ENSF 381
Full Stack Web Development

Lecture 20: Components

Slides: Ahmad Abdellatif, PhD

Instructor: Novarun Deb, PhD

UNIVERSITY OF CALGARY

Outline

- JSX.
- Components.
- Lists.
- Modularization.

React JSX

- JSX stands for JavaScript XML.

- JSX allows us to write HTML elements and components within JavaScript code:

```
const element = <h1>Hello, World!</h1>;
```

- Under the hood, React transforms this JSX code into JavaScript code using a process called **transpilation**, **making it compatible with browsers**.

React Component

- A reusable and self-contained building block for building user interfaces.

- They can represent anything from simple UI elements, such as buttons or input fields, to more complex structures like entire sections of a webpage or even entire pages.

- Functions that return HTML elements.

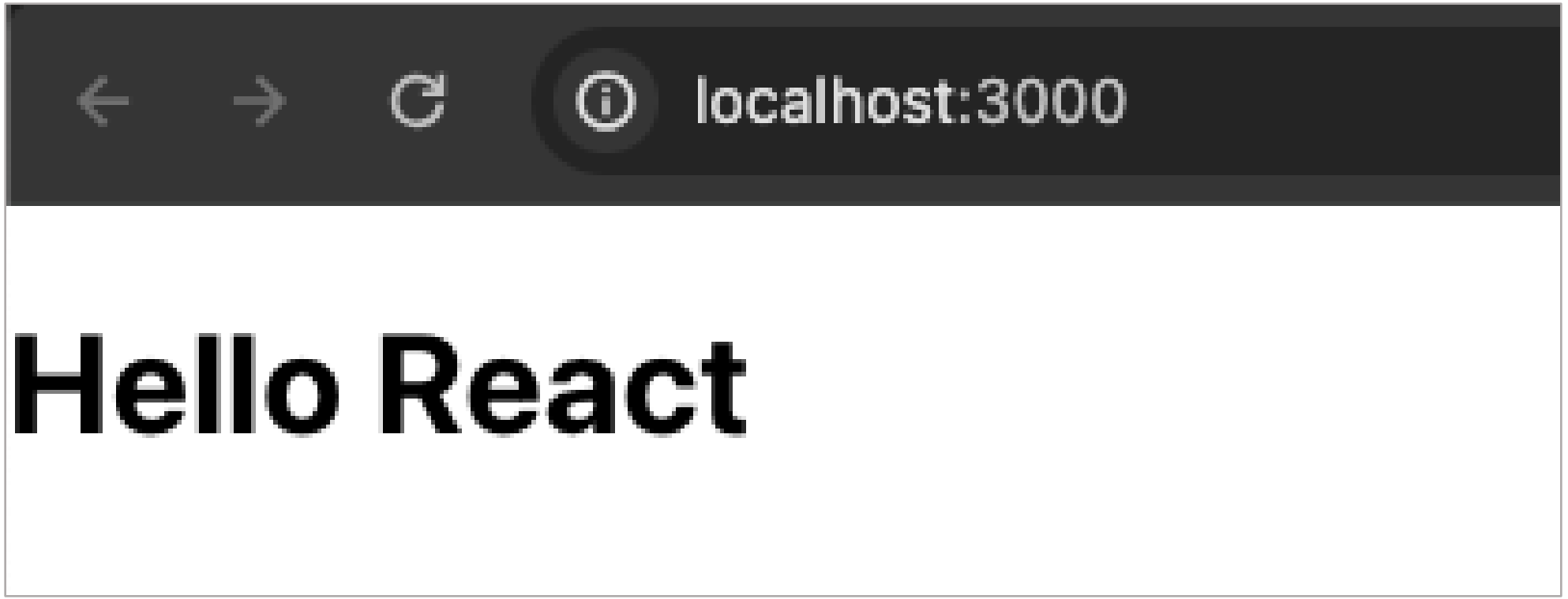
Component - example

In the src/App.js:

```
import React from 'react';

function App() {

return (
<div>
<h1>Hello React</h1>
</div>
);

}

export default App; // Used to export the main functionality or
component from a module.
```

Component - example

Run: `npm start`

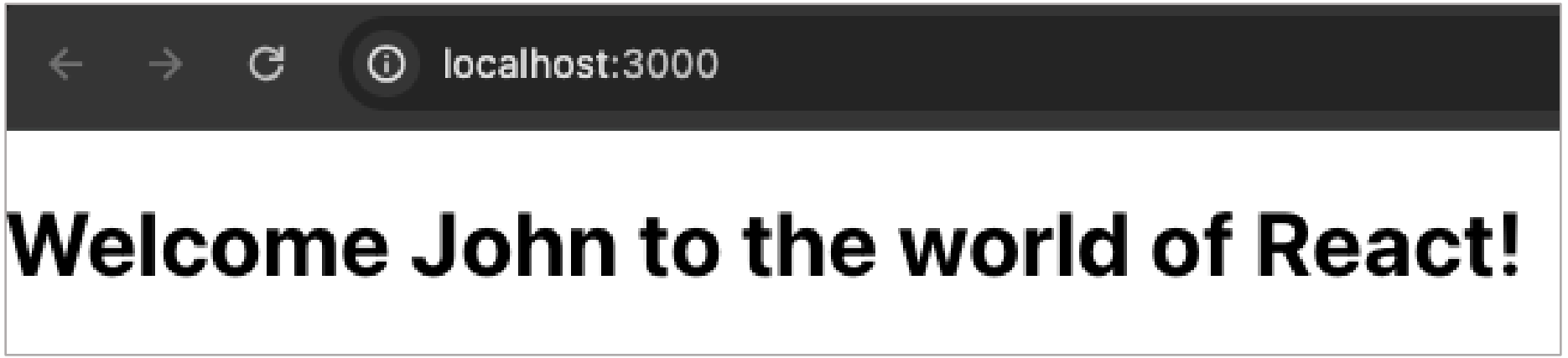


Component – example 2

```
import React from 'react';
function App() {

const name = "John"

return (
<div>
<h1>Welcome {name} to the world of React!</h1>
</div>
);
}
export default App;
```

Embed JavaScript expressions within JSX.

Component – example 2

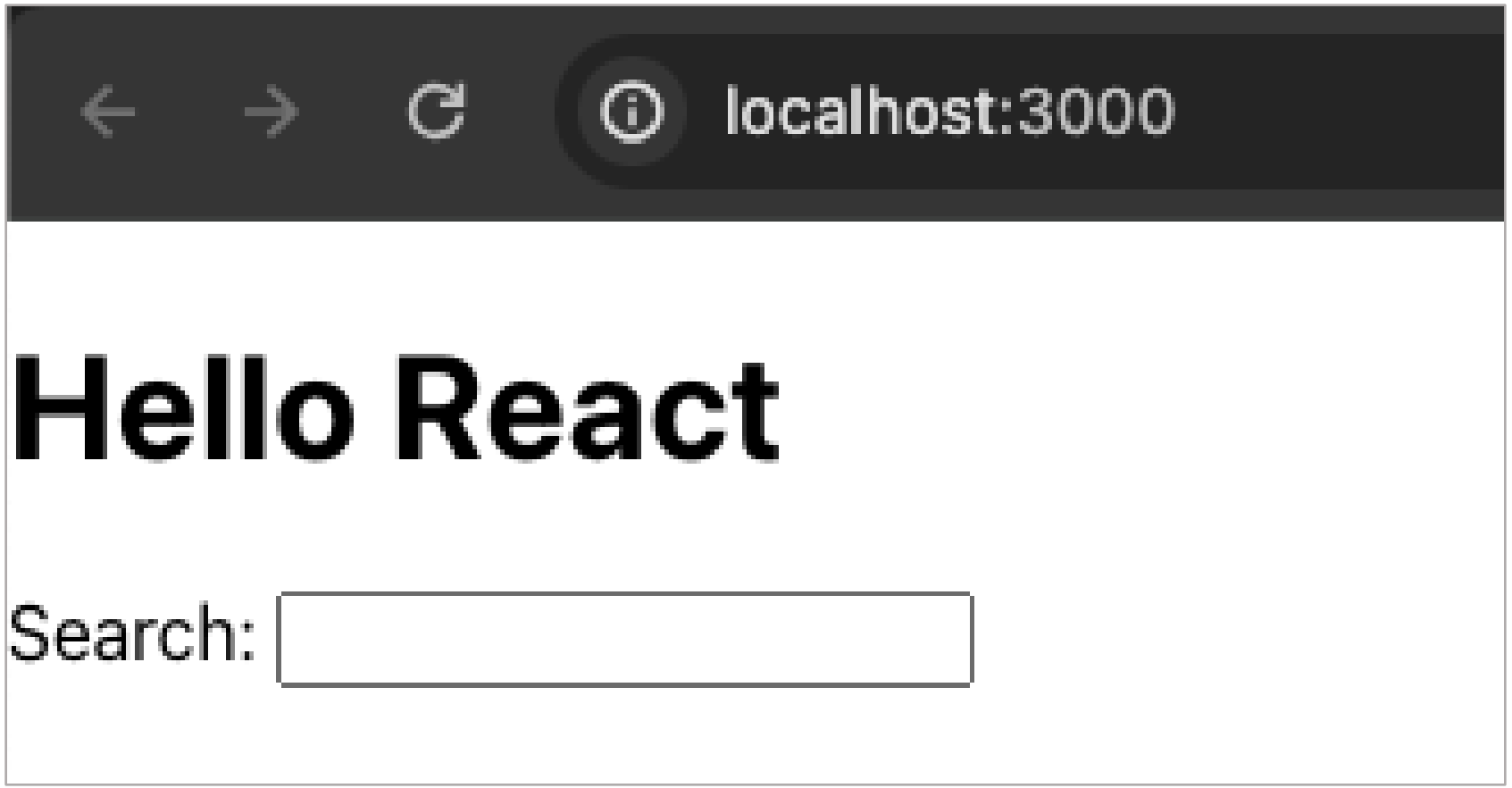


Component – example 3

```
import React from 'react';
const title = 'React';
function App() {
return (
<div>
<h1>Hello {title}</h1>
<label>Search: </label>
<input id="search" type="text"/>
</div>
);
}
export default App;
```

Src: The Road to React: The React.js with Hooks in JavaScript Book

Component – example 3

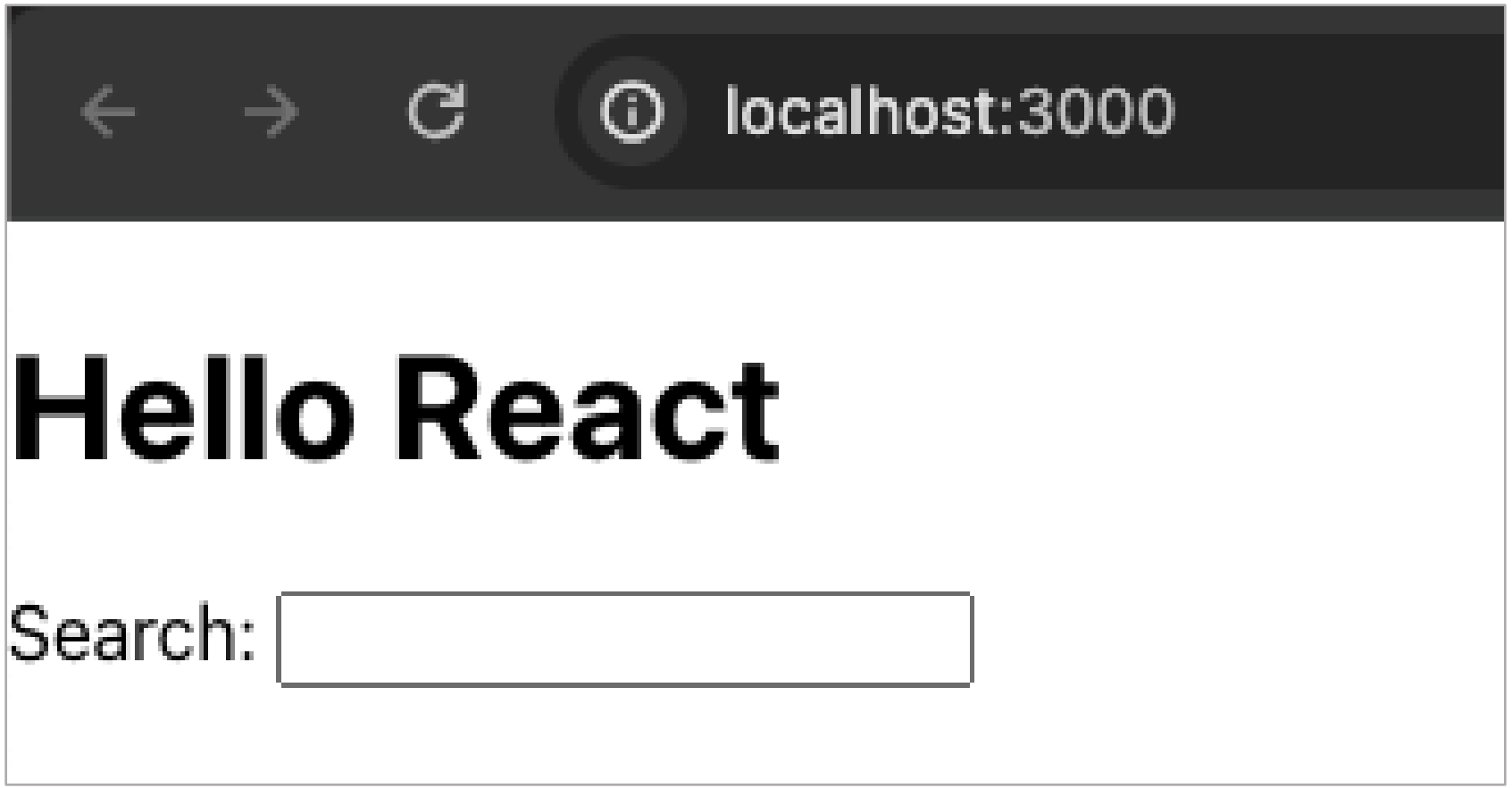


Component – example 4

```
import React from 'react';
function getTitle(title) {
return title;
}
function App() {
return (
<div>
<h1>Hello {getTitle('React')}</h1>
<label htmlFor="search">Search: </label>
<input id="search" type="text" />
</div>
);
}
export default App;
```


Src: The Road to React: The React.js with Hooks in JavaScript Book

Component – example 4



Map function in JavaScript

- An array method that is used to iterate over each element of an array and apply a given function to each element.

- The result is a new array where each element is the result of applying the provided function to the corresponding element of the original array.

- The original array **remains unchanged**.

Map function - Example

```
// Original array
const numbers = [1, 2, 3, 4, 5];

// Using map to create a new array where each element is doubled
const doubledNumbers = numbers.map(function (number) {
  return number * 2;
});


// Output [2, 4, 6, 8, 10]
console.log(doubledNumbers);
```

Lists in React

- A collection of elements or components rendered in a specific order.

- Commonly used to **display dynamic data**, where the number of items may vary, and you want to render each item in a repetitive structure.

List - Example

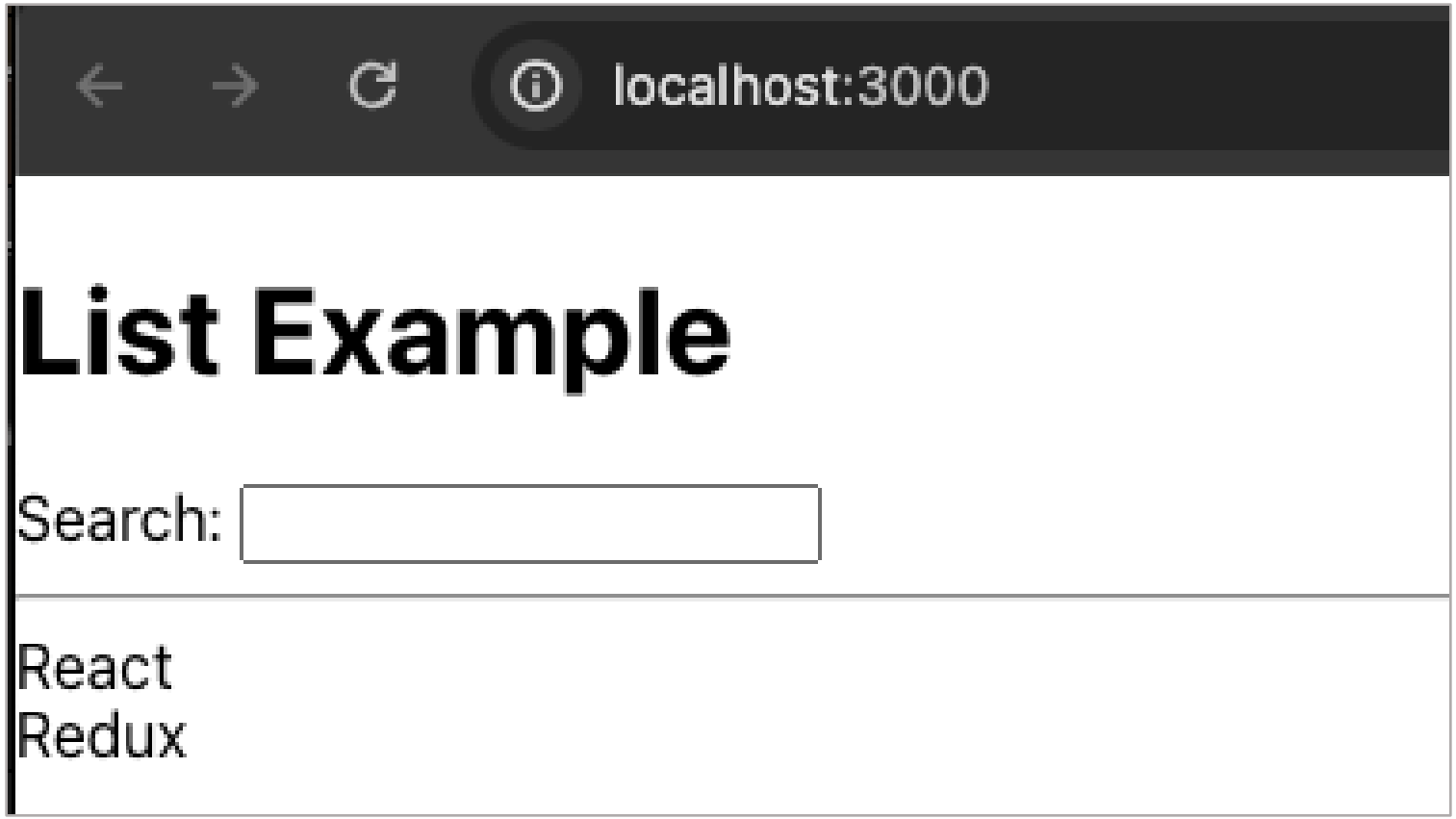
```
import React from 'react';
const list = [
{
title: 'React',
url: 'https://reactjs.org/',
author: 'Jordan Walke',
num_comments: 3,
points: 4,
objectID: 0,
},
{
title: 'Redux',
url: 'https://redux.js.org/',
author: 'Dan Abramov, Andrew Clark',
num_comments: 2,
points: 5,
objectID: 1,
},
];
```

List - Example

```
function App() {
return (
<div>
<h1>List Example</h1>
<label htmlFor="search">Search: </label>
<input id="search" type="text" />
<hr />
{list.map(function(item) {
return <div>{item.title}</div>;
})}
</div>
);
}

export default App;
```


Src: The Road to React: The React.js with Hooks in JavaScript Book

List - Example



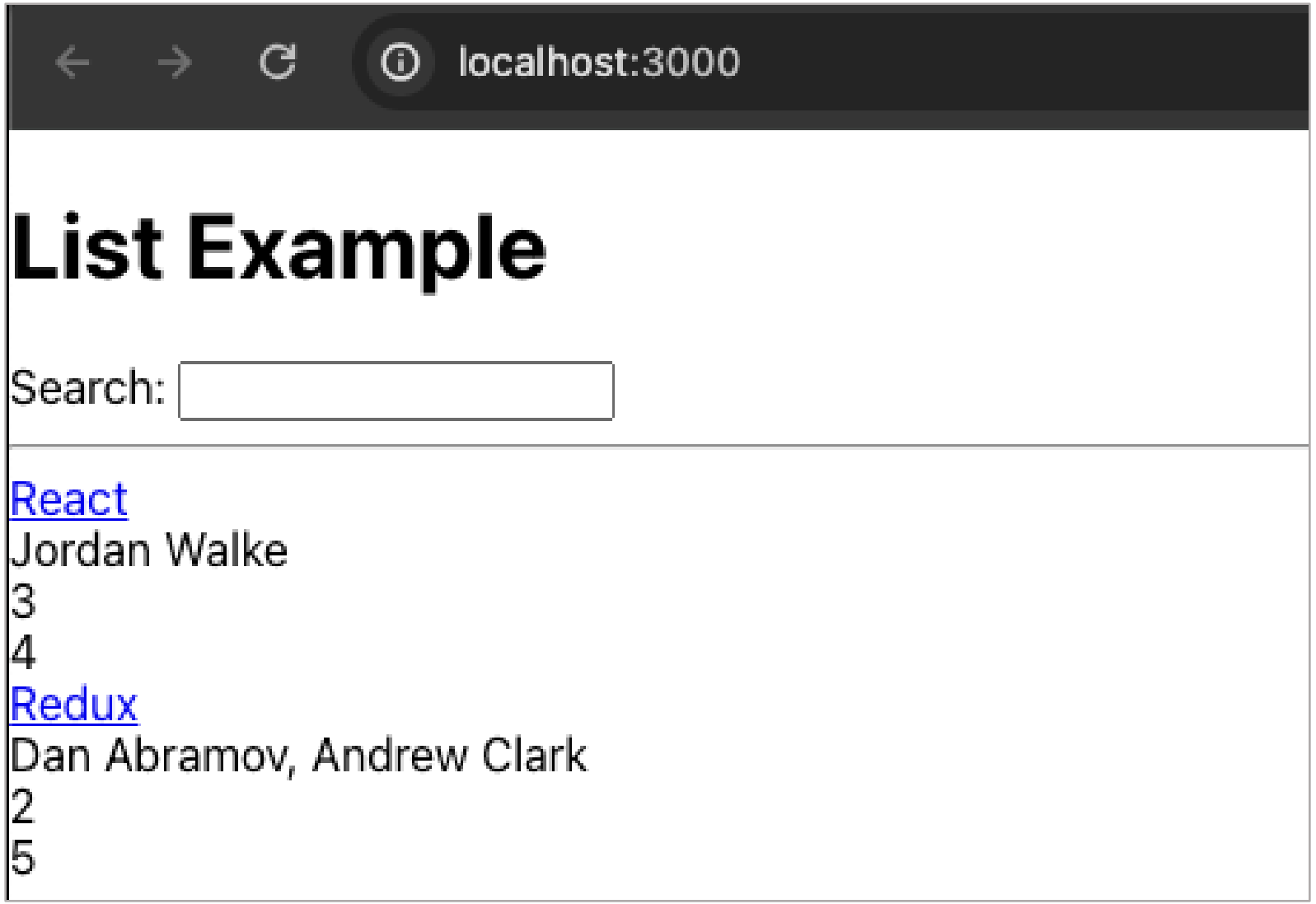
Rendering item's properties

Modify the App function to:

```
function App() {
return (
<div>
<h1>List Example</h1>
<label htmlFor="search">Search: </label>
<input id="search" type="text" />
<hr />
{list.map(function(item) {
return (
<div id={item.objectID}>
<div>
<a href={item.url}>{item.title}</a>
</div>
<div>{item.author}</div>
<div>{item.num_comments}</div>
<div>{item.points}</div>
</div>
);
})}
</div>
);}
```

Src: The Road to React: The React.js with Hooks in JavaScript Book

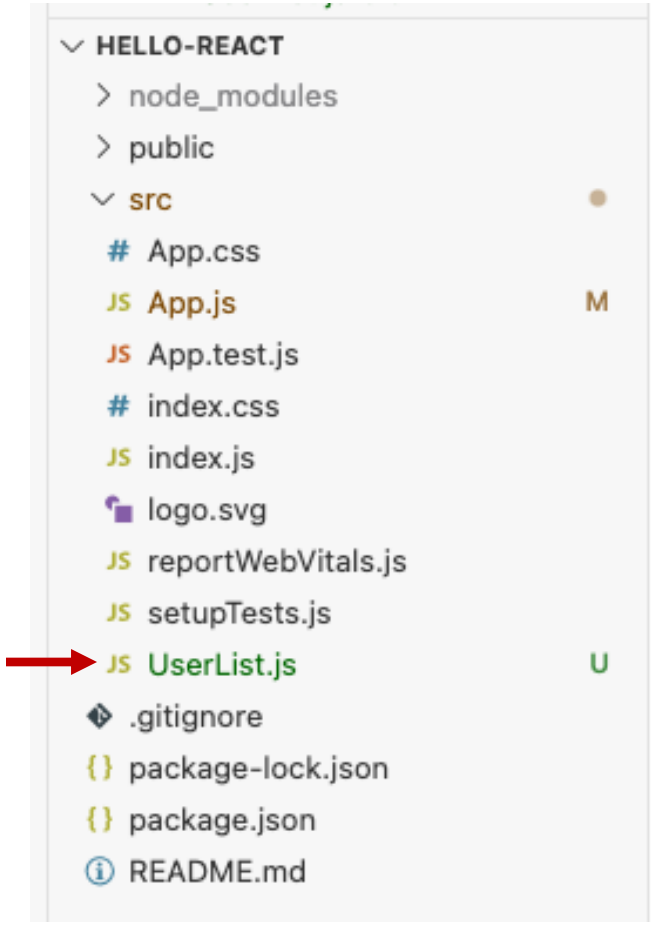
Rendering item’s properties



Modularization

- We need to modularize the code to enhance its maintainability and readability.

- Create the list in a separate file:



Modularization - Code in the UserList component

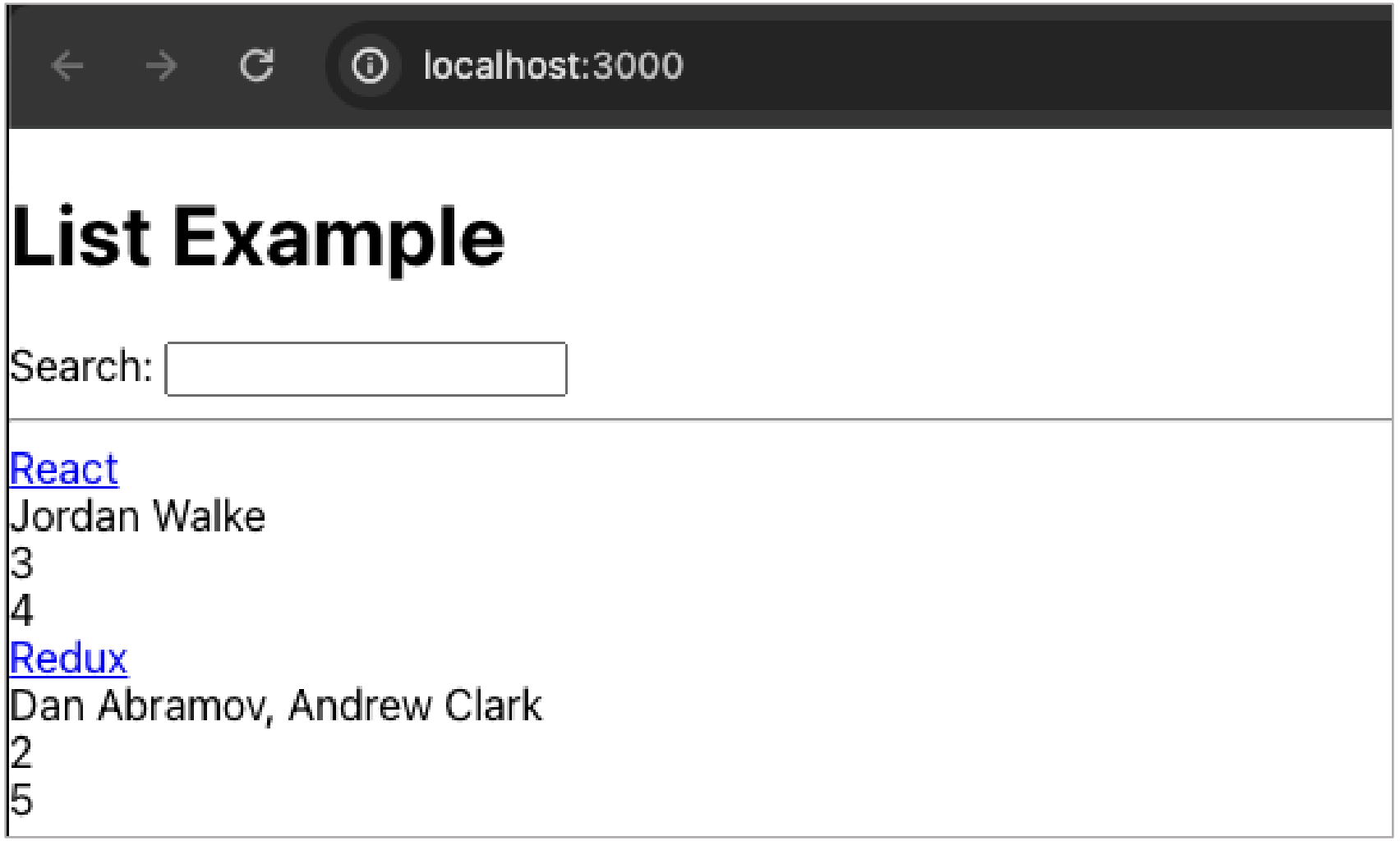
```
import React from 'react';
const list = [
  {
  title: 'React',
  url: 'https://reactjs.org/',
  author: 'Jordan Walke',
  num_comments: 3,
  points: 4,
  objectID: 0,
  },
  {
  title: 'Redux',
  url: 'https://redux.js.org/',
  author: 'Dan Abramov, Andrew Clark',
  num_comments: 2,
  points: 5,
  objectID: 1,
  },
  ];

function UserList() {
return list.map(function(item) {
  return (
  <div id={item.objectID}>
  <div>
  <a href={item.url}>{item.title}</a>
  </div>
  <div>{item.author}</div>
  <div>{item.num_comments}</div>
  <div>{item.points}</div>
  </div>
);
});
}
export default UserList;
```

Modularization - Code in the App component

```
import React from 'react';
import UserList from './UserList';


function App() {
return(
  <div>
  <h1>List Example</h1>
  <label htmlFor="search">Search: </label>
  <input id="search" type="text" />
  <hr />

<UserList />

</div>
);
}

export default App;
```

This will render the content of UserList at that location within the App.js

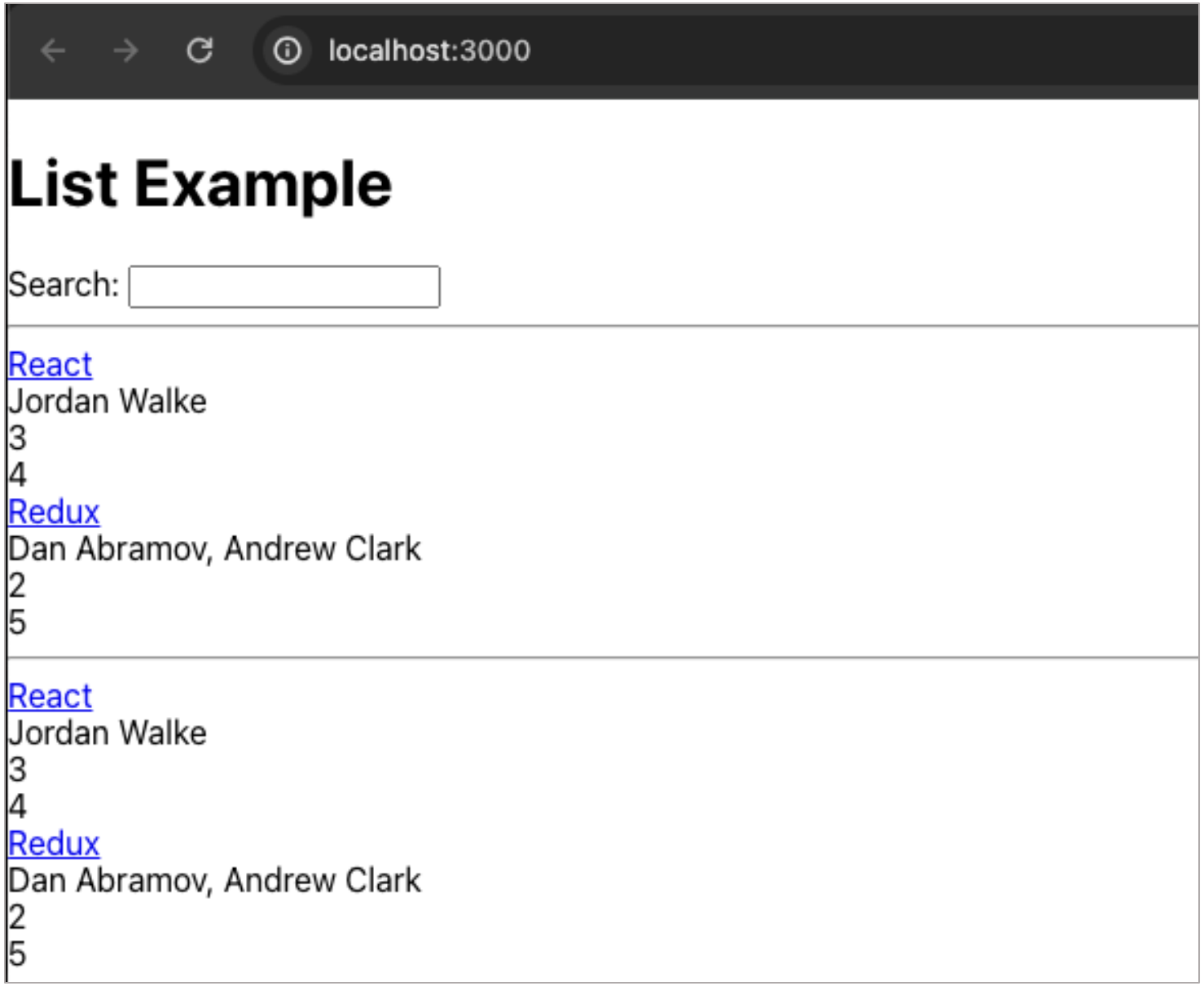
Modularization - output



Can we use the component more than once?

```
import React from 'react';
import UserList from './UserList';


function App() {
return(
  <div>
  <h1>List Example</h1>
  <label htmlFor="search">Search: </label>
  <input id="search" type="text" />
  <hr />

<UserList />
<hr />
<UserList />

</div>
);
}
export default App;
```

Can we use the component more than once?



Conditional statements

- Refer to the use of conditional statements or expressions in React applications to **dynamically render different content or components based on certain conditions**.

1. if Statements:

```
function MyComponent(isLoggedIn) {
  if (isLoggedIn) {
    return <p>Welcome, user!</p>;
  } else {
    return <p>Please log in.</p>;
  }
}
```

Conditional statements

2. Conditional (Ternary) Operator:

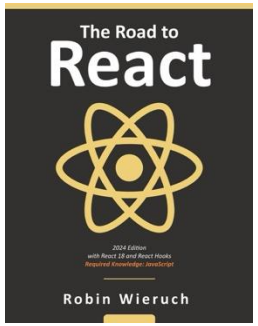
```
function MyComponent(isLoggedIn) {
  return isLoggedIn ? <p>Welcome, user!</p> : <p>Please log in.</p>;
}
```

3. Logical && Operator:

```
function MyComponent(isLoggedIn) {
  return isLoggedIn && <p>Welcome, user!</p> || <p>Please log in.</p>;
}
```

Questions

References



The Road to React: The React.js with Hooks in JavaScript