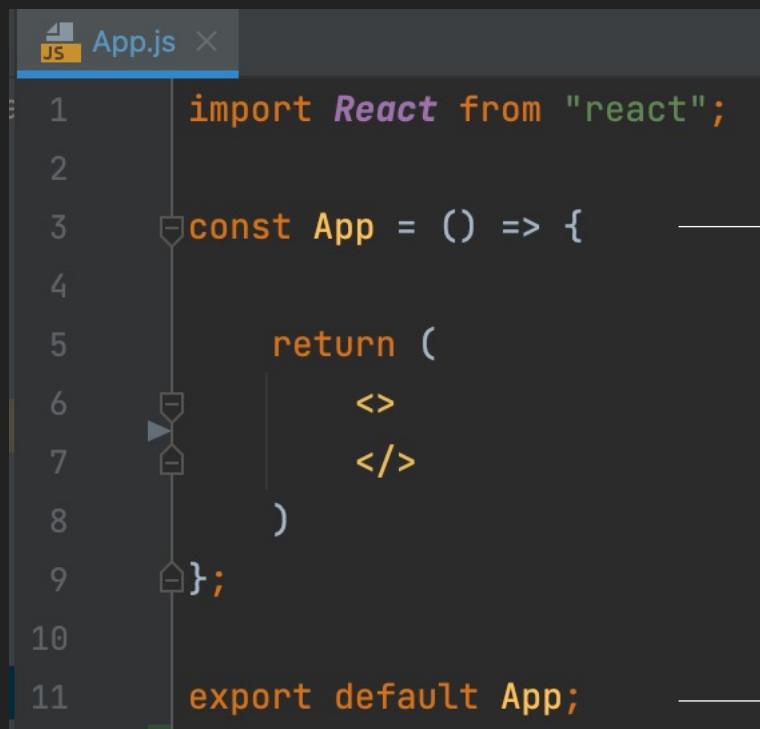
Index.css

HTML

React App

App.js

In HTML, we write class but in JSX we will write **className**



```
1 import React from "react";
2
3 const App = () => {
4
5     return (
6         <>
7             </>
8         )
9     }
10
11 export default App;
```

Function

Note! You only export 1 component from each file

Export

Terminal

Change Directory

```
Terminal: Local (2) × + ▾  
joeychua@JoeyChuas-MacBook-Pro techladies-react-intro % cd my-app  
joeychua@JoeyChuas-MacBook-Pro my-app %
```

Start Server

```
joeychua@JoeyChuas-MacBook-Pro my-app % npm start  
  
> my-app@0.1.0 start  
> react-scripts start
```

Terminal

Terminal: Local × + ▾

Compiled successfully!

You can now view `my-app` in the browser.

Local: <http://localhost:3000>

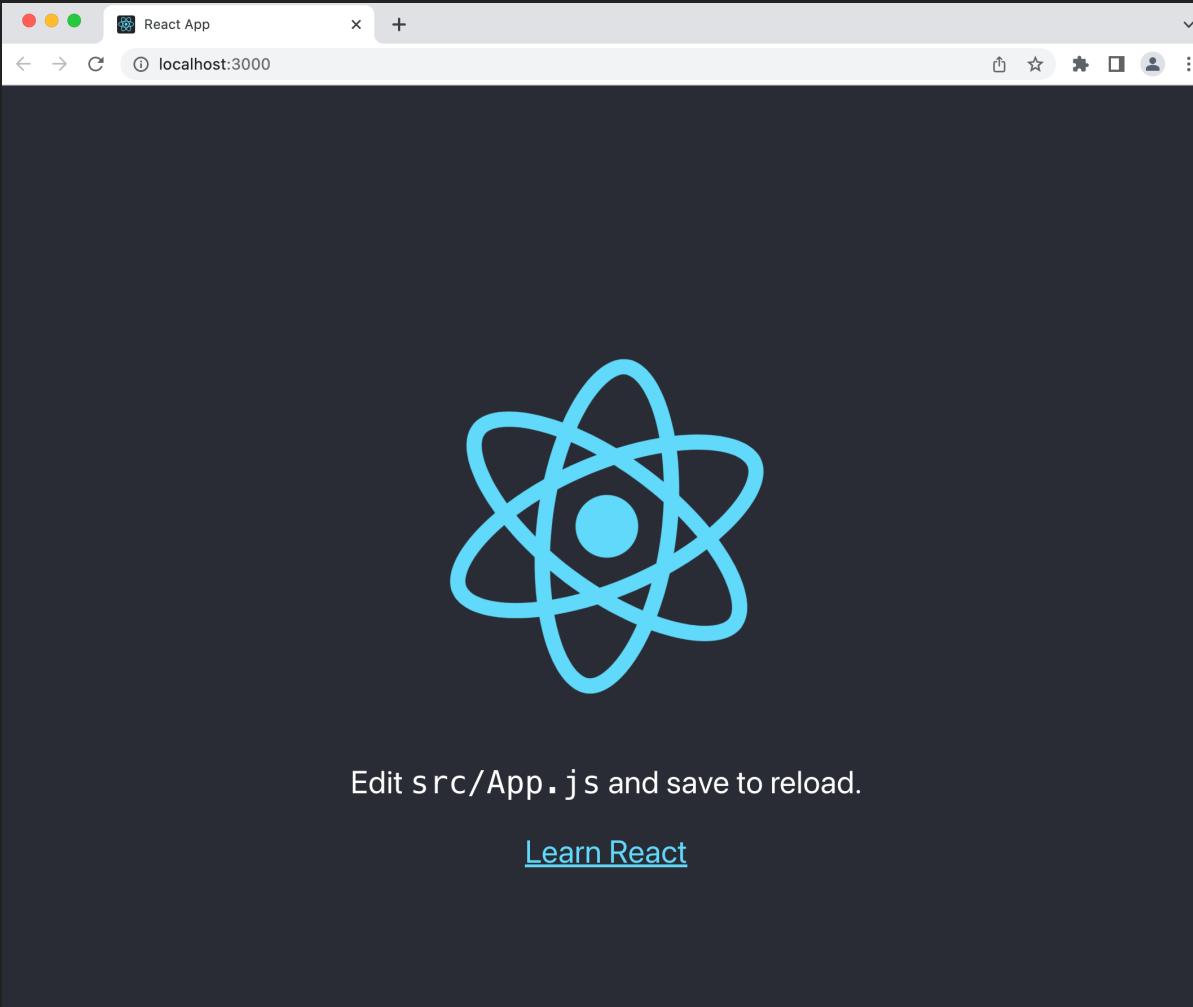
On Your Network: <http://172.20.10.7:3000>

Note that the development build is not optimized.

To create a production build, use `npm run build`.

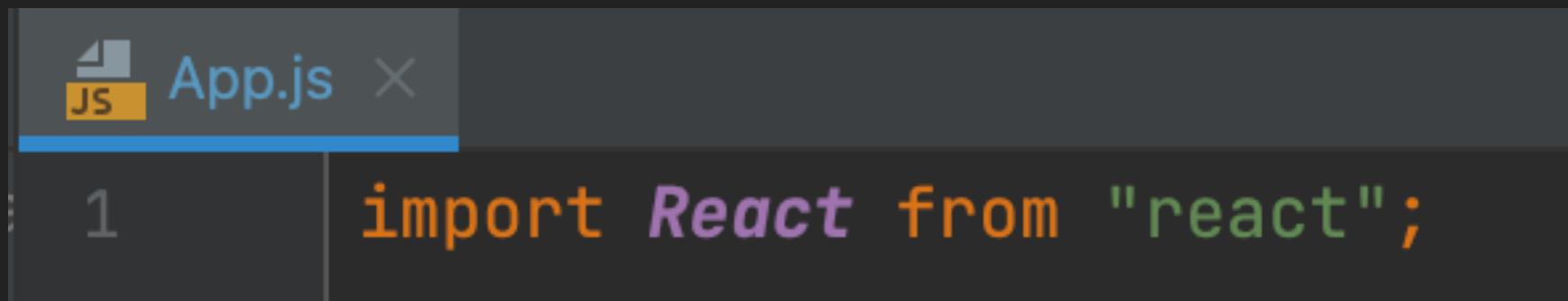
webpack compiled **successfully**

LocalHost



App.js

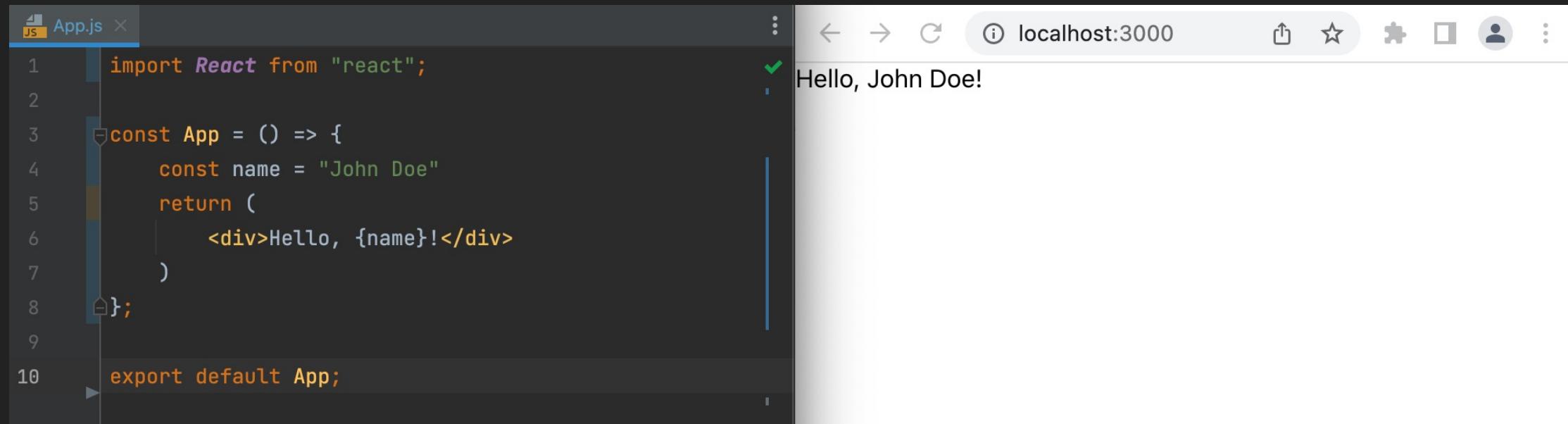
To use the library, we need to **import** React

A screenshot of a code editor showing a single file named "App.js". The file contains one line of code: "import React from "react";". The code editor has a dark theme with syntax highlighting. The word "React" is highlighted in purple, and the file name "App.js" is highlighted in blue in the tab bar.

```
1 import React from "react";
```

App.js

Delete function and change to **const**



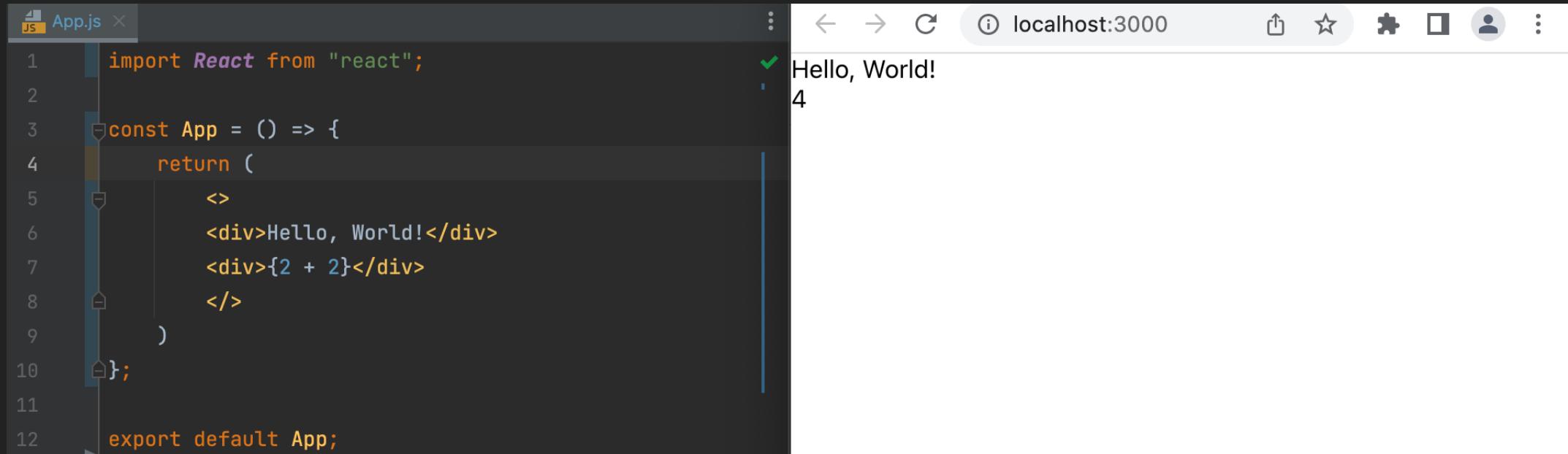
The image shows a split-screen development environment. On the left, a code editor displays the file `App.js` with the following content:

```
1 import React from "react";
2
3 const App = () => {
4     const name = "John Doe"
5     return (
6         <div>Hello, {name}!</div>
7     )
8 }
9
10 export default App;
```

On the right, a web browser window is open at `localhost:3000`, showing the output of the code: `Hello, John Doe!`.

App.js

Dynamic



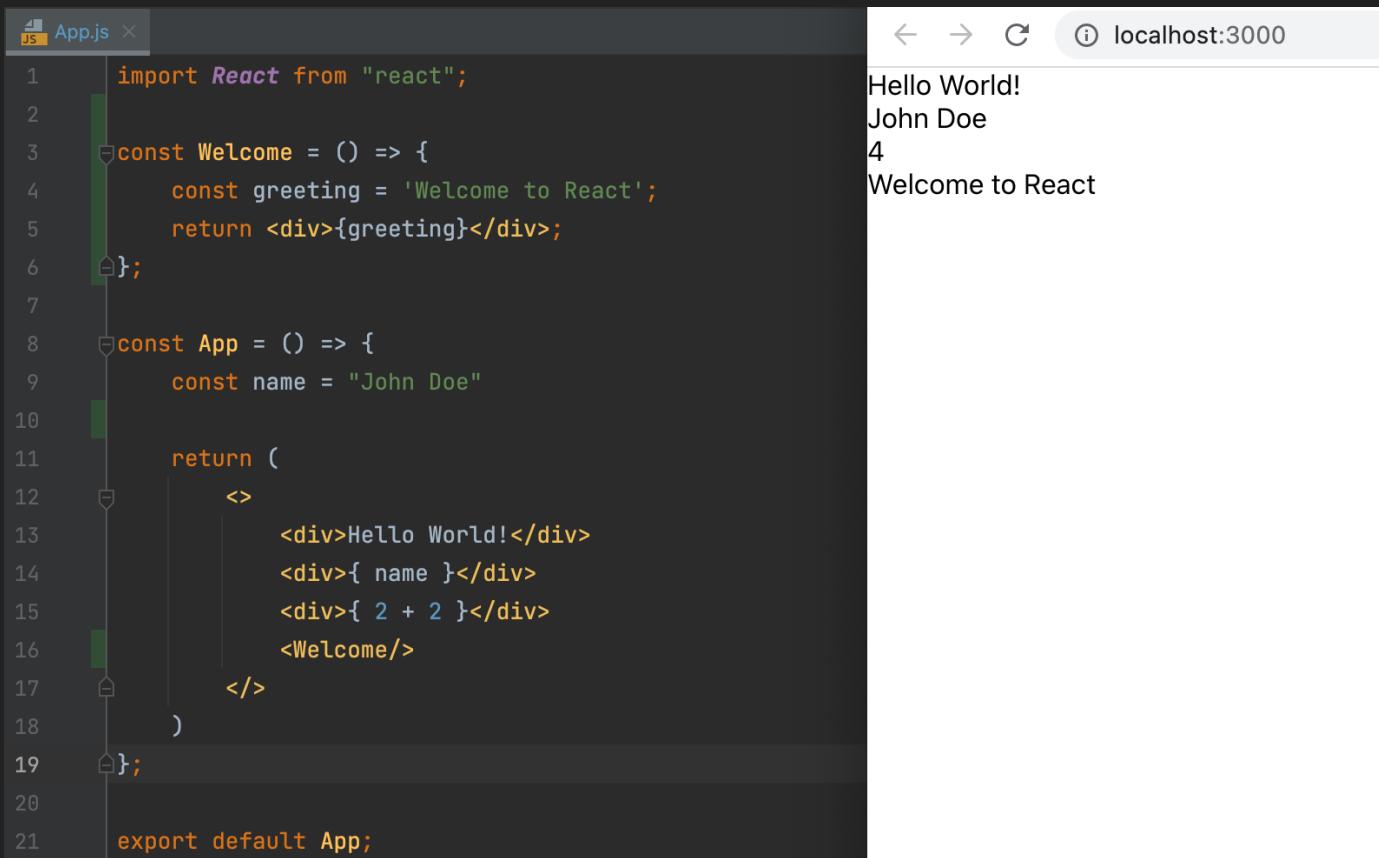
The image shows a split-screen development environment. On the left, a code editor displays the file `App.js` with the following content:

```
1 import React from "react";
2
3 const App = () => {
4     return (
5         <>
6             <div>Hello, World!</div>
7             <div>{2 + 2}</div>
8         </>
9     )
10}
11
12 export default App;
```

On the right, a browser window shows the rendered output at `localhost:3000`. The page displays the text "Hello, World!" and the number "4".

App.js

Props Destructuring



The image shows a code editor on the left and a browser window on the right. The code editor displays the file `App.js` with the following content:

```
1 import React from "react";
2
3 const Welcome = () => {
4     const greeting = 'Welcome to React';
5     return <div>{greeting}</div>;
6 }
7
8 const App = () => {
9     const name = "John Doe"
10
11     return (
12         <>
13             <div>Hello World!</div>
14             <div>{ name }</div>
15             <div>{ 2 + 2 }</div>
16             <Welcome/>
17         </>
18     )
19 }
20
21 export default App;
```

The browser window on the right shows the rendered output at `localhost:3000`:

Hello World!
John Doe
4
Welcome to React

App.js

List and Keys

The image shows a development environment with two main panes. On the left is a code editor for `App.js`, and on the right is a browser window displaying a list of employees.

Code Editor (App.js):

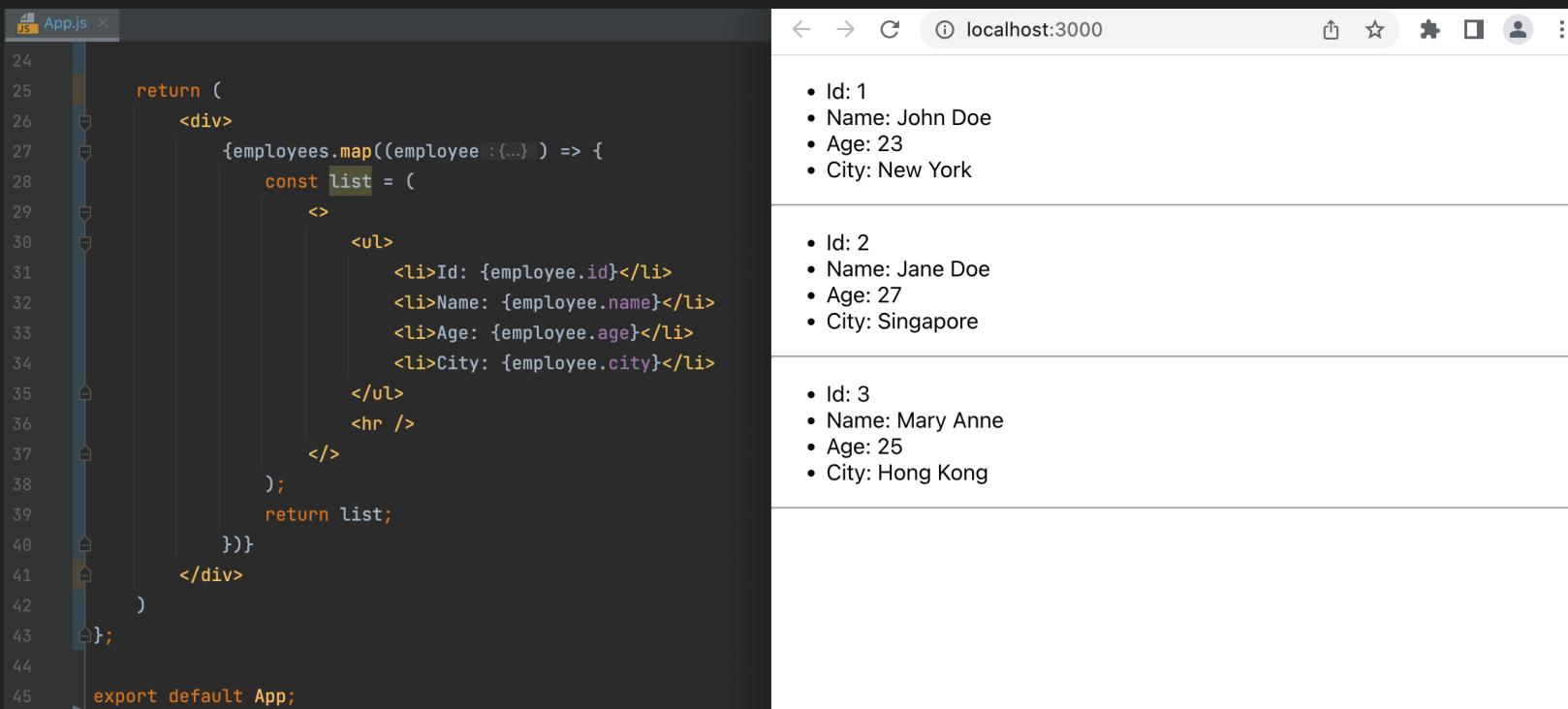
```
1 import React from "react";
2
3 const App = () => {
4     let employees = [
5         {
6             id: 1,
7             name: "John Doe",
8             age: 23,
9             city: "New York",
10            },
11            {
12                id: 2,
13                name: "Jane Doe",
14                age: 27,
15                city: "Singapore",
16            },
17            {
18                id: 3,
19                name: "Mary Anne",
20                age: 25,
21                city: "Hong Kong",
22            },
23        ];
24
```

Browser Output:

- Id: 1
 - Name: John Doe
 - Age: 23
 - City: New York
- Id: 2
 - Name: Jane Doe
 - Age: 27
 - City: Singapore
- Id: 3
 - Name: Mary Anne
 - Age: 25
 - City: Hong Kong

App.js

List and Keys



The image shows a code editor on the left and a browser window on the right.

Code Editor (App.js):

```
24
25     return (
26         <div>
27             {employees.map((employee : {...} ) => {
28                 const list = (
29                     <>
30                         <ul>
31                             <li>Id: {employee.id}</li>
32                             <li>Name: {employee.name}</li>
33                             <li>Age: {employee.age}</li>
34                             <li>City: {employee.city}</li>
35                         </ul>
36                         <hr />
37                     );
38                 return list;
39             })}
40         );
41     );
42 }
43
44 export default App;
```

Browser Output:

- Id: 1
 - Name: John Doe
 - Age: 23
 - City: New York
- Id: 2
 - Name: Jane Doe
 - Age: 27
 - City: Singapore
- Id: 3
 - Name: Mary Anne
 - Age: 25
 - City: Hong Kong

App.js

State

The image shows a code editor on the left and a browser window on the right. The code editor displays the file `App.js` with syntax highlighting for JavaScript. The browser window shows the rendered output of the application at `localhost:3000`.

```
JS App.js ×
24 const App = () => {
25   const name = "John Doe"
26
27   const [count, setCount] = useState(initialState: 0);
28   const [prevCount, setPrevCount] = useState(initialState: 0);
29
30   const handleClick = () => {
31     setCount( value: (prev : number) => {
32       setPrevCount(prev);
33     });
34     setCount( value: count + 1);
35   };
36
37   return (
38     <>
39       <div>Hello World!</div>
40       <div>{ name }</div>
41       <div>{ 2 + 2 }</div>
42       <Welcome/>
43       <Data/>
44       <h3>Current count: {count}</h3>
45       <h3>Previous count: {prevCount}</h3>
46       <button onClick={handleClick}>Increment</button>
47     </>
48   );
49 }
50
51 export default App;
```

The browser output includes:

- Hello World!
- John Doe
- 4
- Welcome to React
 - Dog
 - Cat
- Current count: 10**
- Previous count: 9**
- Increment** button

App.js

State Hook

The image shows a split-screen development environment. On the left is a code editor with the file `App.js` open. The code uses the `useState` hook to manage state. It initializes a name to "John Doe" and two counts: a current count of 0 and a previous count of 0. It includes a function `handleClick` that increments the current count and saves the previous value. The `return` statement contains JSX for displaying the name, a calculation, a welcome message, data, and two counts. A button allows the user to increment the count. On the right is a browser window displaying the application's output at `localhost:3000`. The page shows "Hello World!", "John Doe", "4", and "Welcome to React" with a list of animals. Below this, it displays "Current count: 10" and "Previous count: 9", with an "Increment" button.

```
const App = () => {
  const name = "John Doe"

  const [count, setCount] = useState( initialState: 0 );
  const [prevCount, setPrevCount] = useState( initialState: 0 );

  const handleClick = () => {
    setCount( value: (prev : number) => {
      setPrevCount(prev);
    });
    setCount( value: count + 1 );
  };

  return (
    <>
      <div>Hello World!</div>
      <div>{ name }</div>
      <div>{ 2 + 2 }</div>
      <Welcome/>
      <Data/>
      <h3>Current count: {count}</h3>
      <h3>Previous count: {prevCount}</h3>
      <button onClick={handleClick}>Increment</button>
    </>
  );
};

export default App;
```

localhost:3000

Hello World!

John Doe

4

Welcome to React

- Dog
- Cat

Current count: 10

Previous count: 9

Increment

App.js

Effect Hook

```
JS App.js ×
22
23  const Countdown = ({ hr, min, sec }) => {
24      const [over, setOver] = useState(initialState: false);
25      const [paused, setPaused] = useState(initialState: true);
26      const [[h, m, s], setTime] = useState(initialState: [hr, min, sec]);
27
28      const tick = () => {
29          if (paused || over) {
30              return;
31          }
32
33          if (h === 0 && m === 0 && s === 0) {
34              setOver(value: true);
35          } else if (m === 0 && s === 0) {
36              setTime(value: [h - 1, 59, 59]);
37          } else if (s === 0) {
38              setTime(value: [h, m - 1, 59]);
39          } else {
40              setTime(value: [h, m, s - 1]);
41          }
42
43          const handleReset = () => {
44              setTime(value: [hr, min, sec]);
45              setPaused(value: true);
46              setOver(value: false);
47          };
48
49          const handlePause = () => setPaused(!paused);
50
51          const fmt = (val) => val.toString().padStart(2, '0');
52
53          useEffect(effect: () => {
54              let ticker = setInterval(handler: () => tick(), timeout: 1000);
55              return () => {
56                  clearInterval(ticker);
57              };
58          });
59      };
60  };
61
62  const App = () => {
63      const [hr, setHr] = useState(0);
64      const [min, setMin] = useState(0);
65      const [sec, setSec] = useState(0);
66
67      const handleReset = () => {
68          setHr(0);
69          setMin(0);
70          setSec(0);
71      };
72
73      const handlePause = () => {
74          setPaused(true);
75      };
76
77      const handleStart = () => {
78          setPaused(false);
79      };
80
81      const handleReset = () => {
82          setHr(0);
83          setMin(0);
84          setSec(0);
85      };
86
87      const handleReset = () => {
88          setHr(0);
89          setMin(0);
90          setSec(0);
91      };
92
93      const handleReset = () => {
94          setHr(0);
95          setMin(0);
96          setSec(0);
97      };
98
99      const handleReset = () => {
100         setHr(0);
101         setMin(0);
102         setSec(0);
103     };
104
105     return (
106         <div>
107             <h1>Countdown</h1>
108             <input type="button" value="Reset" onClick={handleReset} />
109             <input type="button" value="Start" onClick={handleStart} />
110             <input type="button" value="Pause" onClick={handlePause} />
111             <hr />
112             <div>
113                 <strong>Hours:</strong> {hr}
114                 <strong>Minutes:</strong> {min}
115                 <strong>Seconds:</strong> {sec}
116             </div>
117             <Countdown hr={hr} min={min} sec={sec} />
118         </div>
119     );
120 }
121
122 export default App;
```

Hands-on



<https://pokeapi.co>

API

Try it now!

pokemon?

Copy

Submit

Need a hint? Try [pokemon/ditto](#), [pokemon-species/aegislash](#), [tvne/3](#), [ability/battle-armor](#), or [pokemon?limit=100000&offset=0](#).

Direct link to results: <https://pokeapi.co/api/v2/pokemon?>

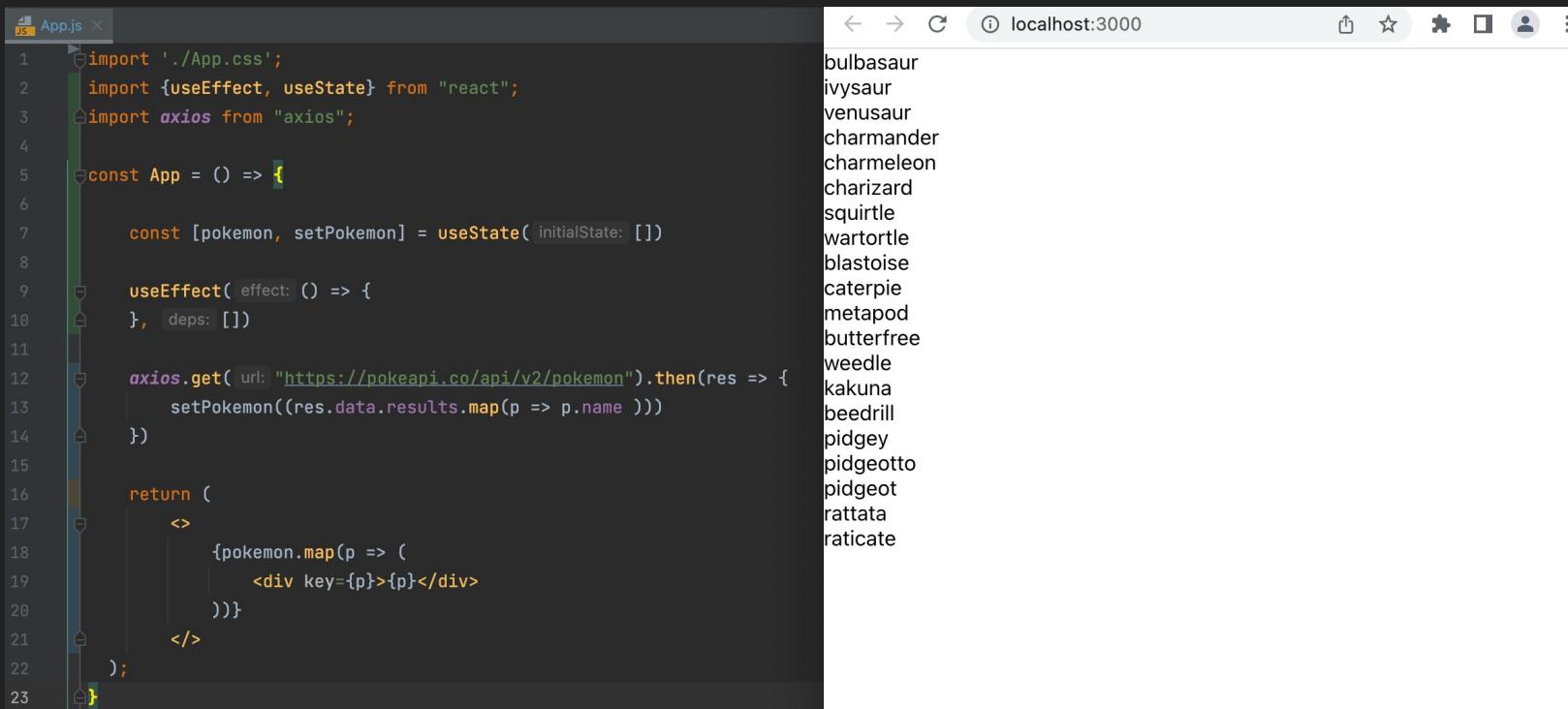
Resource for pokemon?

```
count: 1154
next: "https://pokeapi.co/api/v2/pokemon?offset=20&limit=20"
previous: null
▼ results: □ 20 items
  ▼ 0: {} 2 keys
    name: "bulbasaur"
    url: "https://pokeapi.co/api/v2/pokemon/1/"
  ▼ 1: {} 2 keys
    name: "ivysaur"
    url: "https://pokeapi.co/api/v2/pokemon/2/"
  ▼ 2: {} 2 keys
    name: "venusaur"
    url: "https://pokeapi.co/api/v2/pokemon/3/"
  ▼ 3: {} 2 keys
    name: "charmander"
    url: "https://pokeapi.co/api/v2/pokemon/4/"
  ▼ 4: {} 2 keys
    name: "charmeleon"
    url: "https://pokeapi.co/api/v2/pokemon/5/"
  ▼ 5: {} 2 keys
    name: "charizard"
    url: "https://pokeapi.co/api/v2/pokemon/6/"
  ▼ 6: {} 2 keys
    name: "squirtle"
    url: "https://pokeapi.co/api/v2/pokemon/7/"
```

[View raw JSON \(1.956 kB, 87 lines\)](#)

App.js

Axios, useEffect, useState, map



The image shows a development environment with two windows side-by-side. On the left is a code editor with the file 'App.js' open. The code uses the Axios library to fetch a list of Pokémon names from an API, then maps over the results to render them in a list. On the right is a web browser window displaying the results of the code execution.

```
App.js
1 import './App.css';
2 import {useEffect, useState} from "react";
3 import axios from "axios";
4
5 const App = () => {
6
7   const [pokemon, setPokemon] = useState( initialState: [] )
8
9   useEffect( effect: () => {
10     }, deps: [] )
11
12   axios.get( url: "https://pokeapi.co/api/v2/pokemon" ).then(res => {
13     setPokemon((res.data.results.map(p => p.name )))
14   })
15
16   return (
17     <>
18       {pokemon.map(p => (
19         <div key={p}>{p}</div>
20       ))}
21     </>
22   );
23 }
```

localhost:3000

bulbasaur
ivysaur
venusaur
charmander
charmeleon
charizard
squirtle
wartortle
blastoise
caterpie
metapod
butterfree
weedle
kakuna
beedrill
pidgey
pidgeotto
pidgeot
rattata
raticate

App.js

Pagination

The image shows a split-screen development environment. On the left is a code editor with the file `App.js` open, displaying the following code:

```
1 import './App.css';
2 import {useEffect, useState} from "react";
3 import axios from "axios";
4
5 const App = () => {
6
7   const [pokemon, setPokemon] = useState( initialState: [] );
8   const [currentPageUrl, setCurrentPageUrl] = useState( initialState: "https://pokeapi.co/api/v2/pokemon" );
9   const [nextPageUrl, setNextPageUrl] = useState();
10  const [prevPageUrl, setPrevPageUrl] = useState();
11  const [loading, setLoading] = useState( initialState: true );
12
13  useEffect( effect: () => {
14    setLoading( value: true );
15    let cancel;
16    axios.get(currentPageUrl, config: {
17      cancelToken: new axios.CancelToken( executor: c => cancel = c )
18    }).then(res => {
19      setLoading( value: false );
20      setNextPageUrl(res.data.next);
21      setPrevPageUrl(res.data.previous);
22      setPokemon(res.data.results.map(p => p.name));
23    })
24
25    return () => cancel();
26  }, deps: [currentPageUrl] );
27
28  function gotoNextPage() {
29    setCurrentPageUrl(nextPageUrl);
30  }
31
32  function gotoPrevPage() {
33    setCurrentPageUrl(prevPageUrl);
34  }
35
36  if (loading) return "Loading...";
37
38  return (
39    <>
40      {pokemon.map(p => (
41        <div key={p}>{p}</div>
42      ))}
43      <div>
44        {gotoPrevPage ? <button onClick={gotoPrevPage}>Previous</button> : null}
45        {gotoNextPage ? <button onClick={gotoNextPage}>Next</button> : null}
46      </div>
47    </>
48  );
49
50}
51
52 export default App;
```

On the right is a browser window showing the rendered output at `localhost:3000`. The page displays a list of 20 Pokémon names: bulbasaur, ivysaur, venusaur, charmander, charmeleon, charizard, squirtle, wartortle, blastoise, caterpie, metapod, butterfree, weedle, kakuna, beedrill, pidgey, pidgeotto, pidgeot, rattata, raticate. Below the list are two buttons: "Previous" and "Next".