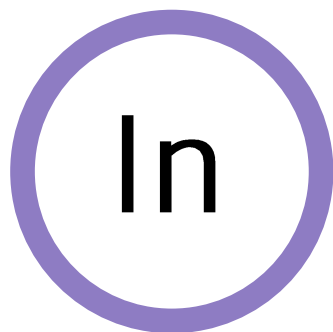


Dapp FrontEnd React

Web3



React 기초



01 React

React

UI를 개발하는데 도움을 주는 JS라이브러리.
웹개발에서 View 계층이다.

01 React

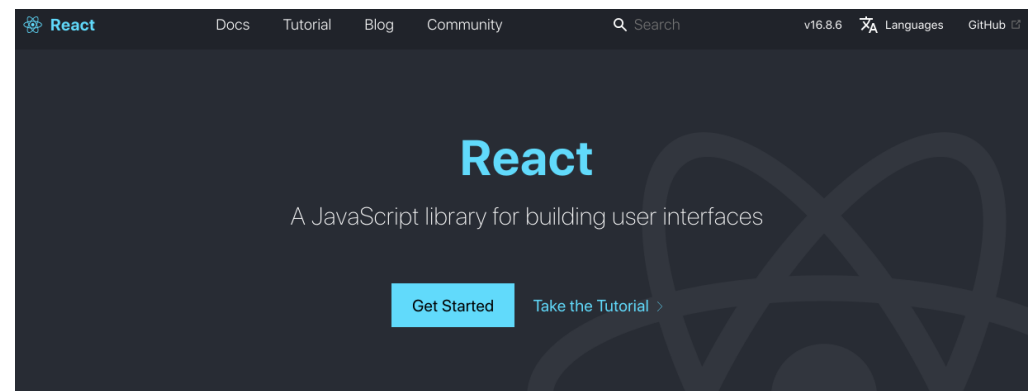
React

UI를 개발하는데 도움을 주는 **JS라이브러리**.
웹개발에서 View 계층이다.

A JavaScript library for building **user interfaces**

React.js: <https://reactjs.org/>

한국 React.js: <https://ko.reactjs.org/>



Declarative

React makes it painless to create interactive UIs. Design simple views for each state in your application, and React will efficiently update and render just the right components when your data changes.

Component-Based

Build encapsulated components that manage their own state, then compose them to make complex UIs.

Since component logic is written in JavaScript

Learn Once, Write Anywhere

We don't make assumptions about the rest of your technology stack, so you can develop new features in React without rewriting existing code.

React can also render on the server using Node.js

01 React

React

Declarative

선언형

Interactive한 UI를 만드는데 도움을 준다.

→ data가 변경되면 효율적으로 컴포넌트를 업데이트 한다.

Component-Based

컴포넌트 기반

스스로 상태를 관리하는 캡슐화된 컴포넌트를 조합해 복잡한 UI를 만든다.

컴포넌트 로직 = JS코드

→ 상태관리 = JS코드

Learn Once,
Write Anywhere

01 React

컴포넌트란 무엇인가?

컴포넌트의 기본 모양

```
class HelloMessage extends React.Component {
  render() {
    return (
      <div>
        Hello {this.props.name}
      </div>
    );
  }
}

ReactDOM.render(
  <HelloMessage name="Taylor" />,
  mountNode
);
```

- 컴포넌트에서 무엇이 입력이고 무엇이 출력일까?

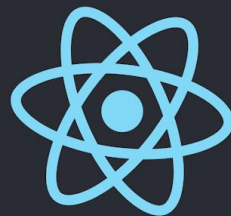
개념적으로 컴포넌트는 자바스크립트 함수와 같다.

Conceptually, components are like JavaScript functions. They accept arbitrary inputs (**called “props”**) and return React elements describing what should appear on the screen.

01 React

React App 쉽게 시작하기

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



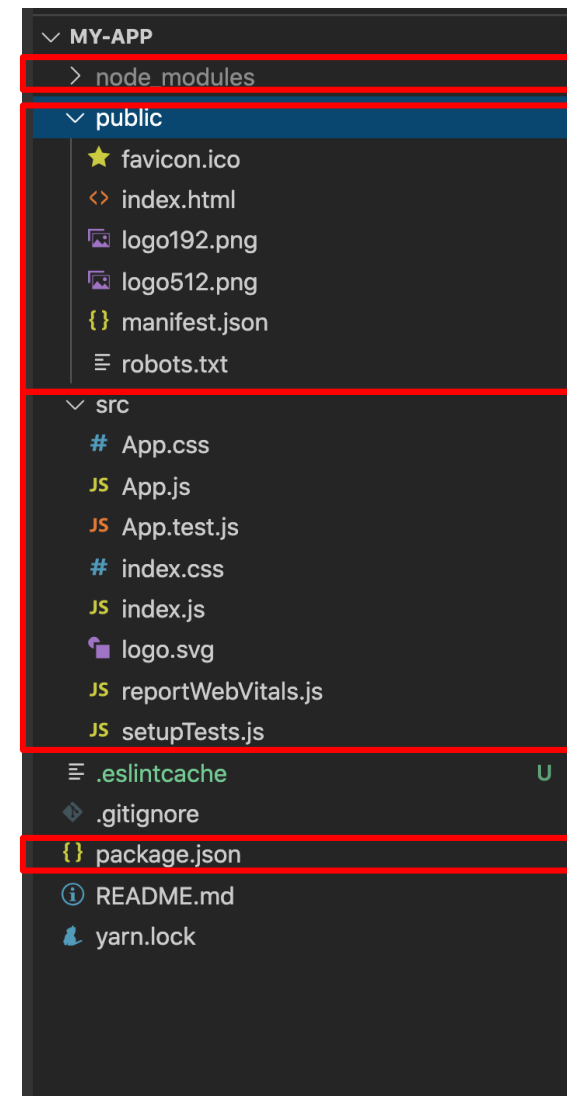
Edit `src/App.js` and save to reload.

[Learn React](#)

01 React

폴더 구조 살펴보기

- node_modules: 의존성 패키지
- public/*: 공유하는 파일. 기본적인 static files
- src/*: 실제 code
- package.json: 프로젝트(application)의 정체성(?)



01 React

src/* : 실제 코드!

- index.js
- index.css
- App.js
- App.css

```
You, 8 minutes ago | 1 author (You)
1 import React from 'react';           You, 8 minutes ago • Initialize project using Create React App
2 import ReactDOM from 'react-dom';
3 import './index.css';
4 import App from './App';
5 import reportWebVitals from './reportWebVitals';
6
7 ReactDOM.render(
8   <React.StrictMode>
9     <App />
10  </React.StrictMode>,
11  document.getElementById('root')
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();
18
```

```
You, 9 minutes ago | 1 author (You)
1 body {           You, 9 minutes ago • Initialize project using Create React App
2   margin: 0;
3   font-family: -apple-system, BlinkMacSystemFont, 'Segoe UI', 'Roboto', 'Oxygen',
4     'Ubuntu', 'Cantarell', 'Fira Sans', 'Droid Sans', 'Helvetica Neue',
5     sans-serif;
6   -webkit-font-smoothing: antialiased;
7   -moz-osx-font-smoothing: grayscale;
8 }
9
10 code {
11   font-family: source-code-pro, Menlo, Monaco, Consolas, 'Courier New',
12     monospace;
13 }
14
```

01 React

src/* : 실제 코드!

- index.js
- index.css
- **App.js**
- **App.css**

```
1  import logo from './logo.svg';      You, 9 minutes ago • Init
2  import './App.css';
3
4  function App() {
5    return (
6      <div className="App">
7        <header className="App-header">
8          <img src={logo} className="App-logo" alt="logo" />
9          <p>
10             Edit <code>src/App.js</code> and save to reload.
11          </p>
12          <a
13             className="App-link"
14             href="https://reactjs.org"
15             target="_blank"
16             rel="noopener noreferrer"
17           >
18             Learn React
19          </a>
20        </header>
21      </div>
22    );
23  }
24
25  export default App;
26
```

01 React

src/* : 실제 코드!

- index.js
- index.css
- **App.js**
- **App.css**

```
1  import logo from './logo.svg';
2  import './App.css';
3
4  function App() {
5    return (
6      <div className="App">
7        <header className="App-header">
8          <img src={logo} className="App-logo" alt="logo" />
9          <p>
10             Edit <code>src/App.js</code> and save to reload.
11          </p>
12          <a
13             className="App-link"
14             href="https://reactjs.org"
15             target="_blank"
16             rel="noopener noreferrer"
17           >
18             Learn React
19          </a>
20        </header>
21      </div>
22    );
23  }
24
25  export default App;
```

You, 9 minutes ago • Init

import

JSX

Export

01 React

JSX(JavaScript XML)

JSX

```
export default class HelloWorldApp extends Component {  
  render(){  
    return(  
      <View style={{ flex: 1, justifyContent: "center", alignContent: "center" }}>  
        <Text>Hello, World</Text>  
      </View>  
    )  
  }  
}
```

Element 만드는 것을 직관적으로 이해하도록 도와주는 문법
→ 컴파일 되어

React.createElement(component, props, ...children)가 호출

01 React

JSX 사용 안할 때!

잠깐Q) JSX를 쓸 때 vs 안쓸 때

```
class HelloMessage extends React.Component {
  render() {
    return React.createElement(
      "div",
      null,
      "Hello ",
      this.props.name
    );
  }
}

ReactDOM.render(React.createElement(HelloMessage, { name: "Bonnie" }), mountNode);
```

```
class HelloMessage extends Component {
  render() {
    // Instead of calling createElement, we return markup
    return <div>Hello {this.props.name}</div>;
  }
}

// 더 이상 createElement를 여기서 호출할 필요가 없다
ReactDOM.render(<HelloMessage name="Bonnie" />, mountNode);
```

01 React

아래처럼 본인 이름이 나오게 코드를 수정해보세요.

•배우지 않아도 알 수 있어요!

아래처럼 본인 이름이 나오게 코드를 수정해보세요.
•배우지 않아도 알 수 있어요!



Hello 신윤수

[Learn React](#)

02 첫 Component 만들기

컴포넌트: 화면을 나타내는 기본 단위

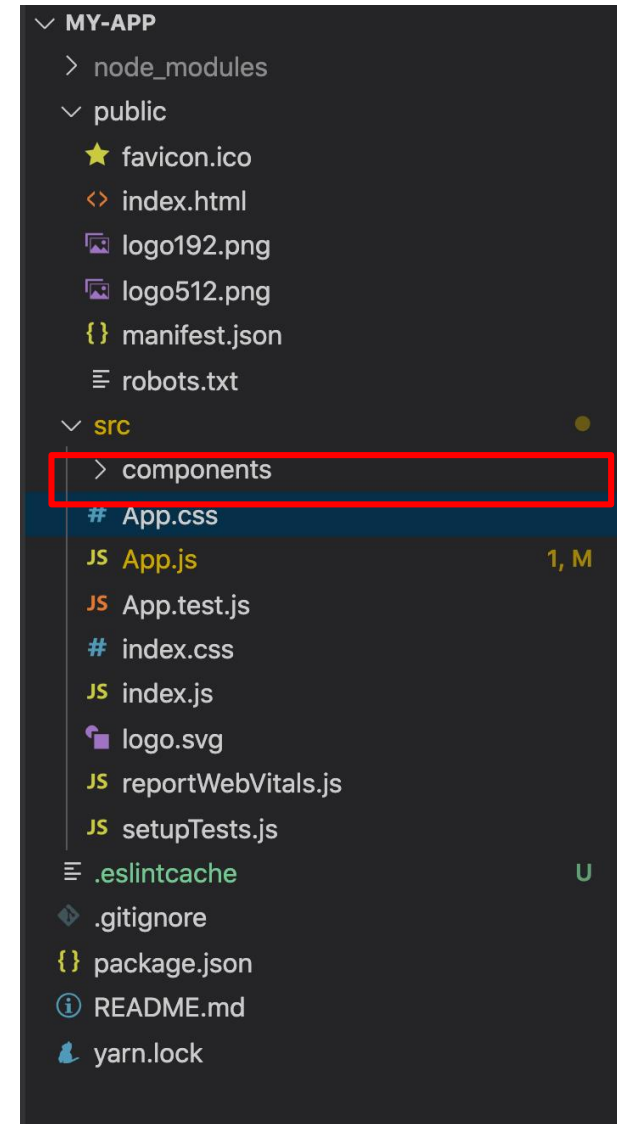
React: 복잡한 UI를 여러개의 컴포넌트로 구성하여 표현한다!

우린 여러가지 component를 조합할 예정이기에 새로운 컴포넌트를 만들어 줄 것.

우선 **root project**에서 **src**라는 디렉토리를 만들고 그 안에서 **components**라는 디렉토리 구성

앞으로 우리가 만드는 컴포넌트는 **재사용**을 위해서 **src/components** 폴더안에 저장할 것 입니다.

HelloWorld라는 샘플 컴포넌트를 하나 만들겠습니다.



02 첫 Component 만들기

src/components/HelloWorld.js

```
1  import React from 'react'
2
3  export default function HelloWorld() {
4    return (
5      <div>
6        <h1>Hello, World!</h1>
7        <p>This is My first React Application</p>
8      </div>
9    )
10 }
```

반드시 필요! React에게 이게
컴포넌트라고 인식시켜줌

Functional Component

복잡한 View를 여러개의
컴포넌트로 구성하여 표현!

주의사항:
반드시 **import React**가 존재해야
함.

02 첫 Component 만들기

index.js를 바꿔서 렌더링을 다르게 해보겠습니다.

```
1  import React from 'react';
2  import ReactDOM from 'react-dom';
3  import './index.css';
4  import App from './App';
5  import reportWebVitals from './reportWebVitals';
6
7  import HelloWorld from './components/HelloWorld'
8
9  ReactDOM.render(
10    <React.StrictMode>
11    <HelloWorld />
12    </React.StrictMode>,
13    document.getElementById('root')
14  );
15
16  // If you want to start measuring performance in your app, pass a function
17  // to log results (for example: reportWebVitals(console.log))
18  // or send to an analytics endpoint. Learn more: https://bit.ly/CRA-vitals
19  reportWebVitals();
20
```

02 첫 Component 만들기

연습문제(2)

아래처럼 만들어보세요.

• 우선 React와 HTML/CSS랑 친해져봅시다.



힌트 App.js를 이용하세요.

App.css는 이미 가운데 정렬이 되어 있습니다.

03컴포넌트의 종류

컴포넌트의 종류

함수형 컴포넌트

```
export default function App() {  
  return (  
    <View style={styles.container}>  
      <Text>Hello React Native!</Text>  
    </View>  
  );  
}
```

- props 전달하고, hook 사용 가능

함수형 컴포넌트

```
export default class App extends Component{  
  render(){  
    return(  
      <View style={styles.container}>  
        <Text>Hello React Native!</Text>  
      </View>  
    )  
  }  
}
```

- props 전달, LifeCycle API사용

모르는 용어가 너무 많지요?
Props, Hook, State, LifeCycle API.. 차차 배울 예정입니다.

03컴포넌트의 종류

Props란 (Properties)

Props

부모 컴포넌트로 전달받는 **파라미터!**
컴포넌트를 **재사용**이 가능하도록 하는 핵심!

컴포넌트가 생성 될 때 **부모로부터 전달받음**
컴포넌트에 **고정된 상태**로 남아있음. (변하지 않는 값)

03 컴포넌트의 종류

Props란 (CaptionImage.js 예제)

```
1  import React from 'react'
2
3  export default function CaptionImage(props) {
4    return (
5      <div>
6        <img src={props.imgUrl} alt="" />
7        <p>{props.caption}</p>
8      </div>
9    )
10 }
11
```

CaptionImage 컴포넌트에서 imgUrl과 caption을 받아서 렌더링한다.

03컴포넌트의 종류

App.js

```
You, seconds ago | 1 author (You)
1  import logo from './logo.svg';
2  import './App.css';
3
4  import CaptionImage from './components/CaptionImage'
5
6  function App() {
7    return (
8      <div className="App">
9        <CaptionImage imgUrl='https://upload.wikimedia.org/wikipedia/commons/d/de/Bananavarieties.jpg'
10         caption='이건 바나나예요.' />
11      </div>
12    );
13  }
14
15  export default App;
```

부모컴포넌트로부터 imgUrl, caption을 받아서 그린다.

⇒ 재사용이 가능하다!

03컴포넌트의 종류

연습문제3 props

트럭사진을 비치하고(어느 사진이든 괜찮습니다.)

•이건트럭입니다. 라고 변경하세요.



이건 트럭이에요.

Auto Tribune

03 컴포넌트의 종류

Props 핸들링 하기

Props는 부모 컴포넌트로부터 전달받는 것이라고 했습니다.
함수의 인자로 이해하면 편할 것입니다.

App.js에는 똑같이 추가하여 CaptionImage컴포넌트만 불러주면됩니다.
CaptionImage처럼 사용하면 같은 패턴의 UI를 구성할 때 **재사용이 용이**합니다.

03컴포넌트의 종류

state 예제

State

- State는 Props와 다르게 자체적으로 가지고 있는 것.
- 컴포넌트 내에서 저장되고 유저의 이벤트나 시간에 따라 변화시킬 수 있는 값
- 초기화가 반드시 필요. `this.state={ }` //클래스형 컴포넌트!
- Hook을 사용할 경우, `useState` 사용 // 함수형 컴포넌트.

보통 state는 서버에서 받아와서 사용하거나 - 회원정보 등
동적인 ui 핸들링시에 사용 - 이벤트 처리

state관리가 React의 핵심!

03 컴포넌트의 종류

state 예제

보통 state는 서버에서 받아와서(회원정보) 사용하거나 동적인 ui 핸들링시에 사용

```
export default class Blink extends Component {  
  constructor(props) {  
    super(props);  
  
    // 아래가 상태입니다.  
    this.state = {showText: true};  
  
    // 2초마다 toggle하는 함수  
    setInterval(()=>{  
      // state 변경  
      this.setState({showText: !this.state.showText})  
    }, 2000)  
  }  
  
  render(){  
    // showText가 True일경우에만 호출  
    let display = this.state.showText ? this.props.text : ' ';  
    return <Text>{display}</Text>  
  }  
}
```

Blink Component

<src/components/Blink.js>

Function이 아니라 Class임에 주목해주세요!
클래스형 컴포넌트입니다.

03컴포넌트의 종류

state 예제

```
export default class Blink extends Component {
  constructor(props){
    super(props);

    // 아래가 상태입니다.
    this.state = {showText: true};

    // 2초마다 toggle하는 함수
    setInterval(()=>{
      // state 변경
      this.setState({showText: !this.state.showText})
    }, 2000)
  }

  render(){
    // showText가 True일경우에만 호출
    let display = this.state.showText ? this.props.text : ' ';
    return <Text>{display}</Text>
  }
}
```

읽을 수 있지요?

1. 생성자가 호출되면 반드시 `super(props)`를 호출해 주어야 합니다.

함수형 컴포넌트에서 인자로 `props` 받은 것 기억나지요?
마찬가지로 클래스형 컴포넌트에서도 `props`를 전달받습니다.
이를 React Component의 클래스로 전달해 초기화 하는 것입니다.

2. `this.state`는 객체입니다. 보통 **constructor** 혹은 나중에 배우게 될 **componentDidMount** 라는 함수에서 정의합니다.

3. `setInterval` 함수를 통해 2초에 한번 상태를 반대로 변경하기로 했습니다.

03 컴포넌트의 종류

props vs state

Props

부모 컴포넌트로 전달받는 파라미터!

컴포넌트를 재사용이 가능하도록 하는 핵심!

- 컴포넌트가 생성 될 때 부모로부터 받고 컴포넌트에 고정된 상태로 남아있음.
(변하지 않는 값)

State

컴포넌트 내에서 저장되고 유저의 이벤트나 시간에 따라 변화시킬 수 있는 값.

- 초기화가 반드시 필요. `this.state={ }`
- **State 변경 판단은 주소값 비교**
- State는 Props와 다르게 자체적으로 가지고 있는 것.
- 컴포넌트 내에서 저장되고 유저의 이벤트나 시간에 따라 변화시킬 수 있는 값

03 컴포넌트의 종류

Component를 만드는 두가지 방법

함수형 컴포넌트

```
export default function App() {  
  return (  
    <View style={styles.container}>  
      <Text>Hello React Native!</Text>  
    </View>  
  );  
}
```

어떨 때 함수형 컴포넌트를 쓰고 어떨 때 클래스형 컴포넌트를 사용할까요.

앞서 이야기 하듯 **state** 혹은 **constructor**나 **componentDidMount** 등 **LifeCycle API**를 이용해야 할 경우 사용합니다.

클래스형 컴포넌트

```
export default class App extends Component{  
  render(){  
    return(  
      <View style={styles.container}>  
        <Text>Hello React Native!</Text>  
      </View>  
    )  
  }  
}
```

잘 모르겠으면 클래스형 써라!

최근에는 **Hook**이 나와서 함수형도 많이 쓰입니다,

클래스형 컴포넌트의 **render** 함수 = 함수형 컴포넌트의 기본

03 컴포넌트의 종류

Hook이란?

Hooks are a new addition in React 16.8. They let you use state and other React features without writing a class.

Hook을 사용하면, 모든 컴포넌트까지(함수형) 독립적으로
재사용가능.

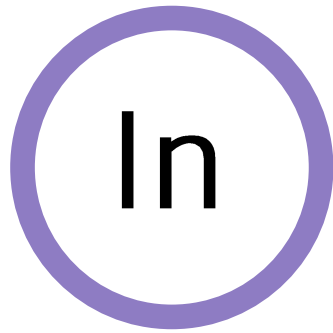
Hook의 종류(기본)

- `useState` - 상태를 사용함.
- `useEffect` - 상태가 변화할때 실행됨.

Hook의 종류(advanced)

- `useCallback` - 함수가 변화할때
재정의
- `useMemo` - 상태가 변화할 때 계산.

Hook은 우리가 만들 수도 있습니다!
과정에서 혼용하여 사용하겠습니다.



React Router

01 React-Router

React Router Dom React Component끼리 페이지 이동

흔히 하는 웹사이트 처럼 페이지 링크가 필요하지요.
외부 라이브러리를 설치할 겁니다.

설치방법: `npm install react-router-dom`

REACT TRAINING / REACT ROUTER

GITHUB NPM GET TRAINING



LEARN ONCE, ROUTE ANYWHERE

REACT ROUTER

Components are the heart of React's powerful, declarative programming model. React Router is a collection of **navigational components** that compose declaratively with your application. Whether you want to have **bookmarkable URLs** for your web app or a composable way to navigate in **React Native**, React Router works wherever React is rendering--so take your pick!

WEB

NATIVE

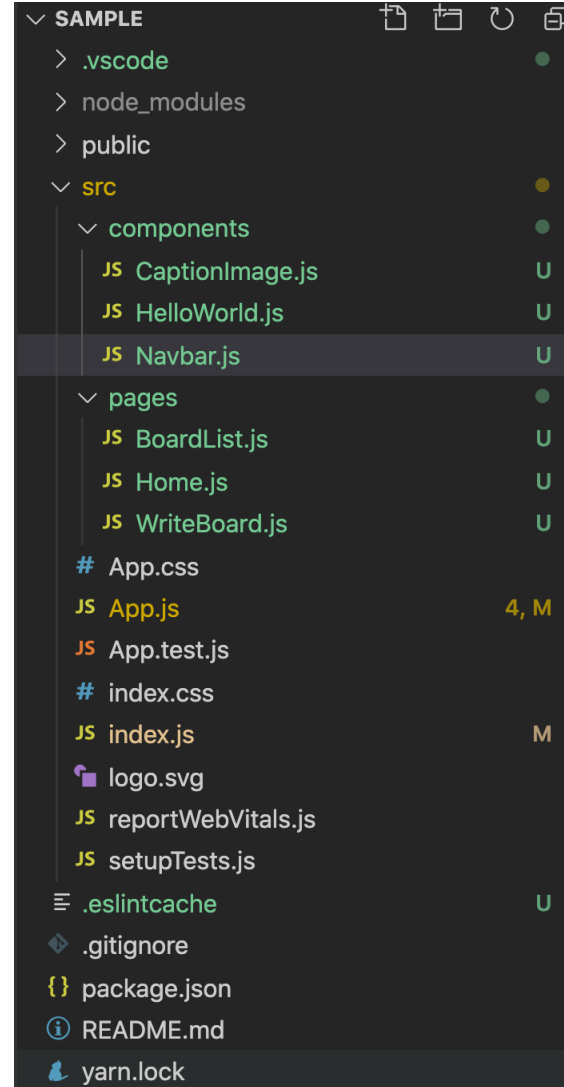
<https://reactrouter.com/>

01 React-Router

pages

구조적 일관성을 위해 pages라는 directory를 만들고, 페이지를 이동하는 단위당 하나의 컴포넌트를 해당 directory에 작성할 겁니다.

⇒ 다음과 같이 3개의 page와 components/Navbar.js를 만들어주세요.



01 React-Router

pages/Home.js

```
1  import React from 'react'
2
3  export default function Home(props) {
4    return (
5      <div>
6        This is Home
7      </div>
8    )
9  }
```


URL의 구조를 잡기 위해서 임시 문자열(This is Home)을 내용으로 넣었습니다.

01 React-Router

components/Navbar.js

```
1  import React from 'react'
2
3  export default function Navbar() {
4    return (
5      <nav style={{width:'100%', backgroundColor: 'red', height: "50px"}}>
6        <div>Logo</div>
7      </nav>
8    )
9  }
```

Page Link는 나중에 넣고! 일단 레이아웃 분리 차원에서 Navbar를 정의합니다.



01 React-Router

React-Router-Dom 사용하기

React-Router-Dom의 주요 Component

- BrowserRouter
- Switch
- Route
- Link

01 React-Router

App.js

오른쪽처럼 작성하면 됩니다.

BrowserRouter가 Switch의 부모컴포넌트이고,
Route는 각 URL에 따라 보여줄 컴포넌트들을
자식 컴포넌트로 가지고 있습니다.

그럼 해당 URL을 직접 쳐서 이동시켜볼까요

<http://localhost:3000/>

<http://localhost:3000/board>

<http://localhost:3000/board/write>

```
1  import './App.css';
2  import {
3    BrowserRouter as Router,
4    Switch,
5    Route,
6  } from "react-router-dom";
7  import Home from './pages/Home';
8  import WriteBoard from './pages/WriteBoard';
9  import BoardList from './pages/BoardList';
10 import Navbar from './components/Navbar'
11
12 function App() {
13   return (
14     <Router>
15       <Navbar/>
16       <Switch>
17         <Route path="/" exact>
18           <Home />
19         </Route>
20         <Route path="/board">
21           <BoardList/>
22         </Route>
23         <Route path="/board/write">
24           <WriteBoard/>
25         </Route>
26       </Switch>
27     </Router>
28   );
29 }
30
31 export default App;
```

01 React-Router

그럼 해당 URL을 직접 쳐서 이동시켜볼까요

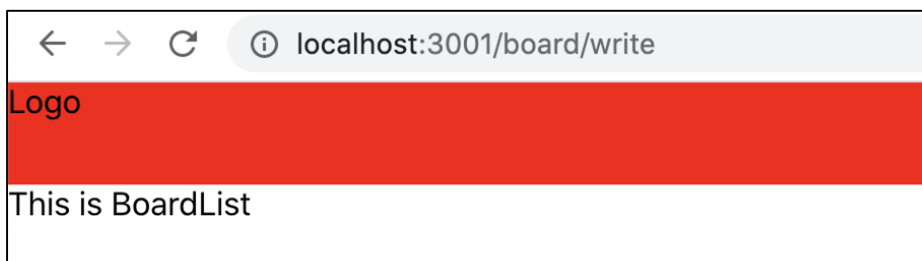
<http://localhost:3000/>

<http://localhost:3000/board>

<http://localhost:3000/board/write>

예상대로 잘 작동하나요?

아마 /board/write가 안 보일거예요.



그 이유는 route의 배치 순서와 연관이 있습니다.
route의 path check는 기본적으로 시작지점만
같으면 매칭됩니다.

앞에서 이미 매칭되는 (path로 시작하는) Route를
만났기 때문에 뒤에는 실행되지 않는거지요.

이를 방지하기 위해선 exact라는 props를 true로
전달해 주거나 순서를 바꿔주어야 합니다.

```
1 import './App.css';
2 import {
3   BrowserRouter as Router,
4   Switch,
5   Route,
6 } from 'react-router-dom';
7 import Home from './pages/Home';
8 import WriteBoard from './pages/WriteBoard';
9 import BoardList from './pages/BoardList';
10 import Navbar from './components/Navbar';
11
12 function App() {
13   return (
14     <Router>
15       <Navbar/>
16       <Switch>
17         <Route path="/" exact>
18           <Home />
19         </Route>
20         <Route path="/board">
21           <BoardList/>
22         </Route>
23         <Route path="/board/write">
24           <WriteBoard/>
25         </Route>
26       </Switch>
27     </Router>
28   );
29 }
30
31 export default App;
```

App.js

01 React-Router

App.js

```
function App() {  
  return (  
    <Router>  
      <Navbar/>  
      <Switch>  
        <Route path="/" exact>  
          <Home />  
        </Route>  
        <Route path="/board/write">  
          <WriteBoard/>  
        </Route>  
        <Route path="/board">  
          <BoardList/>  
        </Route>  
      </Switch>  
    </Router>  
  );  
}
```

순서 변경

```
function App() {  
  return (  
    <Router>  
      <Navbar/>  
      <Switch>  
        <Route path="/" exact>  
          <Home />  
        </Route>  
        <Route path="/board" exact>  
          <BoardList/>  
        </Route>  
        <Route path="/board/write">  
          <WriteBoard/>  
        </Route>  
      </Switch>  
    </Router>  
  );  
}
```

exact 키워드 명시

그럼 이제 의도한대로 작동 됩니다.
그럼 해당 URL을 이동시키는 컴포넌트를 사용해야 겠지요?

01 React-Router

Navbar.js

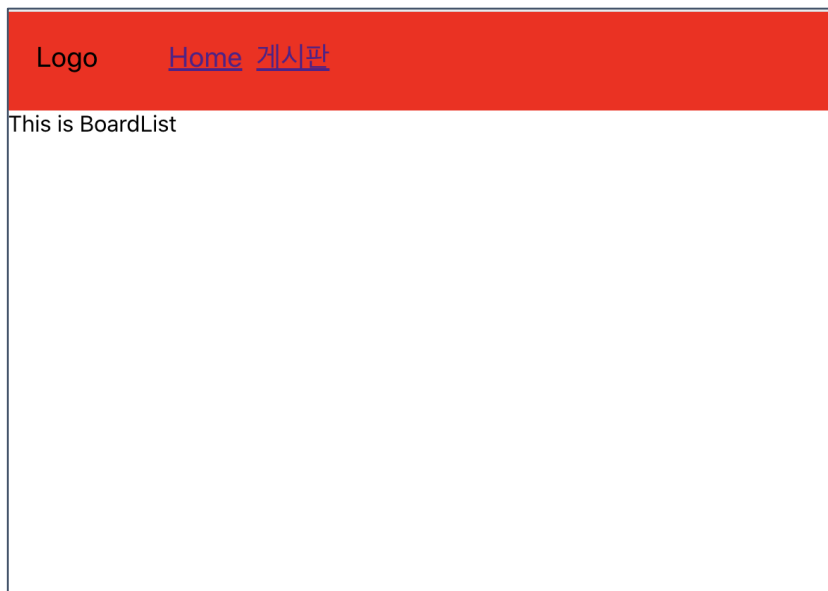
Navbar.js에 필요한 Navigation을 넣을 예정입니다.

컴포넌트 <Link/>

```
1  import React from 'react'
2
3  import {Link} from 'react-router-dom'
4  export default function Navbar() {
5    return (
6      <nav style={{width:'100%', backgroundColor: 'red', height: "30px", padding:20, fontSize:20}}>
7        <div style={{display:"inline-block", marginRight:50, }}>Logo</div>
8        <div style={{display:"inline-block"}}>
9          <Link to="/" style={{marginRight: 10}}>
10             Home
11          </Link>
12          <Link to="/board" style={{marginRight: 10}}>
13             게시판
14          </Link>
15        </div>
16      </nav>
17    )
18  }
```

디자인은 나중에 생각하고, 우선 React-Router 에 대해 알아보시다.

01 React-Router



클릭할 수 있는 링크들이 생겼습니다.

Link 컴포넌트도 `anchor(a)`태그를 사용합니다.
하지만, 내부적으로 동작은 완전히 다릅니다.

`anchor`태그는 기본적으로 서버에 request를
날리지만, 해당 `BrowserRouter`의 `Link`
컴포넌트는 `Client Side Router`이기 때문에
요청을 새로 날리지 않습니다.

⇒ 성능 최적화가 가능하고, 사용자
interaction에 빠르게 대응할 수 있습니다.

01 React-Router

가끔 작업을 하다 보면, Page Component에서 Router객체에 접근할 필요가 있을 때가 있습니다.

이럴테면, 뒤로가기 버튼이라던가, 로그인 버튼 클릭시 페이지 이동이라던가..

그 경우 Router Component내부에서 Router객체에 접근을 해주어야 합니다.

App.js를 다음과 같이 수정합니다.

```
1 import './App.css';
2 import {
3   BrowserRouter as Router,
4   Switch,
5   Route,
6 } from 'react-router-dom';
7 import Home from './pages/Home';
8 import WriteBoard from './pages/WriteBoard';
9 import BoardList from './pages/BoardList';
10 import Navbar from './components/Navbar';
11
12 function App() {
13   return (
14     <Router>
15       <Navbar/>
16       <Switch>
17         <Route path="/" exact component={Home}/>
18         <Route path="/board/write" component={WriteBoard}/>
19         <Route path="/board" component={BoardList}/>
20       </Switch>
21     </Router>
22   );
23 }
24
25 export default App;
```

해당 PageComponent에서 우리는 Router 객체에 접근해야 합니다.
Router객체는 보시는 것처럼 상위(부모) 컴포넌트에 속해 있습니다
자식컴포넌트에서 부모컴포넌트의 데이터(객체)를 전달 받는 방법은?

App.js

Props(?)

State(?)

01 React-Router

Props로 Router객체 전달 받기

console을 찍어봅시다.

App.js에서 우린 해당 컴포넌트에 props를 전달하지 않았음을 확인합니다.

```
▼ {history: {...}, location: {...}, match: {...}, staticContext: undefined} ⓘ  
  ▶ history: {length: 20, action: "PUSH", location: {...}, createHref: f, push: f, ...}  
  ▶ location: {pathname: "/", search: "", hash: "", state: undefined, key: "j7rh3i"}  
  ▶ match: {path: "/", url: "/", isExact: true, params: {...}}  
    staticContext: undefined  
  ▶ __proto__: Object
```

```
export default function Home(props) {  
  console.log(props);  
  return (  
    <div>  
      This is Home  
    </div>  
  )  
}
```

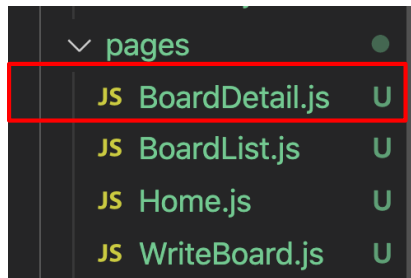
다음과 같이 3개의 key가 존재하는 객체를 받았습니다.
이를 적절히 조작해주면, 우리는 조금 더 나은 UI를 구성할 수 있습니다.

- history: 브라우저가 이동한 경로를 기록함. (push함수, goBack함수, goForward함수가 존재)
- location: 현 위치에 대한 정보를 기록함
- match: 어떤 경로와 matching 되었는지, 전달받은 parameter는 무엇인지를 얻을 수 있음.

01 React-Router

Router객체 활용하기 (BoardDetail)

BoardDetail.js를 만듭니다.



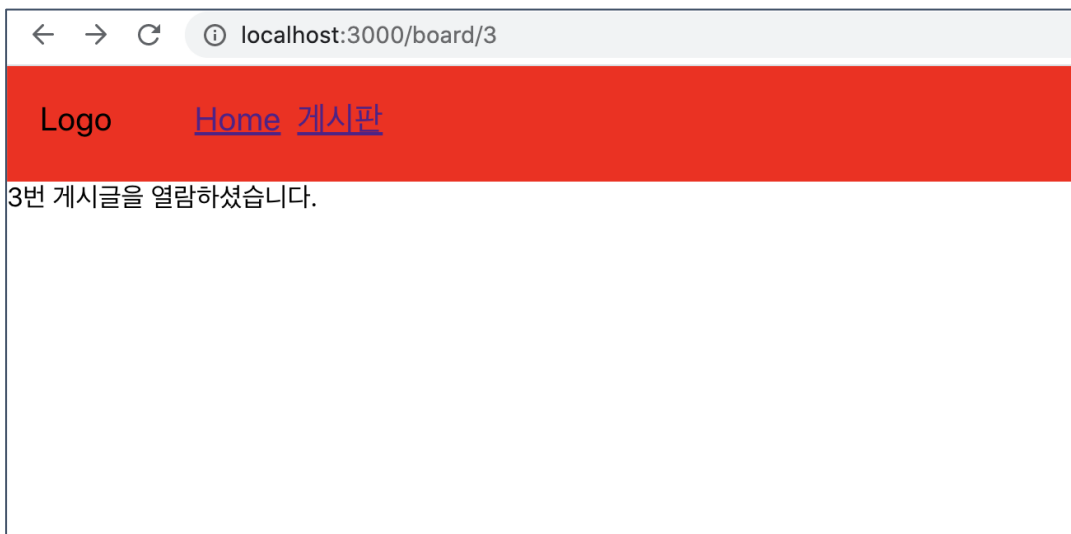
```
1 import React from 'react'
2
3 export default function BoardDetail(props) {
4   const boardId = props.match.params.id
5   return (
6     <div>
7       {boardId}번 게시글을 열람하셨습니다.
8     </div>
9   )
10 }
```

Component에 접근하는 Route를 정의합니다.

```
function App() {
  return (
    <Router>
      <Navbar/>
      <Switch>
        <Route path="/" exact component={Home}/>
        <Route path="/board/write" exact component={WriteBoard}/>
        <Route path="/board" exact component={BoardList}/>
        <Route path="/board/:id" exact component={BoardDetail}/>
      </Switch>
    </Router>
  );
}
```

01 React-Router

다음처럼 나옵니다.



이 말이 의미하는 바는?

- ⇒ Router객체를 받으면 URL을 통해 전달받는 parameter에 접근 가능하다.
- ⇒ 게시글(데이터 리스트)중 특정 게시물(데이터)의 ID를 가져올 수 있다.
- ⇒ 세부 ID를 이용해 서버로 GET요청을 보낼 수 있다.
- ⇒ 게시글(데이터) 상세 데이터를 가져올 수 있고, 이를 독립적으로 그려줄 수 있다.

01 React-Router

React-Router-Dom 정리.

React-Router-Dom의 주요 Component

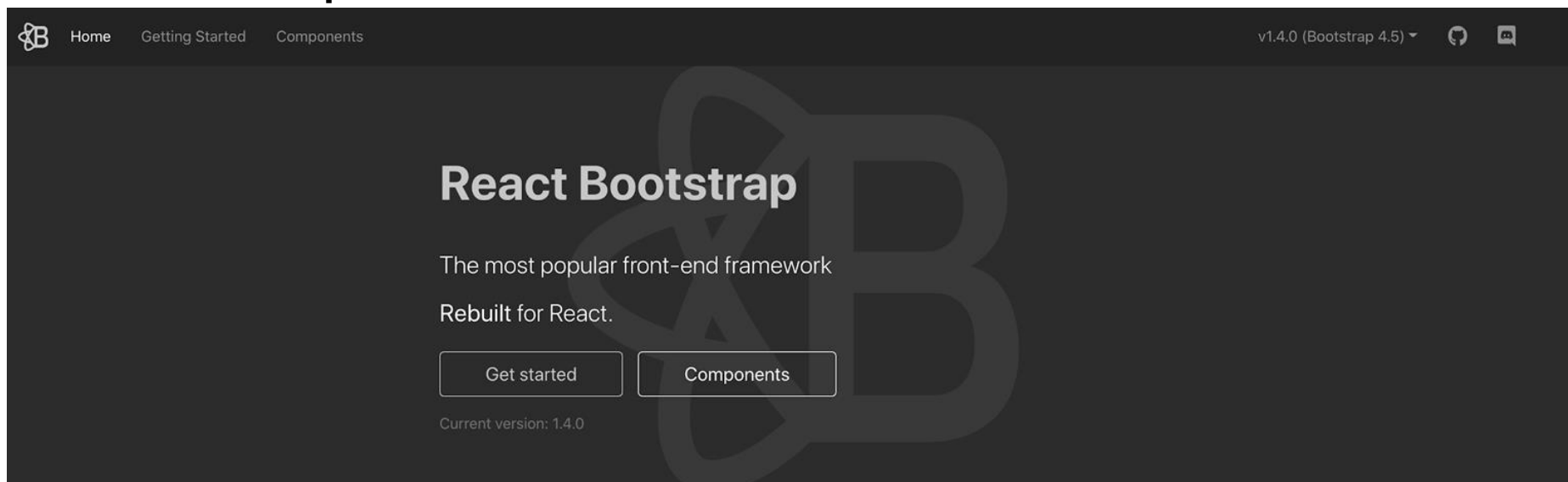
- **BrowserRouter**: 라우터를 적용할 영역(최상위 위치)
- **Switch**: 라우터가 적용되어 바뀔 영역
- **Route**: URL 경로와 매칭될 Component
- **Link**: URL경로로 이동시키는 Component.

Component에서 Router 객체 접근하기

- Props로 전달받아서 사용.
- 전달받는 Props는 {history, location, match}를 각각 포함하고 있음.
 - history: 브라우저가 이동한 경로를 기록함. (push함수, goBack함수, goForward함수가 존재)
 - location: 현 위치에 대한 정보를 기록함
 - match: 어떤 경로와 matching 되었는지, 전달받은 parameter는 무엇인지를 얻을 수 있음.

01부록

React-Bootstrap



<https://react-bootstrap.github.io>

컴포넌트가 조금 복잡하게 구성되어 있습니다. 하지만 이것 이해할 필요는 없습니다. 다만, 잘 사용하려고 하시면 됩니다.

React-Bootstrap과 비슷한 것들이 material-ui, ants-ui, core-ui 등 굉장히 많습니다. 이것 다 외우는 것은 현실적으로 불가능합니다. 하지만 이것들은 각각 디자인 철학의 차이일 뿐 가장 중요한 점들은 다 비슷합니다. 매뉴얼을 보며 잘 사용하시고, 자주 사용하시는 것들의 컨셉만 기억하십시오.