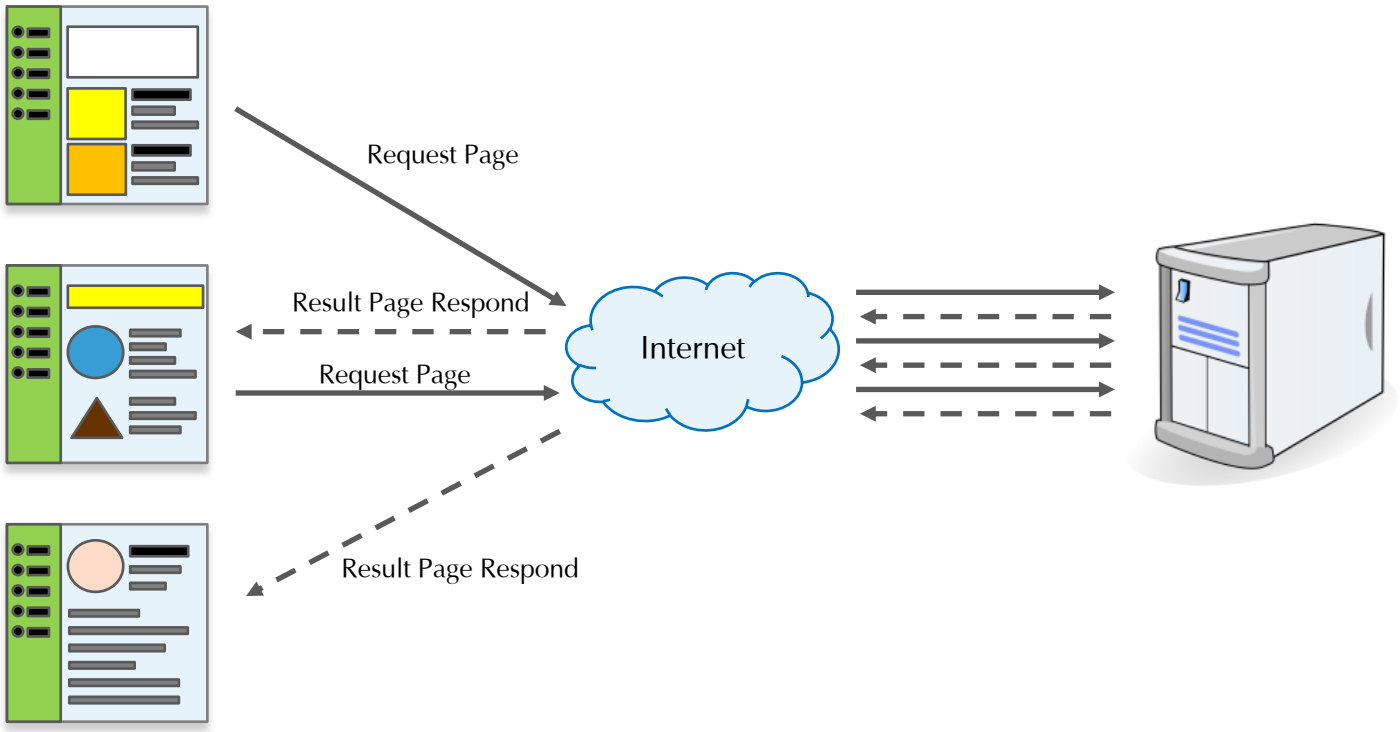
1. Intro to React

1.1 MPA vs SPA
1.2 Introduction of React
1.3 React Application Development Components
1.4 Development Environment Configuration

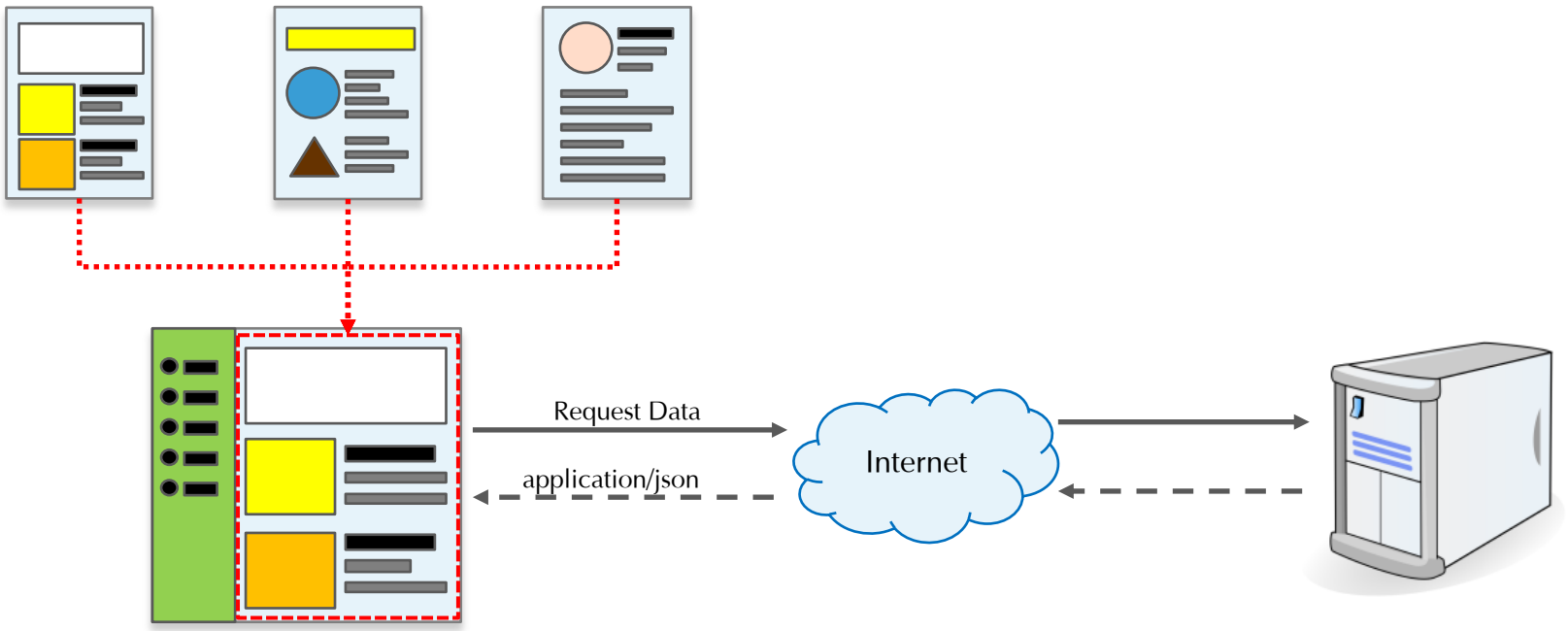
1.1 MPA vs SPA(1/2)

- ✓ MPA(Multi-Page Application) is a form that creates a page in the server and returns it according to the client's request.
- ✓ From the client's point of view, all the pages requesting the server exists, so the request only proceeds according to the user's needs.
- ✓ In MPA method, the server responds to requests from all clients, and must generate and respond to all pages.
- ✓ MPA method requests the entire page from server and refreshes page even there is a small change.



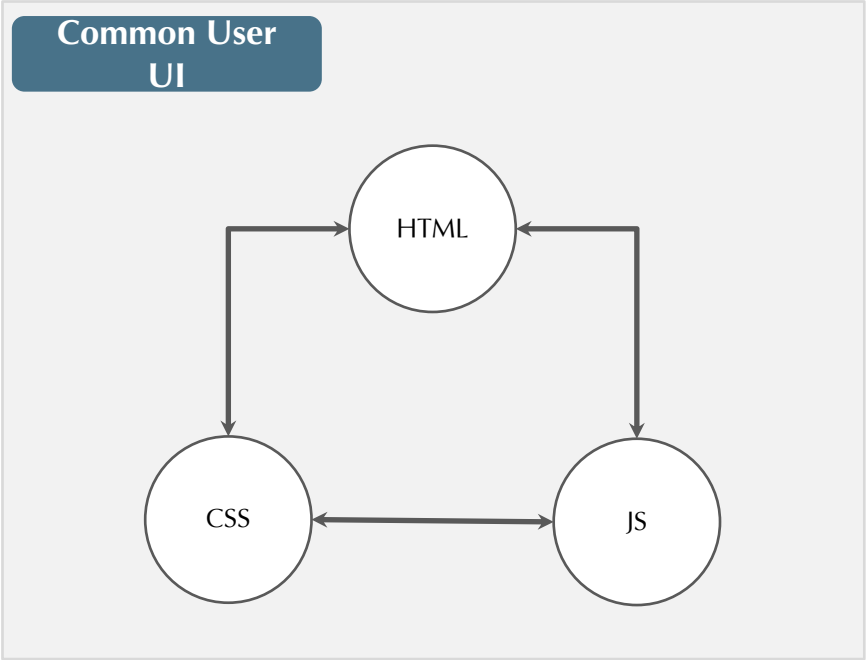
1.1 MPA vs SPA(2/2)

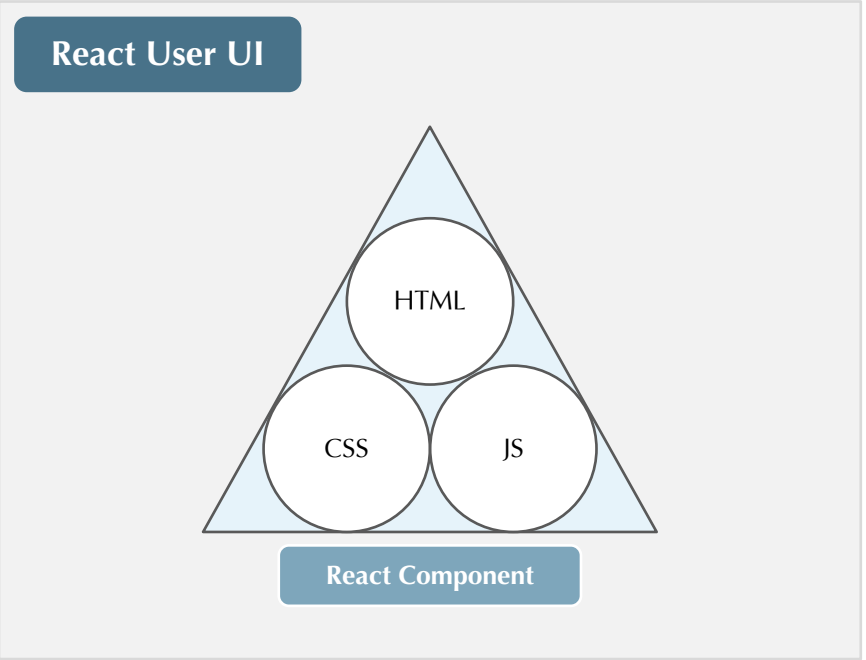
- ✓ SPA(Single-Page Application) is a method that receives a single page from the server and composes the page on the client.
- ✓ The client requests data (JSON) from the server and updates the page using the received data.
- ✓ From the server's point of view, the load is light because it only returns the requested data without creating a page even if a request occurs from multiple clients.



1.2 Introduction of React

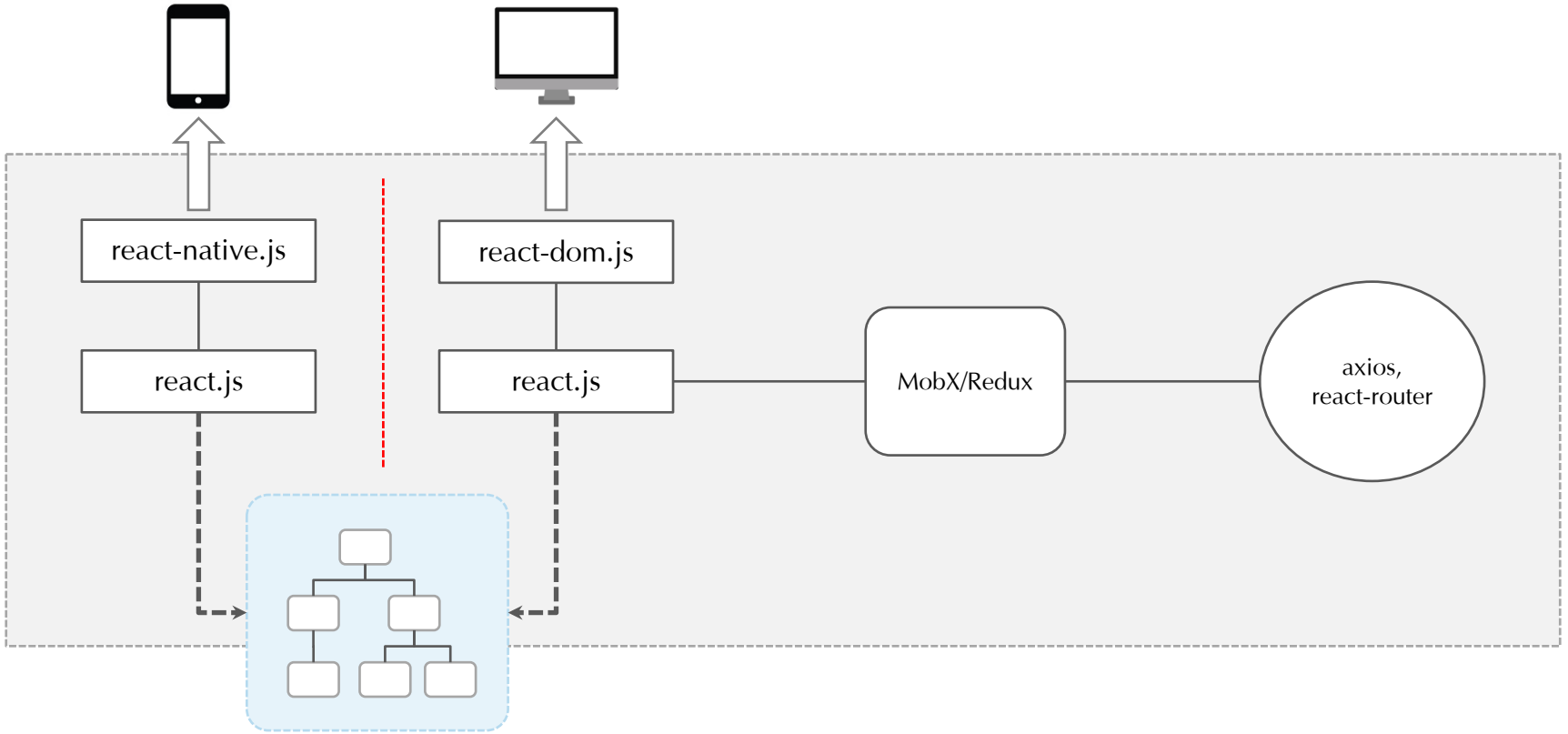
- ✓ React is a Javascript library for configuring SPA-based web user interface(UI).
- ✓ React makes it easy to create independent and reusable UI components.
- ✓ Creating a component means combining HTML, CSS, and JS that were previously managed separately into one element.
- ✓ Key to creating components that make up React UI is how to divide the page you want to configure into components.





1.3 React Application Development Components

- ✓ The basic unit that makes up a React application is an encapsulated component.
- ✓ The library used to define and utilize React components is the react.js core library.
- ✓ react-dom.js and react-native.js libraries that render UI elements on the screen and used with react.js.
- ✓ React is a library that focuses only on displaying UI elements on the page. Thus, it composes the application with various kinds of third-party libraries.



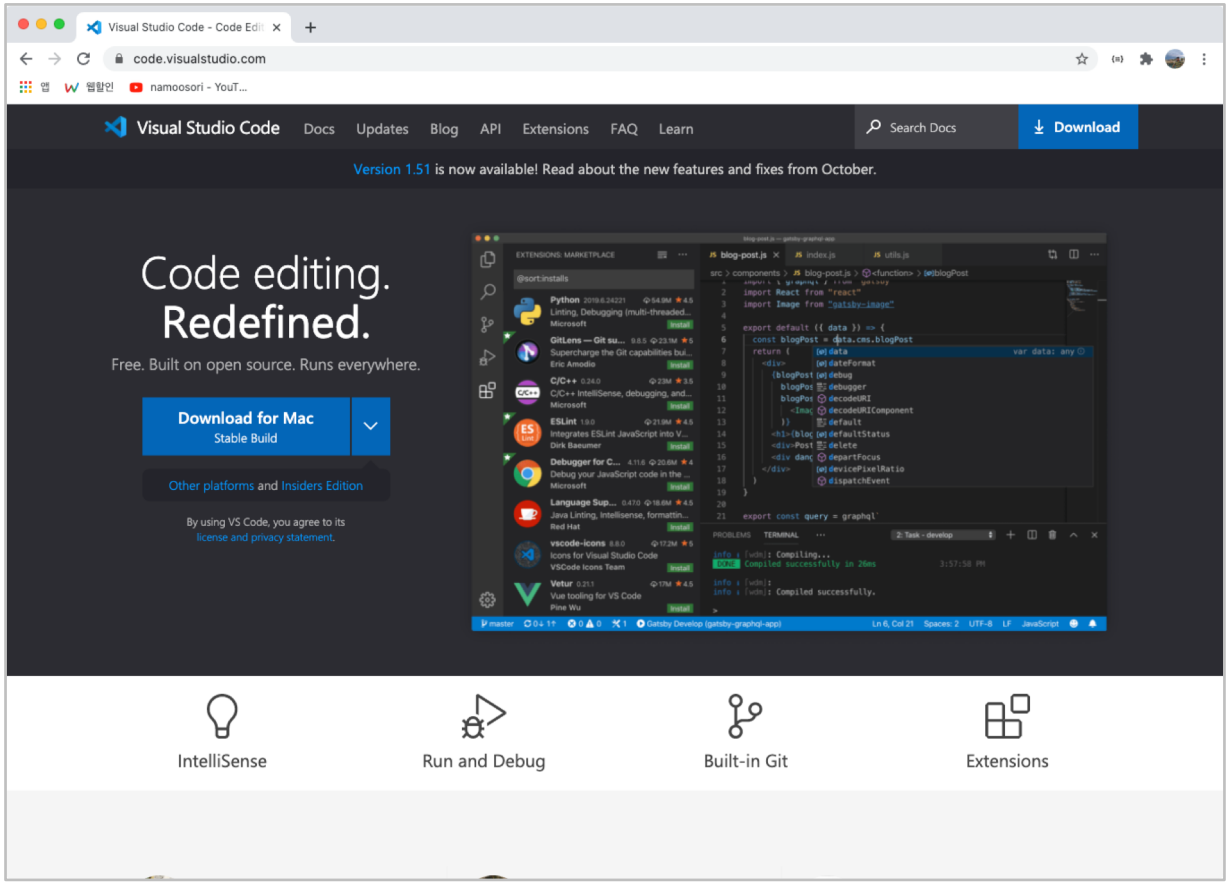
1.4 Development Environment Configuration (1/5)

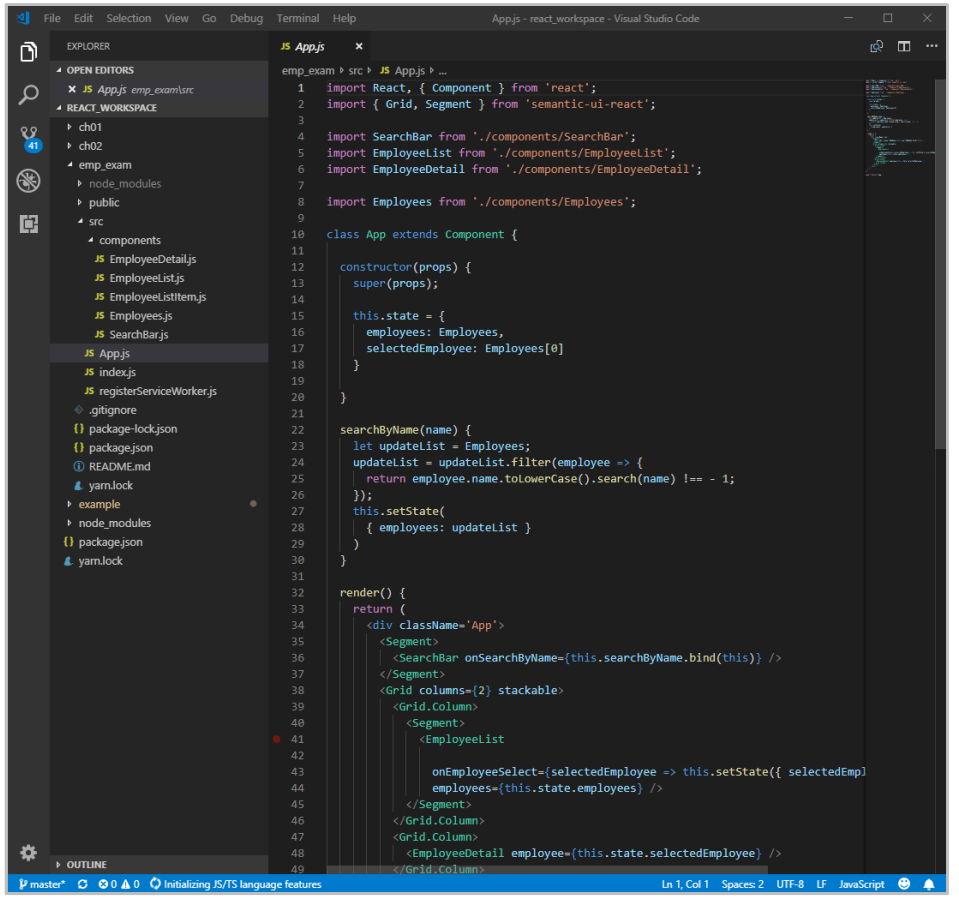
✓ **React development requires the following environment.**

- Code Editor: It is an editing tool for writing real code. Representative tools include Atom, Bracket, VS code (Visual Studio Code), etc.
- node.js & npm: node.js is an execution environment in which Javascript can run. Npm (node package manager) is a tool for managing various JavaScript packages, and a package is a set of modules available in node.js.
- Web Browser: A typical web browser suitable for UI development using React generally uses Google Chrome.
- git: Certain packages or modules require git installation.

1.4 Development Environment Configuration (2/5)

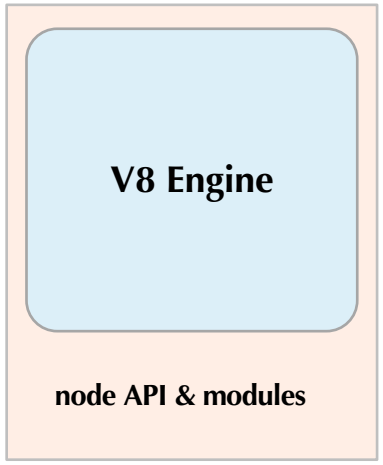
- ✓ Visual Studio Code is a code editing tool developed by MS.
- ✓ Download the appropriate version for your development computer from the site through the following URL.
  - https://code.visualstudio.com



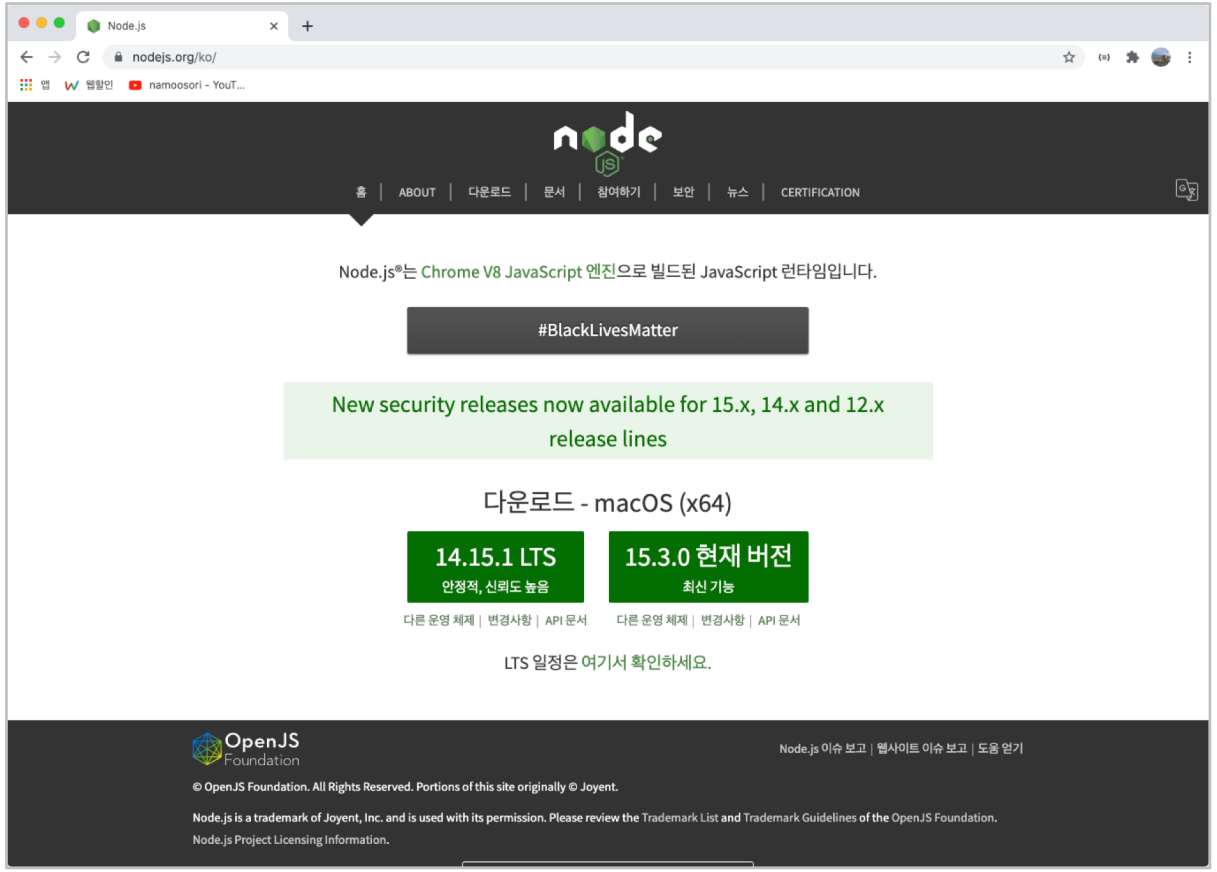


1.4 Development Environment Configuration (3/5)

- ✓ Javascript was basically developed as a web-based language. Therefore, the scope of execution was limited to the web browser.
- ✓ node.js provides a JavaScript execution environment based on the V8 engine. This means that you can run JavaScript in environments other than web browsers.
- ✓ Install node.js suitable for the development environment through the following URL.
  - https://nodejs.org/

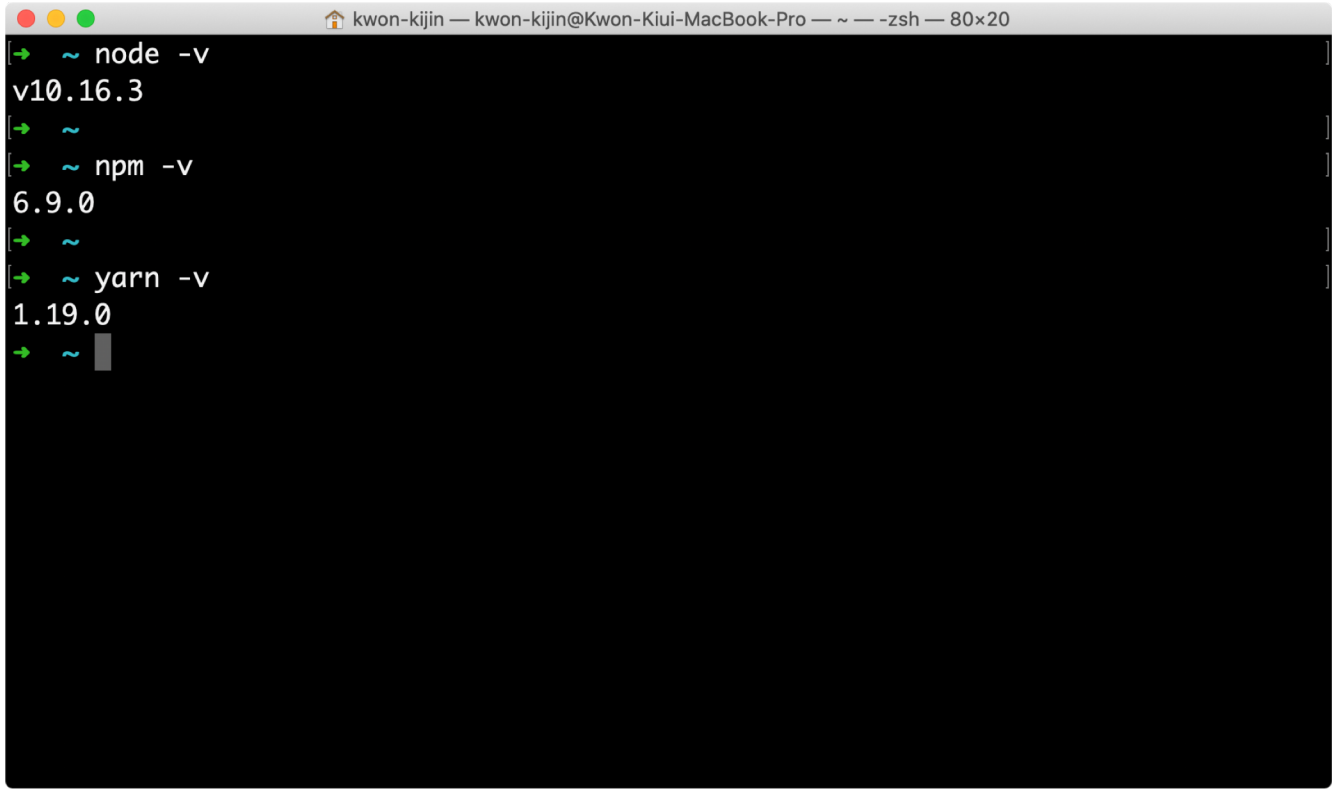


node.js



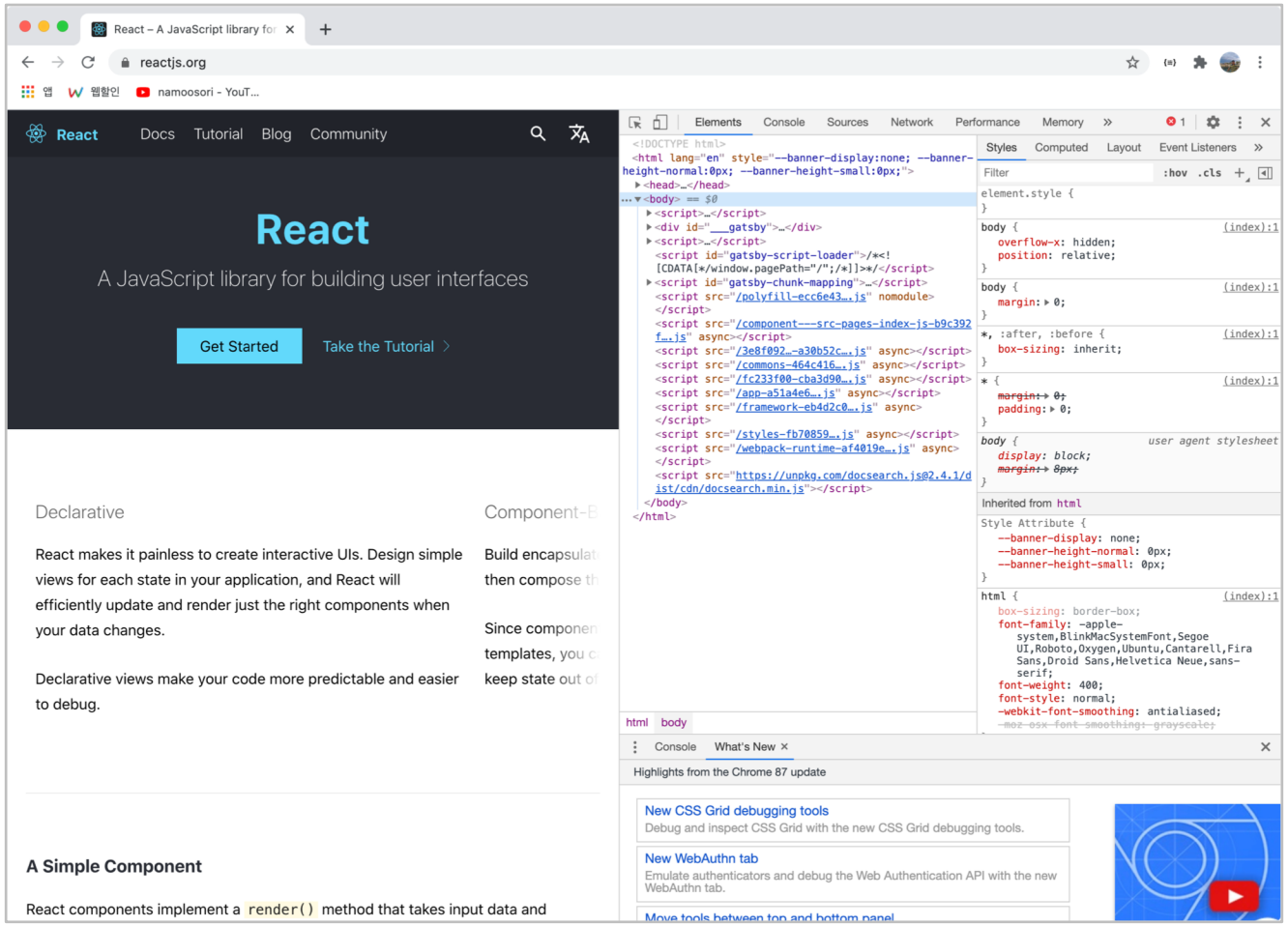
1.4 Development Environment Configuration (4/5)

- ✓ **npm(Node Package Module) is a tool for managing JavaScript packages and modules.**
- ✓ **npm is installed by default when node.js is installed. Confirmation of installation can be confirmed with the following command.**
  - node -v : check node version.
  - npm -v : check npm version.
  - yarn -v : check yarn version.



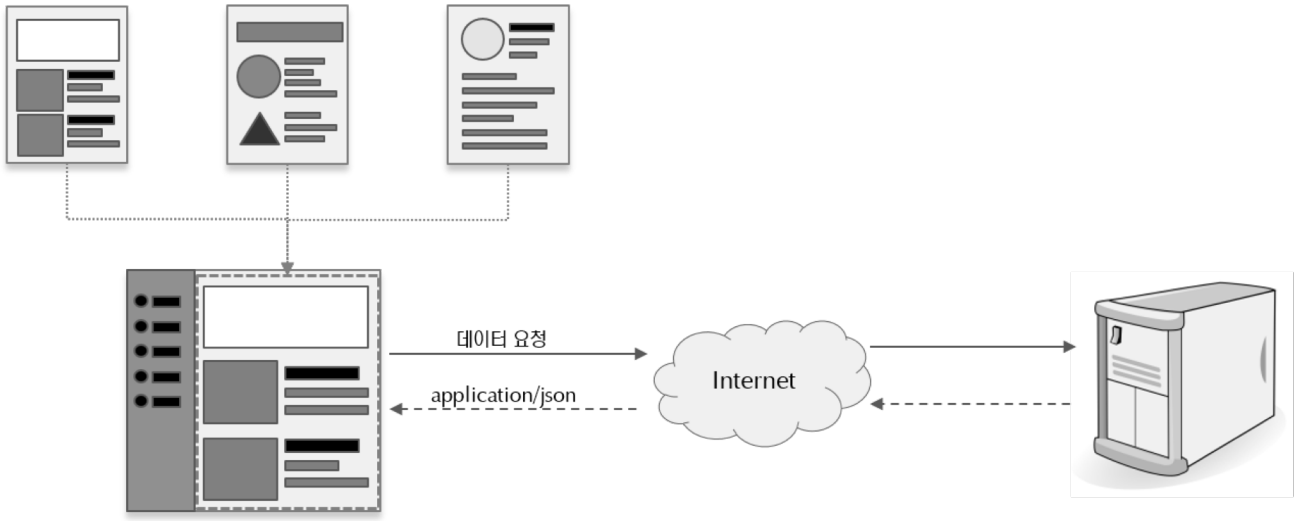
1.4 Development Environment Configuration (5/5)

- ✓ The web browser can be used without separate installation such as IE, Chrome, and Firefox.
- ✓ The web browser to be used for development must have developer tools available.
- ✓ Most web browsers support developer tools.



Summary

- ✓ React is a library of UI components made in JavaScript and uses these independent components to compose UI.
- ✓ MPA Method receives and displays the entire information about the new page from the server when the client requests it.
- ✓ SPA receives only the data it needs from the server, except for the first page.

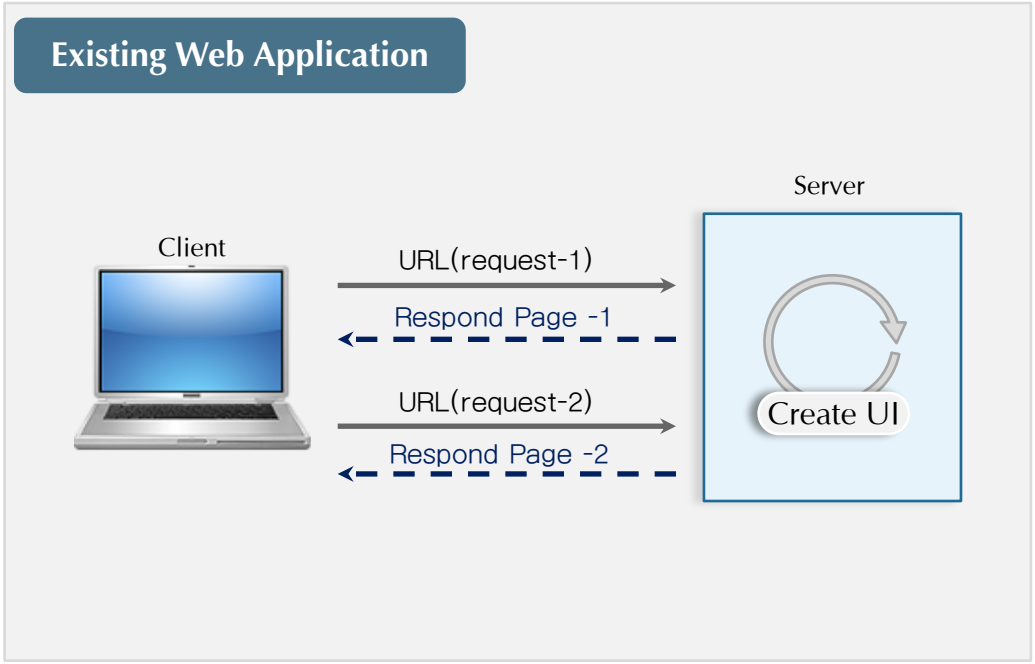


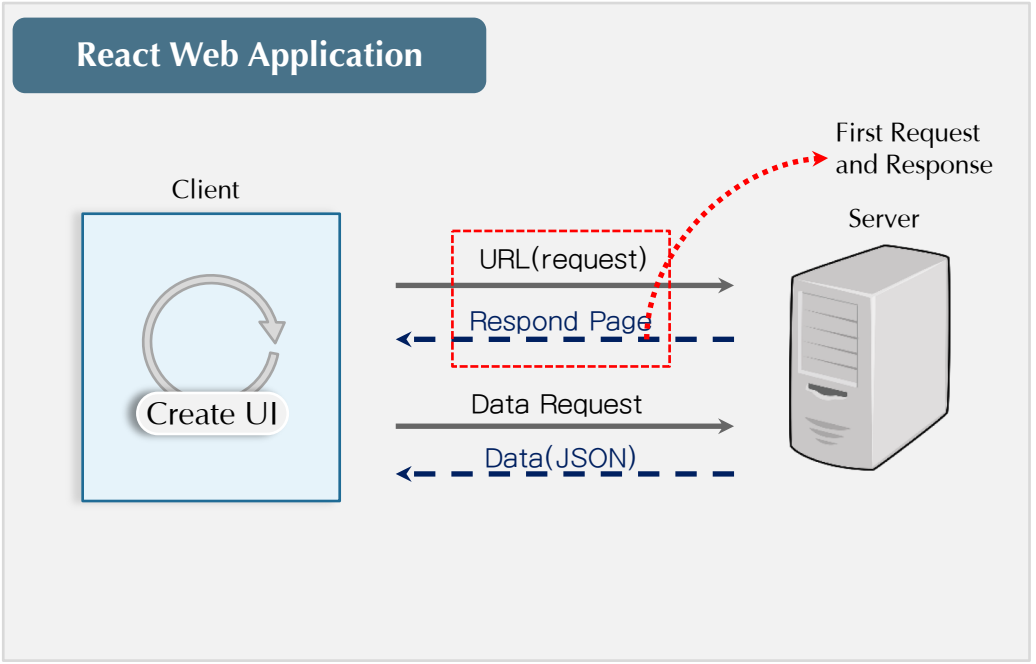
2. React Basic

2.1 How React works
2.2 React Element
2.3 Virtual DOM
2.4 JSX

2.1 How React works

- ✓ Traditional web applications respond by generating a UI (screen composition + data) on the server upon request from the client.
- ✓ React requests necessary data from the server and constructs the user UI using the data received from the server.
- ✓ Only necessary data is received through the server, and the client manages the entire user UI.
- ✓ Representing the user UI on the clients is called Client Side Rendering (CSR).





2.2 React Element

- ✓ React Element is an object that contains information about browser DOM elements.
- ✓ It is an object literal with a description of how the actual browser DOM should be built.
- ✓ It is easy to create because it is a simple object, not an actual DOM.
- ✓ ReactElement cannot change the value it contains, and it is created newly if necessary.

```
let h1 = React.createElement('h1', null, 'Hello React!');
// Create React Element
ReactDOM.render(
  h1,
  document.getElementById('root')
);
// Rendering to the browser DOM
console.log(h1);
// React Element Information Output
```

Hello React!

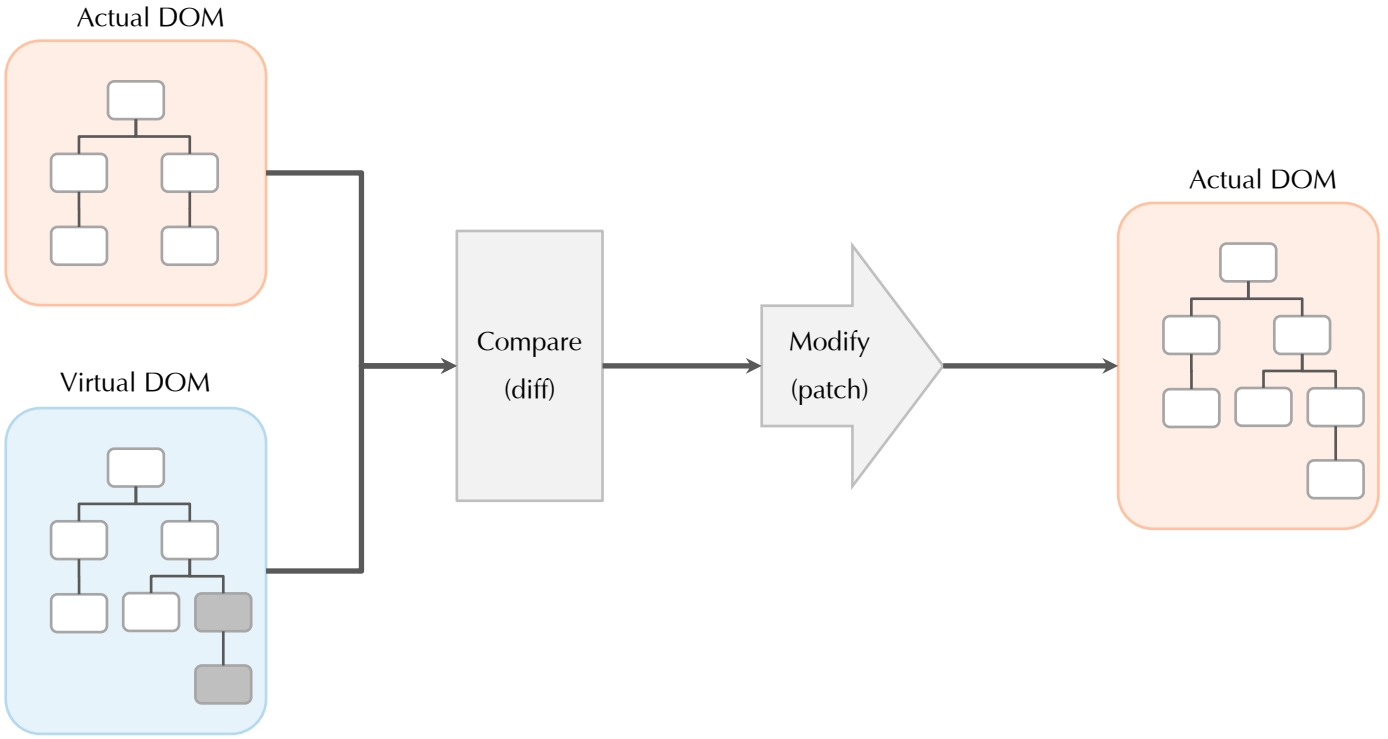
```
1.{$$typeof: Symbol(react.element), type: "h1",
key: null, ref: null, props: {…}, …}
    1. $$typeof: Symbol(react.element)
    2. key: null
    3. props: {children: "Hello React!"}
    4. ref: null
    5. type: "h1"
    6. _owner: null
    7. _store: {validated: false}
    8. _self: null
    9. _source: null
    10. __proto__: Object
```

```
function tick() {
  let element = React.createElement('h1', null, new Date().toLocaleTimeString());
  ReactDOM.render(element, document.getElementById('root'));
}
setInterval(tick, 1000);
```

PM 1:45:54

2.3 Virtual DOM

- ✓ Virtual DOM is a collection of data that reflects the real DOM of the browser and is composed of ReactElements.
- ✓ Changing the screen in a web browser means changing the actual DOM structure, and the existing method is a change method through direct access.
- ✓ React changes elements that need to be changed through a data structure comparable to the actual DOM called Virtual DOM.
- ✓ Process of understanding and changing differences between real DOM and the virtual DOM is compared (diff) and patched.

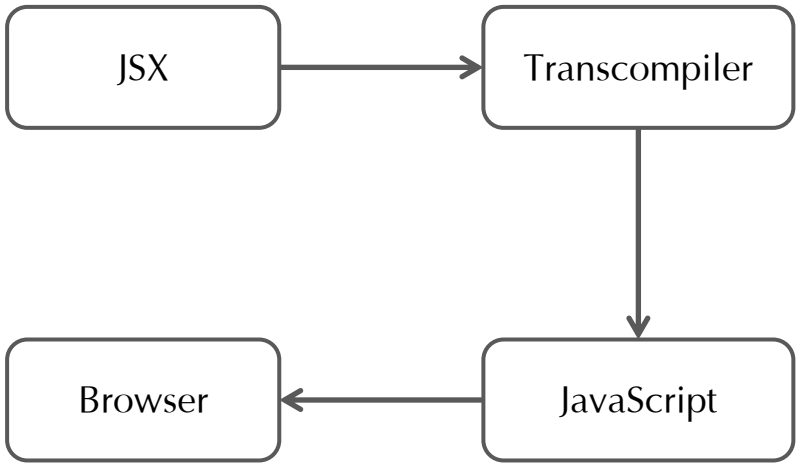


2.4 JSX (1/2)

- ✓ JSX(JavaScript XML) is an extension of JavaScript and was created for grammatical convenience.
- ✓ Code written in JSX is converted into JavaScript code through a transcompiler.
- ✓ Using JSX, you can write XML format code similar to HTML instead of React.createElement() function.
- ✓ When trying to define nested React elements, there is a limit to implementing them in JavaScript code, and in this case, JSX can be used to achieve a concise implementation.

```
//JSX
ReactDOM.render(
  <h1 className='heading'>Hello World</h1>,
  document.getElementById('content')
)
```

```
//JavaScript
ReactDom.render(
  React.createElement(
    'h1',
    { className: 'heading' },
    'Hello World'
  )
)
```



2.4 JSX (2/2)

- ✓ **Because JSX is similar to HTML, you can declare the structure of a component in a familiar way.**
- ✓ **There are several things to consider when defining components using JSX.**
  - Case sensitive: React components and basic HTML elements are case sensitive, and components are written in Pascal Case.
    Example) Component-<List />, <Button/> HTML Element : <h2 />, <input />, <a />
  - Use of {} : Write JavaScript code inside {braces} to reflect JavaScript code in JSX code.
  - Use of reserved words: In the use of standard HTML, reserved words such as class and for are used instead of className and htmlFor in JSX.

```
class DateTimeNow extends React.Component {
    render(){
        let dateTimeNow = new Date().toLocaleDateString();
        return React.createElement(
            'span',
            null,
            'CurrentTime : ${dateTimeNow}.' // ECMAScript
 6
        );
    }
}
```

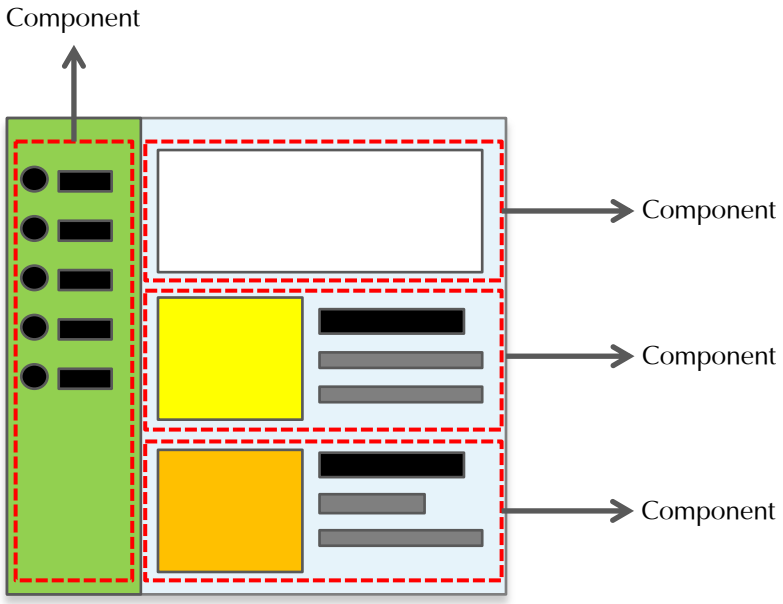
```
class DateTimeNow extends React.Component {
    render(){
        let dateTimeNow = new Date().toLocaleDateString();
        return <span> Current Time {dateTimeNow}.</span>
    }
}
```

3. React Component

3.1 React Component Overview
3.2 Props Overview
3.3 Validation of Props - PropTypes
3.4 Application of Props

3.1 React Component Overview(1/2)

- ✓ Components are the basic unit of React UI construction, and React provides a variety of ways to create components.
- ✓ Encapsulated components can be reused and reconfigured, and user UI is composed of a set of components.
- ✓ React UI consists of a set of components, and components are divided into class-based components and function-based components.
- ✓ Components created on a class base must override the render() method, and the render() method returns a single React Element.




```
let element = React.createElement('h1', null, "Component Test");

class MyComponent extends React.Component {
  render(){
    return React.createElement('div', null, element, element);
  }
}

ReactDOM.render(
  React.createElement(MyComponent, null), document.getElementById('root')
)
```

Component Test
Component Test

3.1 React Component Overview(2/2)

- ✓ Components are a way to group various elements (function, style, markup, etc.) that make up a screen.
- ✓ It becomes the boundary that divides the user UI into parts, and components can contain other components.
- ✓ Components are independent and reusable, so the necessary functions can be independently configured.
- ✓ Components are divided into class-based(stateful) components and function-based(stateless) components.

```
class Customer extends React.Component {
  render() {
    const { id, name, orders } = this.props;

    return (
      <div className="customer">
        <h2>{id}</h2>
        <p>
          <span>Name : {name}</span><br/>
          <span>Order Quantity : {orders.length} </span>
        </p>
      </div>
    )
  }
}
```

```
function App(){
    return <HelloMessage />
}

function HelloMessage(){
    const message = "Hello~~";
    return <h1> {message} </h1>
}
```

```
const App = () => {
    return <HelloMessage />
}

const HelloMessage = () => {
    const message = 'Hello~~';
    return <h1>{message}</h1>
}
```

3.2 Props Overview

- ✓ **Class-based components defined by extending the React.Component class have a props object.**
- ✓ **Props are objects to which data transferred from the parent component is allocated at the time the component is created.**
- ✓ **There are two types of data transmitted from the parent component through props.**
  - Html element attribute : href, title, style, class, etc.
  - Arbitrary data allocated from the parent component: this.props.property name
- ✓ **Data passed through props cannot be changed in the component.**

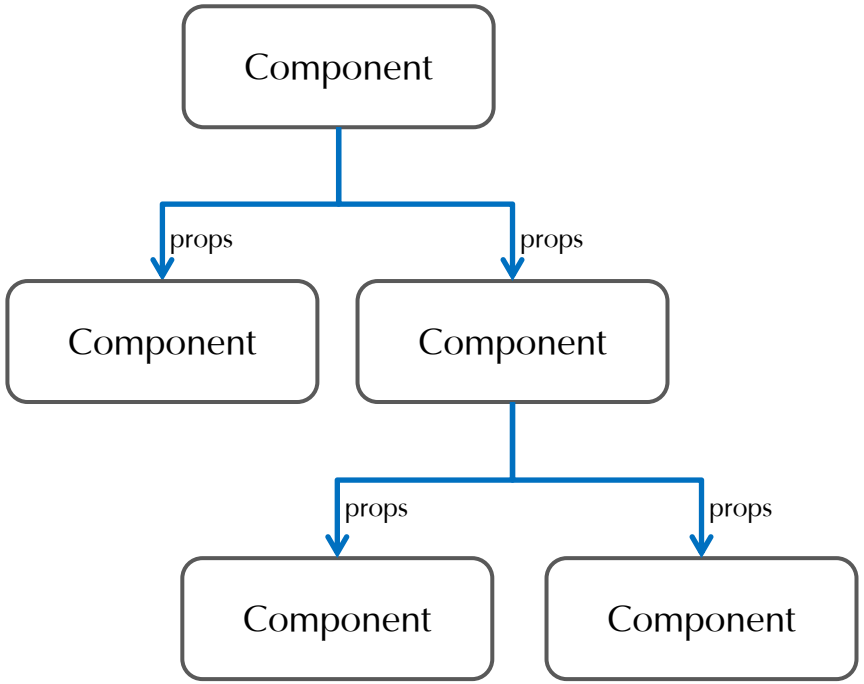
```
<Customer id='kim@namoosori.io' name='Kim' orders={['A0001', 'B0002', 'C0003']}/>
```

```
class Customer extends React.Component {
  render() {
    const { id, name, orders } = this.props;
    return (
    <div className="customer">
        <h2>{id}</h2>
        <p>
          <span>Name : {name}</span><br/>
          <span>Order Quantity : {orders.length}</span>
        </p>
      </div>
    )
  }
}
```

kim@namoosori.io
Name : Kim
Order Quantity : 3

3.3 Validation of Props – PropTypes(1/2)

- ✓ JavaScript is a language with loose type validation, which means that the restrictions on the data types assigned to variables are loosely.
- ✓ The component that receives the data is responsible for the verification of the data type passed to the Props.
- ✓ React.PropTypes performs validation on the types of props passed from the parent component.
- ✓ Use the prop-types library to validate the data assigned to the component's props.



Type	Verification
Array	React.PropTypes.array
Boolean	React.PropTypes.bool
Function	React.PropTypes.func
Number	React.PropTypes.number
Object	React.PropTypes.object
String	React.PropTypes.string

3.3 Validation of Props – PropTypes(2/2)

- ✓ Validates props received from the parent component through PropTypes.
- ✓ PropTypes are type validation for props, and if the types are different, a warning is displayed through the developer console.
- ✓ PropTypes are not required to be applied, but they are used for debugging and preventing bugs.
- ✓ It is also possible to set default values for props using DefaultProps.

```
<Customer id='kim@namoosori.io' name='Kim' orders='A0001, B0002, C0003'/>
```

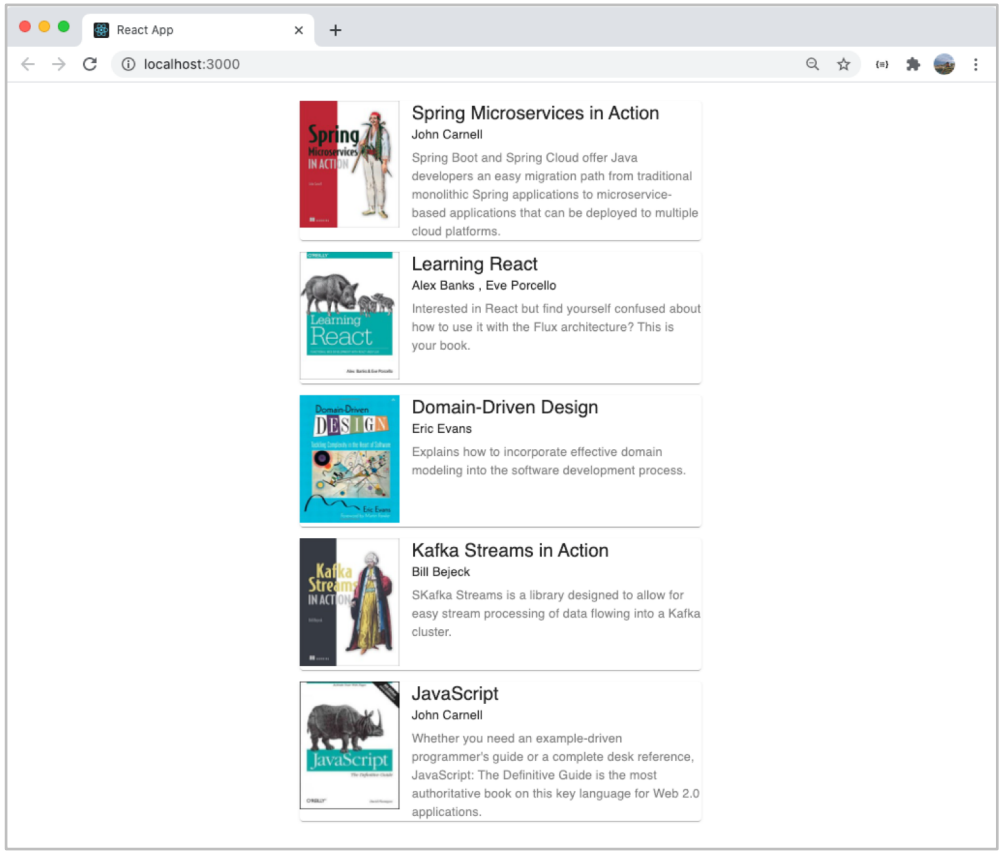
```
class Customer extends Component {
  static propTypes = {
    id : PropTypes.string.isRequired,
    name : PropTypes.string.isRequired,
    orders : PropTypes.array.isRequired
  }

  render() {
      const { id, name, orders } = this.props;
      return (
      <div className="customer">
        <h2>{id}</h2>
        <p>
          <span>Name : {name}</span><br/>
          <span>Order Quantity : {orders.length}</span>
        </p>
      </div>
    )
  }
}
```

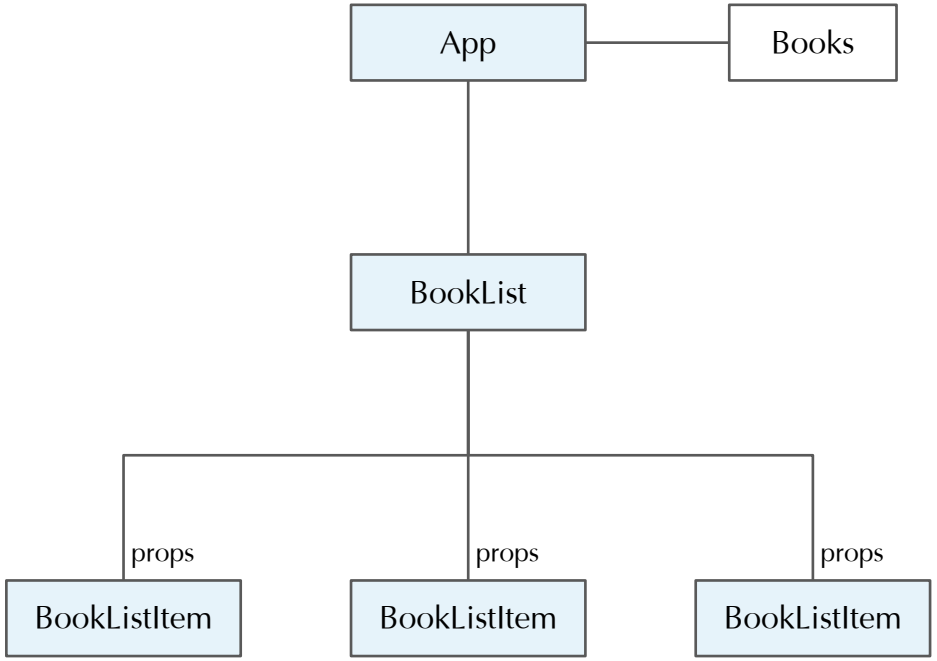
kim@namoosori.io
Name : Kim
Order Quantity : 19

```
▸Warning: Failed prop type: Invalid prop `orders` of type `string` supplied to `Customer`, expected `array`.
    in Customer (at App.js:14)
    in App (at src/index.js:7)
```

3.4 Application of Props

- ✓ The following is an example of passing a value to a child component in the containing hierarchy through props.
- ✓ App component refers to the value to be displayed in the list through Books.js.
- ✓ The BookListItem component receives data to be expressed through props from BookList.
- ✓ Material-UI React is used to express components.



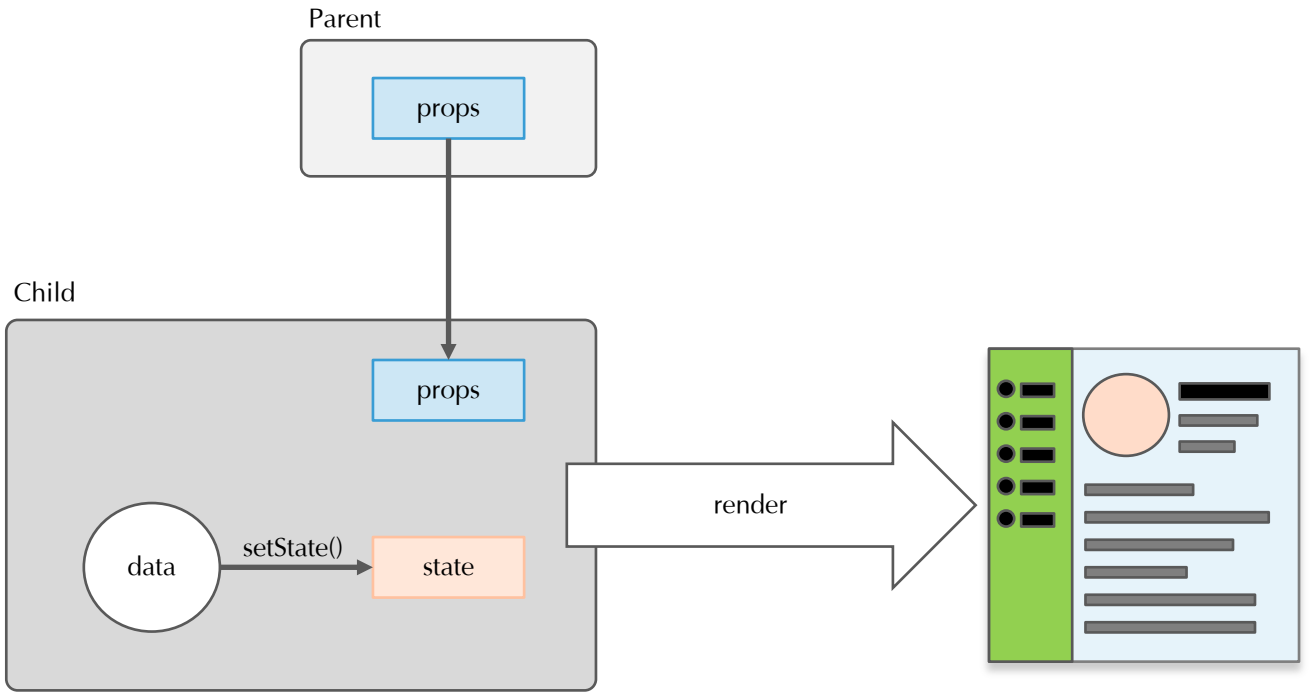


4. State

4.1 State Overview
4.2 State initialization and Change
4.3 Application of State

4.1 State Overview

- ✓ Props are immutable data from the component's point of view.
- ✓ The object used to manage mutable data in a component is state.
- ✓ State exists only in class-based components that inherit the React.Component class.
- ✓ Initialization of the state value is implemented in the declaration or constructor of the object field, and the change of the state value uses the setState() method.



4.2 State Initialization and Change

- ✓ Constructors are special methods that are called once when a class is instantiated.
- ✓ Initialization by assigning a value to the state object is performed in the field declaration or constructor of the component object.
- ✓ When defining a constructor for state initialization, you pass props to the parent class by calling the super constructor. This is the code to use this reference.
- ✓ To change the value of the state object, you must call the setState() method to change it.

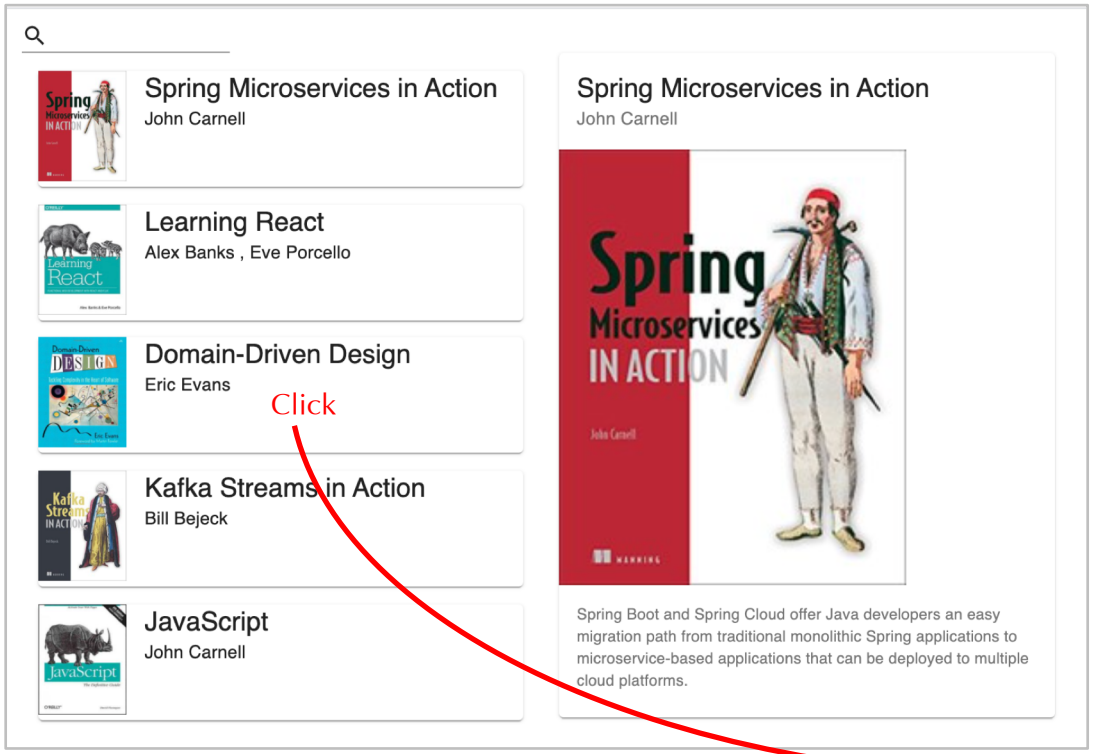
```
class StateApp extends Component {
  constructor(props) {
    super(props);
    this.state = {
      books: Books,
      selectedBook: Books[0]
    }
  }
  searchByTitle(title) {
    let updateList = Books;
    updateList = updateList.filter(book => {
      return book.title.toLowerCase().search(title.toLowerCase()) !== -1;
    });
    this.setState({
      books: updateList
    })
  }
```

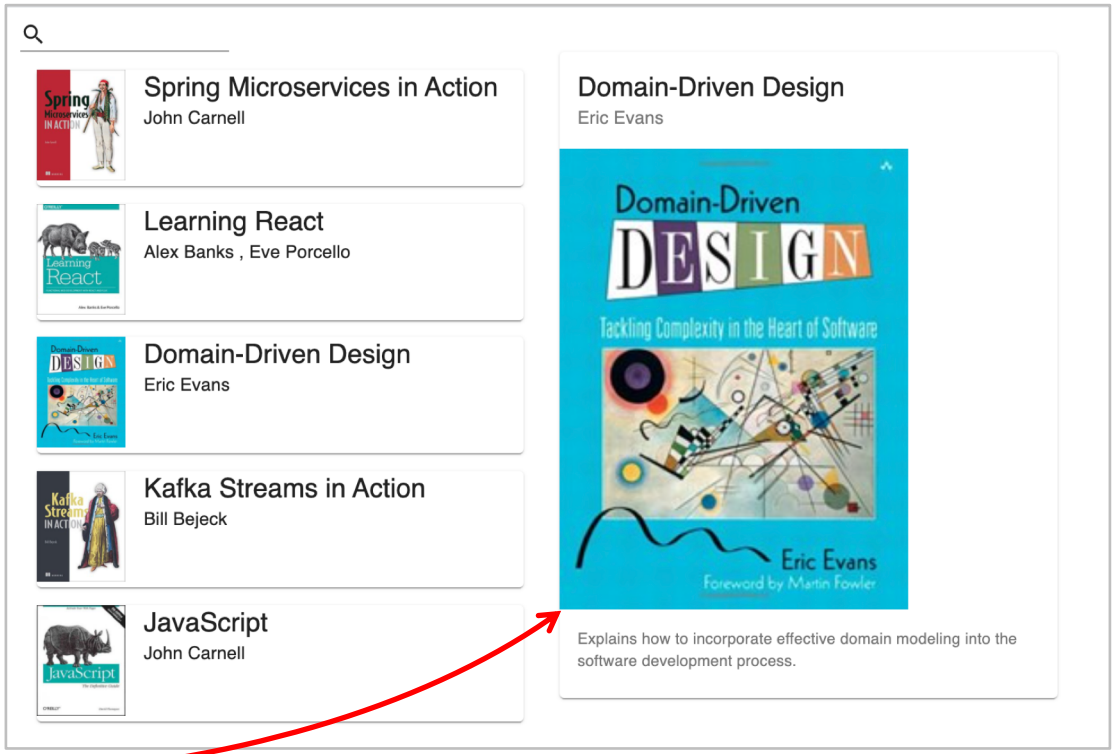
state initialization

state change

4.3 Application of State (1/2)

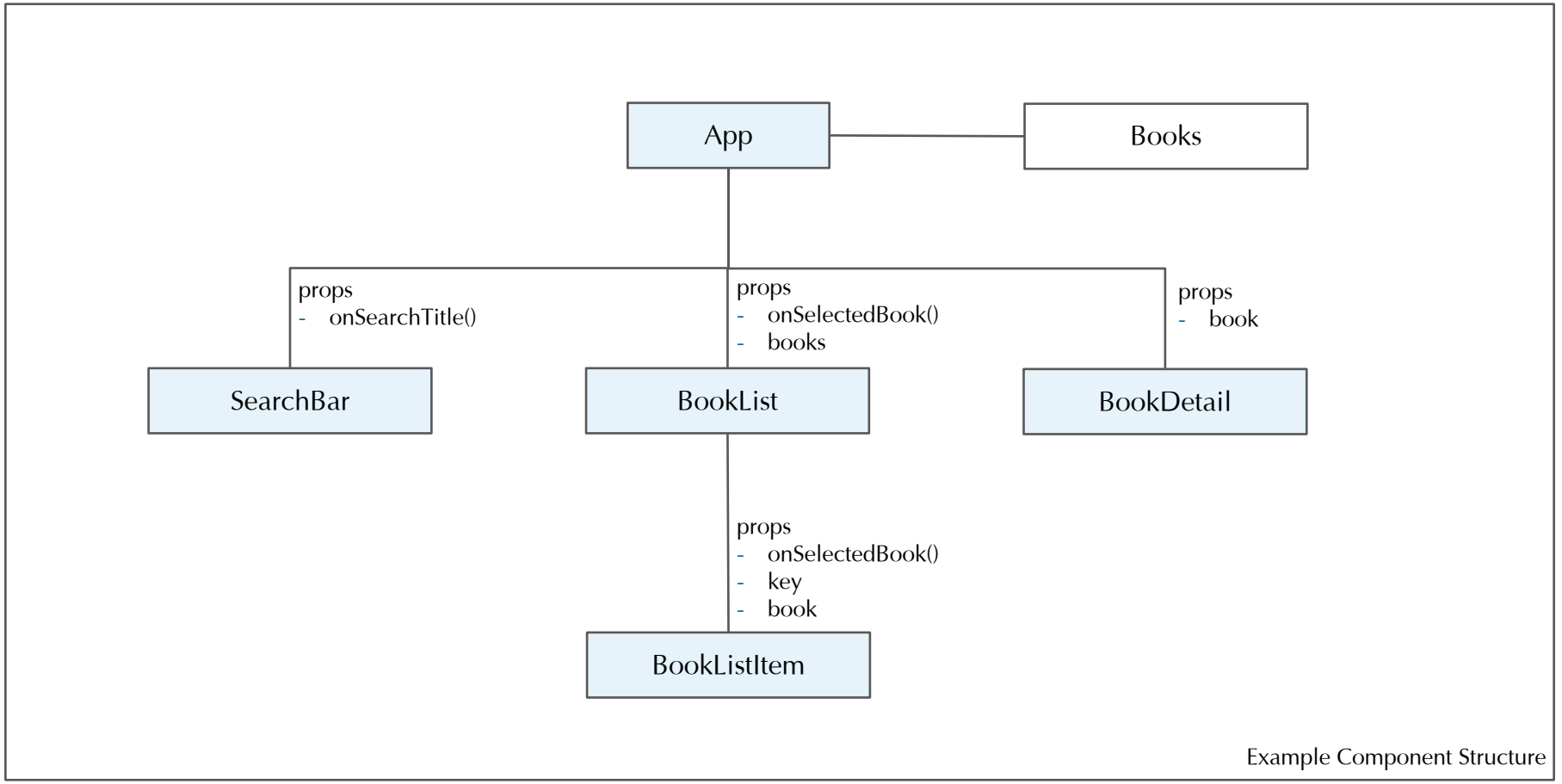
- ✓ The state object manages the data of the component and updates the screen by changing this data.
- ✓ Every class-based component that extends the React.Component class has a state object.
- ✓ It increases the complexity of the program for the various components that make up one UI to each have separate state and manage it.
- ✓ The root component has the entire state, and the sub-components receive immutable data through props.





4.3 Application of State (2/2)

- ✓ As state is a JavaScript object, it can contain various kinds of data.
- ✓ App component has books data and is managed through state object.
- ✓ SearchBar, BookList, BookDetail component receive book information as props object data from App component.



Example Component Structure

End of Document

✓토론

감사합니다...

- 넥스트리(주)
- www.nextree.io