React

Frontend library / framework

- 정적 페이지 → HTML + CSS
- 동적 페이지 → Javascript, 사용자와의 interact 작업

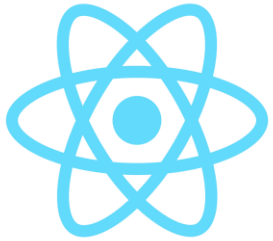

```html
<div>
  <h1>Counter</h1>
  <h2 id="counter">0</h2>
  <button id="increase">+</button>
</div>
```

```html
<script>
  let counter = 0;
  let currentNumner = document.getElementById("counter");
  let btnInc = document.getElementById("increase");

  btnInc.onclick = function() {
    console.log(counter);
    counter++;
    currentNumner.innerText = counter;
  }
</script>
```

```html
<script>
  let counter = 0;

  $("#increase").on("click", function() {
    counter++;
    $("#counter").text(counter);
  });
</script>
```

Frontend library / framework

- Frontend library or framework → DOM 관리 및 상태 값 업데이트
- Angular, Backbone, Ember, React, Vue.js



- Angular
  - Google
  - 라우터, HTTP 클라이언트 등의 필요한 도구가 프레임워크 안에 내장
  - Typescript

Frontend library / framework

- React
  - Facebook
  - Component
  - HTTP 클라이언트, 라우터, 상태 관리 등의 기능이 내장되어 있지 않기 때문에, 자유롭게 사용 가능하며, 직접 라이브러리 용이

- Vue
  - 사용 간단, HTML을 템플릿처럼 사용 가능
  - CDN의 파일 로딩하여 스크립트 실행
  - 공식 라우터, 상태관리 라이브러리 존재

- React
  - https://reactjs.org/

> "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*."

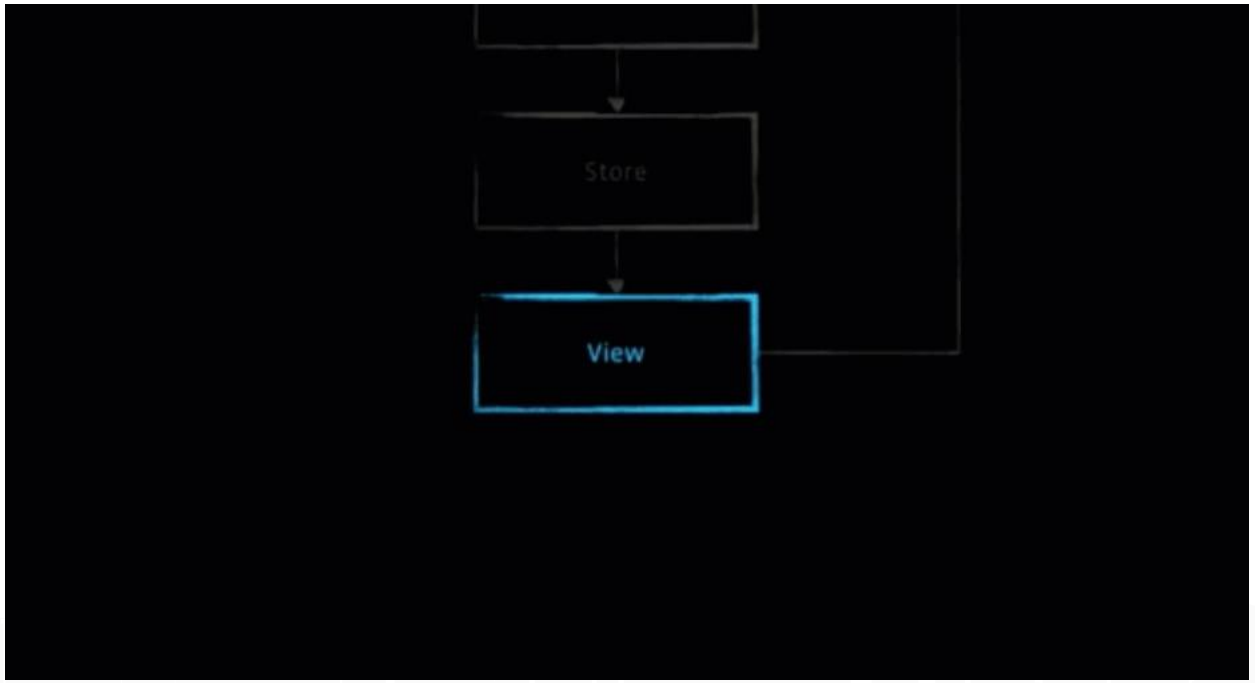
- MVC (***Model***, View, Controller)
- MVVM (***Model***, View, View Model)
- MVW (***Model***, View, Whatever)

→ 양방향 바인딩 모델

모델의 값이 변경되면 뷰에도 변화(Mutation)

- 데이터의 Mutation
  - 특정 이벤트 → 모델 변화 → DOM에 데이터 반영

> 그냥 Mutation 을 하지 말자. 그 대신에, 데이터가 바뀌면 그냥 뷰를 날려버리고 새로 만들어버리면 어떨까?

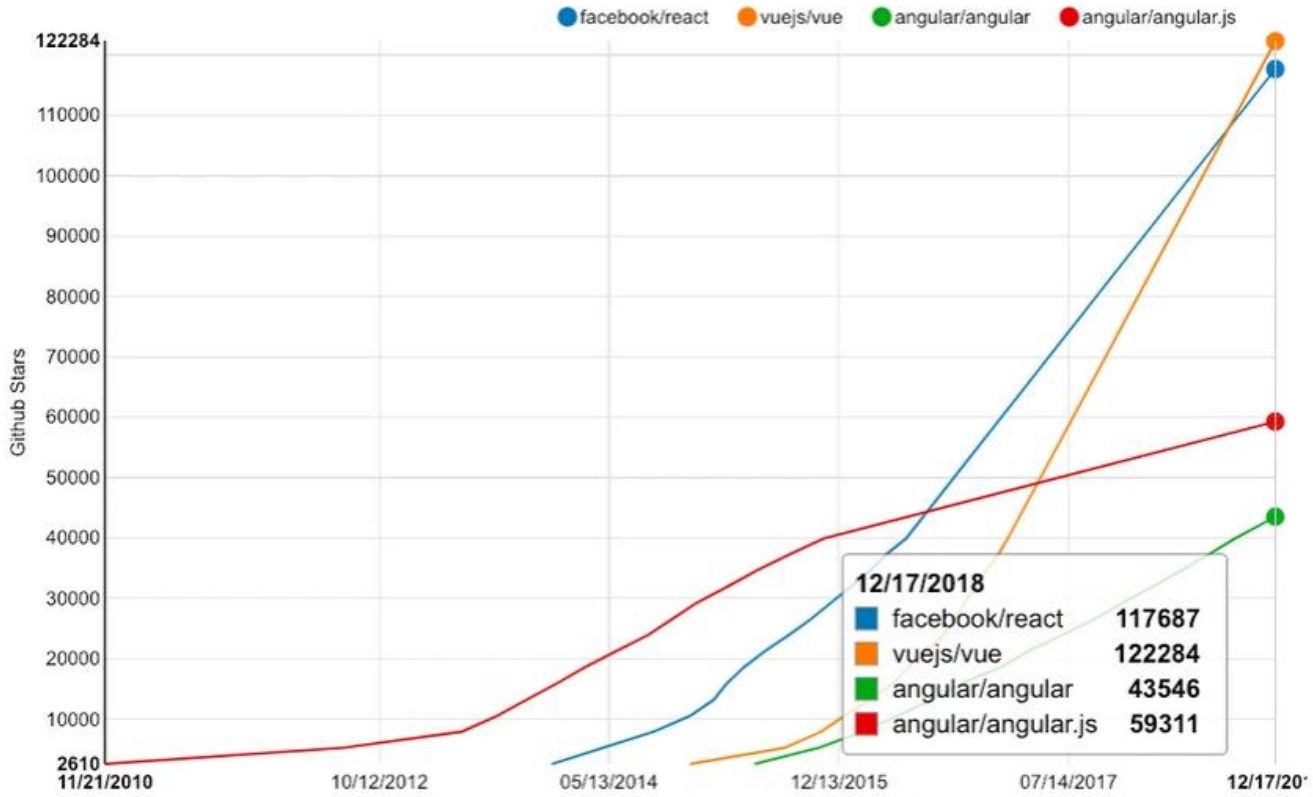


→ ***Virtual DOM***

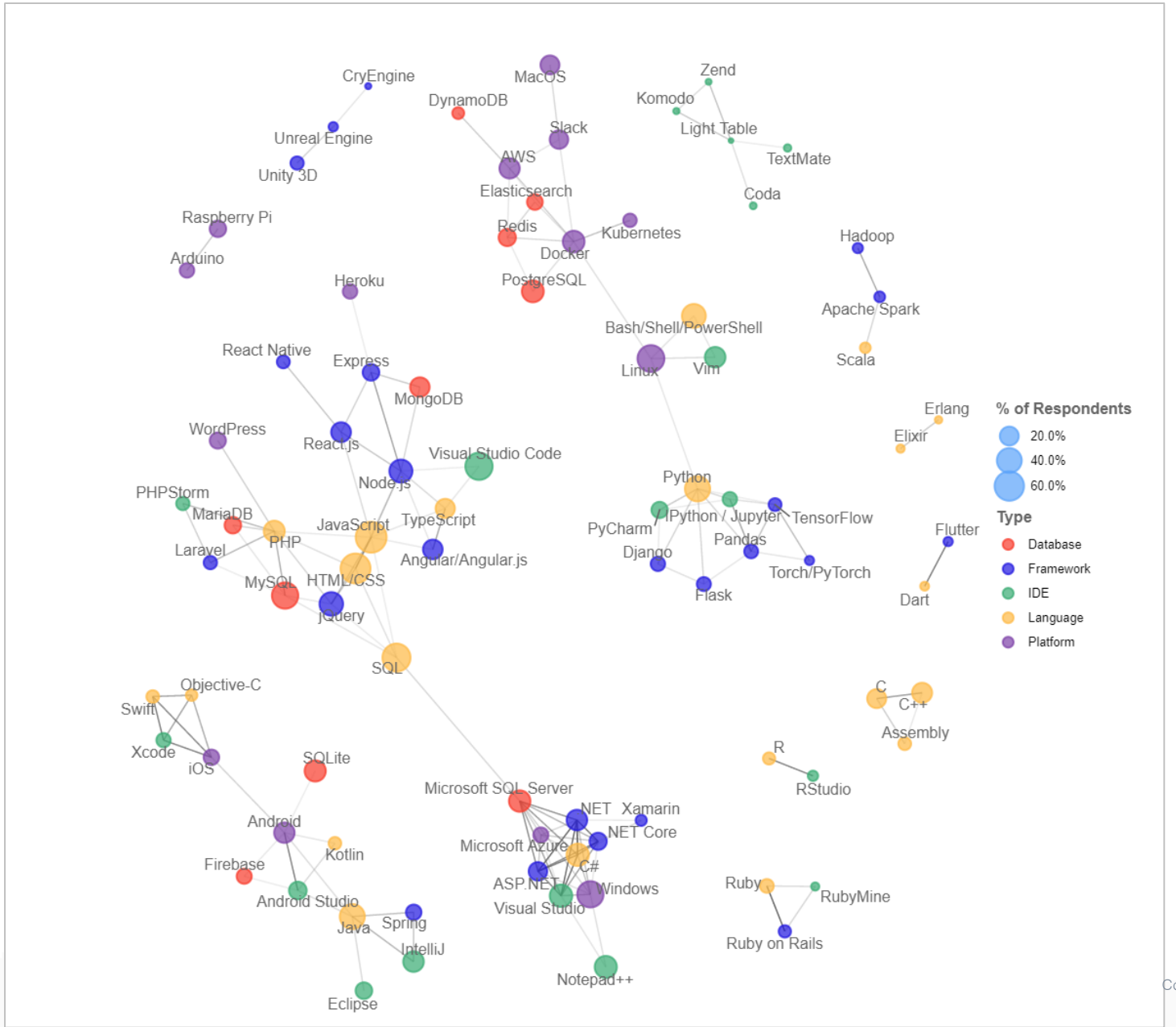
https://www.youtube.com/watch?v=muc2ZF0QIO4

React 특징

- libraries
- references
  - airbnb, bbc, cloudflare, codeacademy, coursera, dailymotion, ebay, walmart, yahoo ...



React 특징



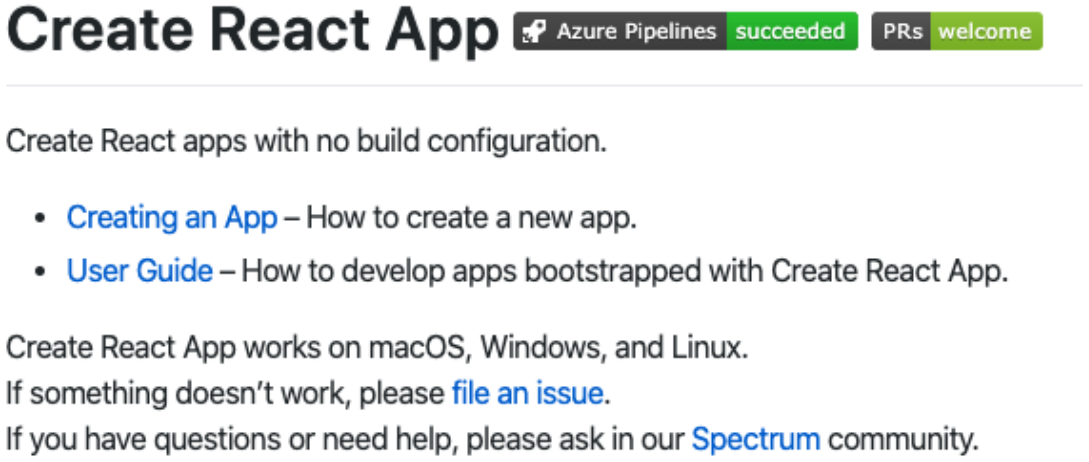
Connected technologies in software development.
Image: Stack Overflow

React 특징

- 높은 자유도
  - 라우터, 상태관리는 자체 내장되어 있지 않기 때문에, 자율적인 3rd party library 사용 가능
    - ex) Router → React-router, Next.js, After.js
    - ex) State → Redux, MobX

React Project

- Node.js
- Yarn
  - facebook에서 만든 자바스크립트 패키지 매니저 설치) npm install –g yarn
  - yarn –version
  - 더 나은 속도, 더 나은 캐싱 시스템
- Webpack
- Babel
- create-react-app
  - 페이스북에서 제공하는 React template
  - https://github.com/facebook/create-react-app



React Project

- Webpack
  - 리액트 프로젝트는 컴포넌트를 여러가지 파일로 분리해서 저장 → JSX 문법
  - 여러가지 파일을 한개로 결합하기 위한 도구
- Babel
  - https://babeljs.io/
  - 최신 사양의 자바스크립트 코드를 IE나 구형 브라우저에서도 동작하는 ES5 이하의 코드로 변환 (트랜스파일링)

```
// ES6 화살표 함수와 ES7 지수 연산자
[1, 2, 3].map(n => n ** n);
```

```
// ES5
"use strict";
[1, 2, 3].map(function (n) {
  return Math.pow(n, n);
});
```

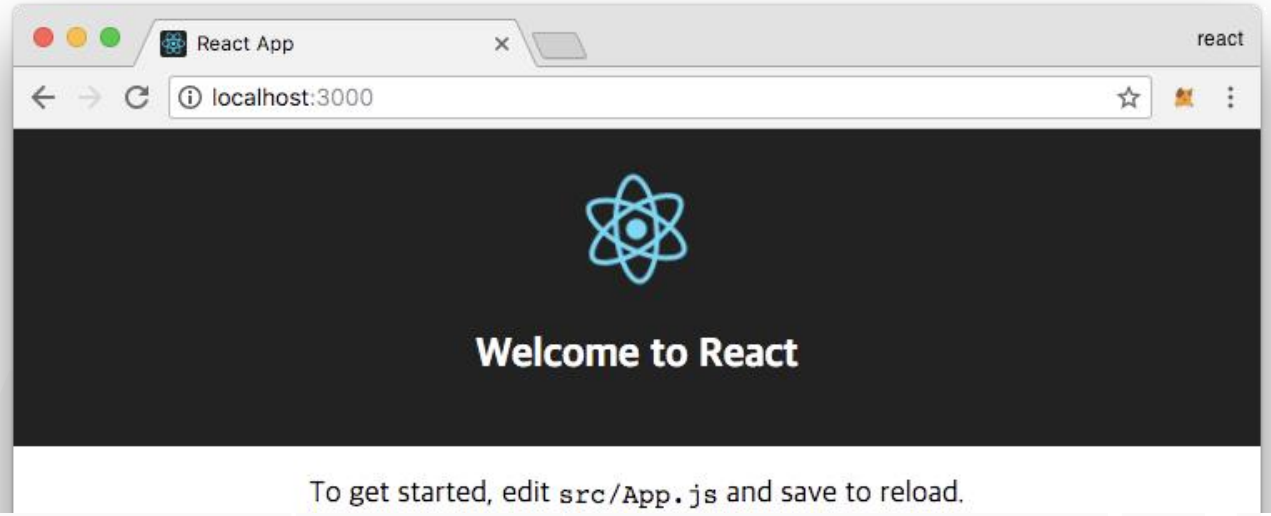
React Project

- create-react-app
  - react template 생성

Quick Overview

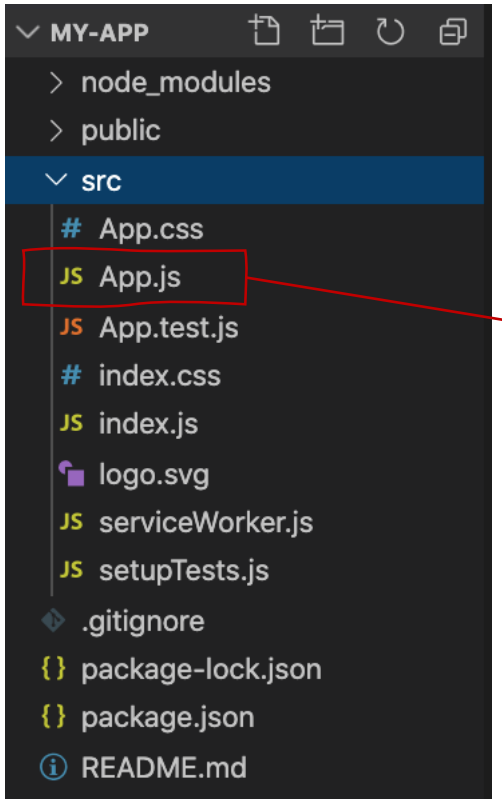
```
npx create-react-app my-app
cd my-app
npm start
```

- http://localhost:3000/



React Project

- 프로젝트 구조



App.js 컴포넌트 시작 파일

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

React Project

- App.js

```
1  import React from 'react';
2  import logo from './logo.svg';
3  import './App.css';
```

리액트 Component ,외부 Component
CSS, images 등 불러오기

```
function App() {…
}
```

```
class App extends Component() {…
}
```

컴포넌트 생성 → Class or Function
내부에서 JSX를 반환

```
28  export default App;
```

외부에서 사용할 수 있도록 선언

- JSX 작성 규칙
  - React.createElement 코드를 HTML 처럼 작성할 수 있도록 지원

```
1  import React, { Component } from 'react';
2
3  class App extends Component {
4    render() {
5      return (
6        <div>
7
8        </div>
9      );
10   }
11 }
12
13 export default App;
```

가장 기본적인 JSX 구문

- *하나의 root element를 가짐*
- *모든 element는 closer 필요 ex) div, Fragment*
- *Javascript의 값 사용 시 { , } 사용*

- JSX 작성 규칙
  - 조건부 렌더링

```
render() {
  const time = 10;
  const name = 'Dowon';
  return (
    <div>
      {
        time < 15
          ? (<div>Hello, {name}</div>)
          : (<div>Bye, {name}</div>)
      }
    </div>
  );
}
```

삼항 연산자

- *삼항 연산자나 AND 연산자를 사용한 조건부 렌더링*
- *if문 사용할 수 없음 (IIFE 표기법 사용해야 함)*

```
{
  name === 'Dowon' && (<div>Manager</div>)
}
```

AND 연산자

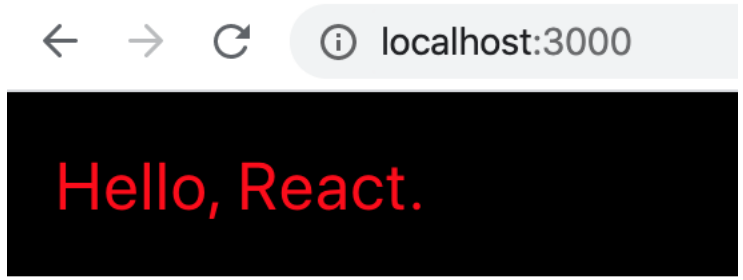
- JSX 작성 규칙
  - IIFE

```
render() {
  const time = 10;
  const name = 'Dowon';
  return (
    <div>
      {
        (function() {
          if (time < 12) return (<div>Good morning, {name}</div>);
          if (time < 18) return (<div>Good afternoon, {name}</div>);
          if (time < 22) return (<div>Good evening, {name}</div>);
        })()
      }
    </div>
  );
}
```

IIFE 표시법에 의한 if 구문 사용

- JSX 작성 규칙
  - style, className

```
render() {
  const css = {
    background: 'black',
    padding: '20px',
    color: 'red',
    fontSize: '25px'
  }

  return (
    <div style={css}>
      Hello, React.
    </div>
  );
}
```



객체 형태로 style 작성

- JSX 작성 규칙
  - style, className

```css
.App-header {
  background-color: #282c34;
  min-height: 100vh;
  display: flex;
  flex-direction: column;
  align-items: center;
  justify-content: center;
  font-size: calc(10px + 2vmin);
  color: white;
}
```

App.css

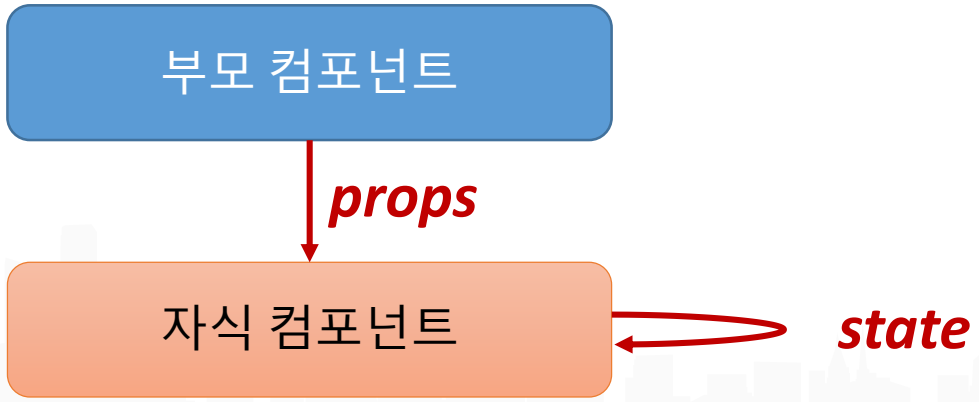
```jsx
import './App.css'

class App extends Component {
  render() {
    return (
      <div className="App-header">
        Hello, React.
      </div>
    );
  }
}
```

App.js

- JSX 작성 규칙
  - comment

```
render() {
  return (
    // 주석입니다.
    <div className="App-header"
      // 주석입니다.
    >
      {/* 주석입니다. */}
      Hello, React.
    </div>
  );
}
```

- props
  - 부모 컴포넌트가 자식 컴포넌트에게 전달하는 값
  - 자식 컴포넌트에서는 props의 값을 수정할 수 없음
  - props 값은 this. 키워드를 이용하여 사용
- state
  - 컴포넌트 내부에 선언하여 사용되는 보관용 데이터 값
  - 동적인 데이터 처리



- props

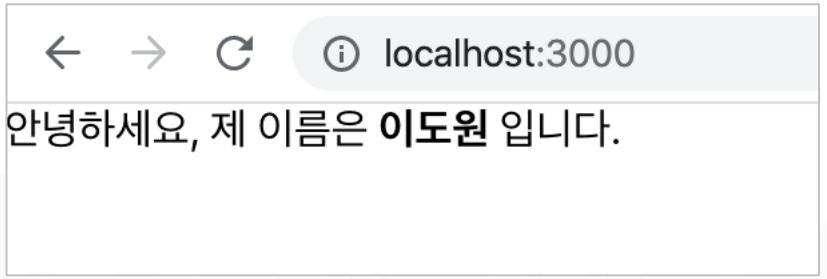
```jsx
import React, { Component } from 'react';

class MyIntro extends Component {
  render() {
    return (
      <div>
        안녕하세요, 제 이름은 <b>{this.props.name}</b> 입니다.
      </div>
    );
  }
}

export default MyIntro;
```

MyIntro.js

```jsx
const MyIntro2 = ({name}) => {
    return (
      <div>
        안녕하세요, 제 이름은 <b>{name}</b> 입니다.
      </div>
    );
}
```

```jsx
class App extends Component {
  render() {
    return (
      <MyIntro name="이도원" />
    );
  }
}
```

App.js



- state

```
class Counter extends Component {
  state = {
    number: 0
  }

  render() {
    return (
      <div>
        <h1>카운터</h1>
        <div>값: 0</div>
        <button >+</button>
        <button >-</button>
      </div>
    );
  }
}
```

Counter.js

state의 값으로 대체 하려면?

→ {this.state.number}

- state

```
handleIncrease = () => {
  this.setState({
    number: this.state.number + 1
  });
}

handleDecrease = () => {
  this.setState({
    number: this.state.number - 1
  });
}
```

Counter.js

- *state의 값을 변경하기 위해 사용*
- *{ number }의 값만 변경*

```
state = {
  number: 0,
  info: {
    name: 'react',
    age: 10
  }
}
```

```
this.setState({
  info: {
    name: 'Dowon'
  }
});
```

새로운 info

```
this.setState({
  info: {
    name: 'Dowon',
    ...this.state.info
  }
});
```

기존 info

이벤트 설정

```
render() {
  return (
    <div>
      <h1>카운터</h1>
      <div>값: {this.state.number}</div>
      <button onClick={this.handleIncrease}>+</button>
      <button onClick={this.handleDecrease}>-</button>
    </div>
  );
}
```

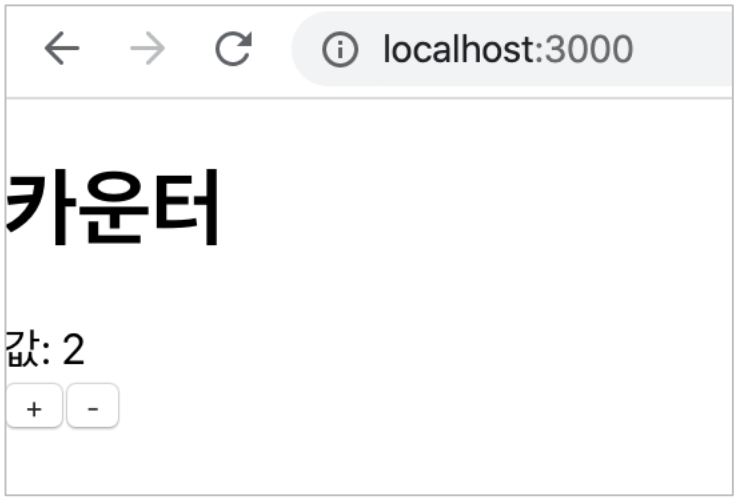
Counter.js

- *이벤트 이름은 camelCase*
- *이벤트에 전달되는 값은 함수*

- Counter 페이지 실행

```
class App extends Component {
  render() {
    return (
      <Counter />
    );
  }
}
```

App.js



- LifeCycle API

```
constructor(props) {
  super(props);
  console.log('constructor');
}

componentWillMount() {
  console.log('componentWillMount (deprecated)');
}

componentDidMount() {
  console.log('componentDidMount');
}
```

Counter.js

```
shouldComponentUpdate(nextProps, nextState) {
  // 5 의 배수라면 리렌더링 하지 않음
  console.log('shouldComponentUpdate');
  if (nextState.number % 5 === 0) return false;
  return true;
}

componentWillUpdate(nextProps, nextState) {
  console.log('componentWillUpdate');
}

componentDidUpdate(prevProps, prevState) {
  console.log('componentDidUpdate');
}
```

Counter.js

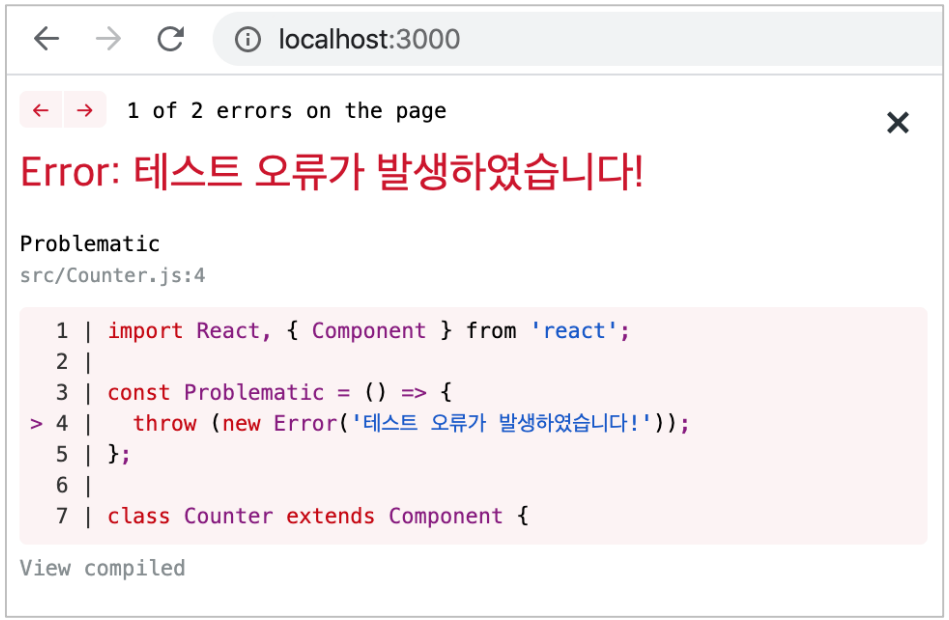
에러 발생

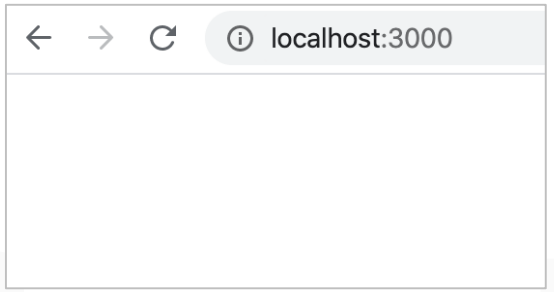
```
const Problematic = () => {
  throw (new Error('테스트 오류가 발생하였습니다!'));
};
```

```
componentDidCatch(error, info) {
  this.setState({
    error: true
  });
}
```

```
<div>값: {this.state.number}</div>
{ this.state.number === 4 && <Problematic /> }
<button onClick={this.handleIncrease}>+</button>
```

Counter.js

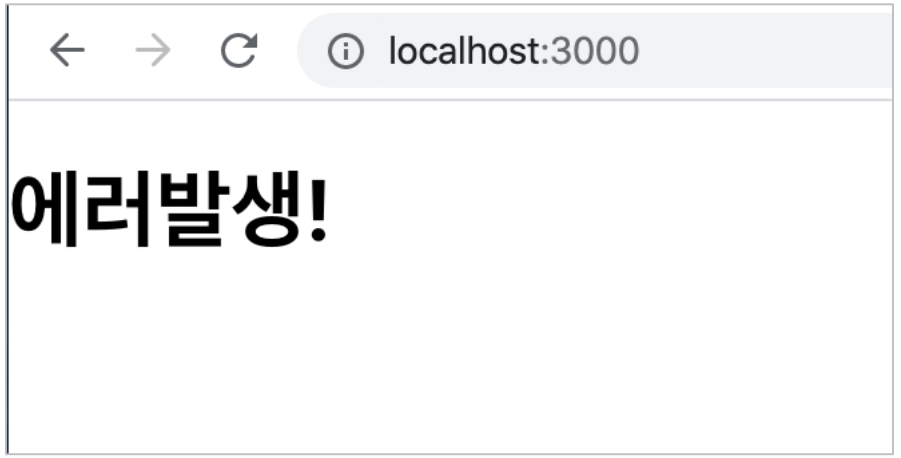




- 에러 발생

```
render() {
  if (this.state.error) return (<h1>에러발생!</h1>);

  return (
    <div>
      <h1>카운터</h1>
      <div>값: {this.state.number}</div>
      { this.state.number === 4 && <Problematic /> }
      <button onClick={this.handleIncrease}>+</button>
      <button onClick={this.handleDecrease}>-</button>
    </div>
  );
}
```

Counter.js



React Project - Demo

- 프로젝트 생성
  - $ npx create-react-app contact-app
  - $ cd contact-app
  - $ yarn start (or npm start)

- PhoneForm 컴포넌트
  - 사용자 이름과 전화번호 입력
  - src/components/PhoneForm.js 생성

React Project - Demo

- PhoneForm 컴포넌트

```jsx
import React, { Component } from 'react';

class PhoneForm extends Component {
  state = {
    name: ''
  }

  handleChange = (e) => {
    this.setState({
      name: e.target.value
    })
  }

  render() {
    return (
      <form>
        <input
          placeholder="이름"
          value={this.state.name}
          onChange={this.handleChange}
        />
        <div>{this.state.name}</div>
      </form>
    );
  }
}

export default PhoneForm;
```

PhoneForm.js

React Project - Demo

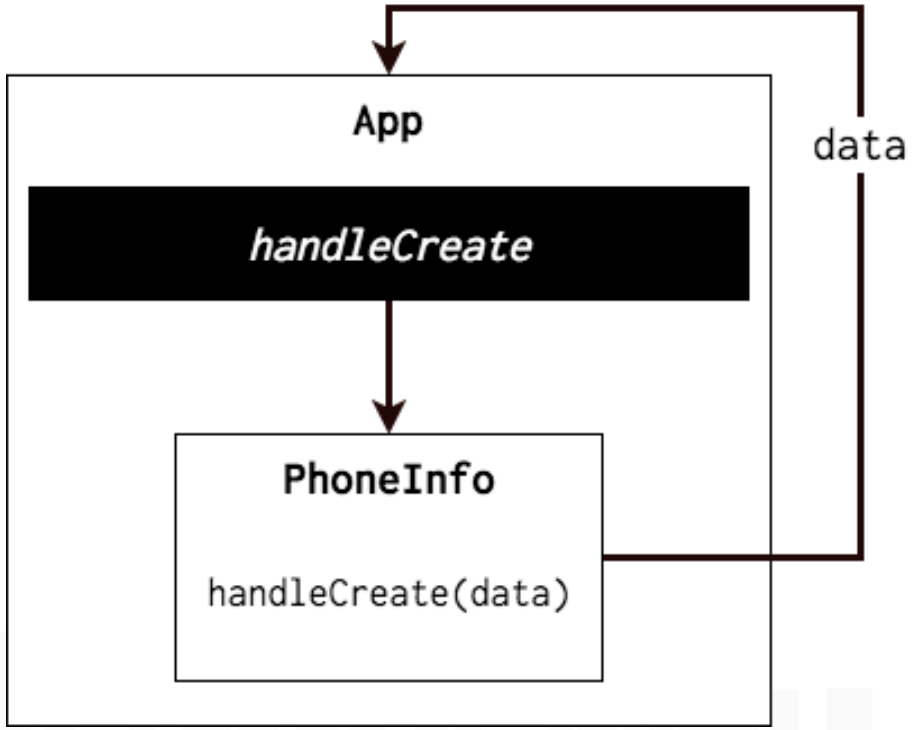
- PhoneForm 컴포넌트
  - 전화번호 필드 추가

```jsx
import React, { Component } from 'react';

class PhoneForm extends Component {
  state = {
    name: '',
    phone: ''
  }

  handleChange = (e) => {
    this.setState({
      [e.target.name]: e.target.value
    });
  }

  render() {
    return (
      <form>
        <input
          placeholder="이름"
          value={this.state.name}
          onChange={this.handleChange}
        />
        <input
          placeholder="전화번호"
          value={this.state.phone}
          onChange={this.handleChange}
          name="phone"
        />
        <div>{this.state.name} {this.state.phone}</div>
      </form>
    );
  }
}

export default PhoneForm;
```

PhoneForm.js

React Project - Demo

- 부모 컴포넌트에 정보 전달
  - state의 값을 부모 컴포넌트에게 전달
  - 부모 컴포넌트의 함수를 자식 컴포넌트에 전달 → 자식 컴포넌트에서 부모의 함수 호출



React Project - Demo

- 부모 컴포넌트에 정보 전달

```
class App extends Component {
  handleCreate = (data) => {
    console.log(data);
  }

  render() {
    return (
      <PhoneForm
        onCreate={this.handleCreate}/>
    );
  }
}
```

App.js

```
handleSubmit = (e) => {
  this.props.onCreate(this.state);
  // 상태 초기화
  this.setState({
    name: '',
    phone: ''
  })
}

render() {
  return (
    <form onSubmit={this.handleSubmit}>
      <button type="submit">등록</button>
    </form>
```

PhoneForm.js

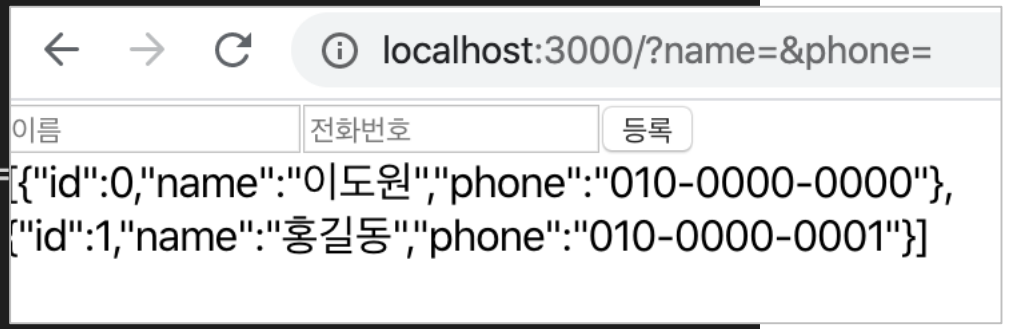
React Project - Demo

- 애플리케이션 상태 관리 → App.js
  - state 에 contacts 배열 생성

```
class App extends Component {
  id = 2;
  state = {
    contacts: [
      {
        id: 0,
        name: '이도원',
        phone: '010-0000-0000'
      },
      {
        id: 1,
        name: '홍길동',
        phone: '010-0000-0001'
      }
    ]
  }
```

App.js

```
  handleCreate = (data) => {
    console.log(data);
    const { contacts } = this.state;
    this.setState({
      contacts: contacts.concat({ id: this.id++, ...data })
    })
  }

  render() {
    const { contacts } =
    return (
      <div>
        <PhoneForm
          onCreate={this.handleCreate} />
        {JSON.stringify(contacts)}
      </div>
    );
```



React Project - Demo

- PhoneInfo 컴포넌트

```jsx
import React, { Component } from 'react';

class PhoneInfo extends Component {
  static defaultProps = {
    info: {
      name: '이름',
      phone: '010-0000-0000',
      id: 0
    }
  }

  render() {
    const style = {
      border: '1px solid black',
      padding: '8px',
      margin: '8px'
    };

    const {
      name, phone, id
    } = this.props.info;

    return (
      <div style={style}>
        <div><b>{name}</b></div>
        <div>{phone}</div>
      </div>
    );
  }
}

export default PhoneInfo;
```

PhoneInfo.js

React Project - Demo

- PhoneInfoList 컴포넌트

```jsx
import React, { Component } from 'react';
import PhoneInfo from './PhoneInfo';

class PhoneInfoList extends Component {
  static defaultProps = {
    data: []
  }

  render() {
    const { data } = this.props;
    const list = data.map(
      info => (<PhoneInfo key={info.id} info={info}/>)
    );

    return (
      <div>
        {list}
      </div>
    );
  }
}

export default PhoneInfoList;
```

PhoneInfoList.js

```jsx
<div>A</div>
<div>B</div>
<div>C</div>
<div>D</div>
```

Key를 사용하지 않을 때

```jsx
<div key={0}>A</div>
<div key={1}>B</div>
<div key={2}>C</div>
<div key={3}>D</div>
```

Key를 사용할 때

React Project - Demo

- App.js에서 PhoneInfoList 컴포넌트 렌더링

```
31    render() {
32      return (
33        <div>
34          <PhoneForm
35            onCreate={this.handleCreate} />
36          <PhoneInfoList data={this.state.contacts}/>
37        </div>
38      );
39    }
```

App.js

React Project - Demo

- 데이터 삭제

```
> const myarr = [1, 2, 3, 4, 5];
```

  - 3만 삭제

```
> myarr.slice(0,2).concat(myarr.slice(3,5))

> [...myarr.slice(0,2), ...myarr.slice(3,5)]

> myarr.filter(n => n !== 3);
```

React Project - Demo

- 데이터 삭제

```
30    handleRemove = (id) => {
31      const { contacts } = this.state;
32      this.setState({
33        contacts: contacts.filter(info => info.id !== id)
34      })
35    }
36
37    render() {
38      return (
39        <div>
40          <PhoneForm
41            onCreate={this.handleCreate} />
42          <PhoneInfoList
43            data={this.state.contacts}
44            onRemove={this.handleRemove}
45          />
46        </div>
47      );
48    }
```

App.js

React Project - Demo

데이터 삭제

```
4  class PhoneInfoList extends Component {
5    static defaultProps = {
6      data: [],
7      onRemove: () => console.warn('onRemove not defined'),
8    }
9
10   render() {
11     const { data, onRemove } = this.props;
12     const list = data.map(
13       info => (
14         <PhoneInfo
15           key={info.id}
16           info={info}
17           onRemove={onRemove}
18         />)
19     );
```

PhoneInfoList.js

```
12   handleRemove = () => {
13     // 삭제 버튼이 클릭되면 onRemove 에 id 넣어서 호출
14     const { info, onRemove } = this.props;
15     onRemove(info.id);
16   }
```

```
29     return (
30       <div style={style}>
31         <div><b>{name}</b></div>
32         <div>{phone}</div>
33         <button onClick={this.handleRemove}>삭제</button>
34       </div>
35     );
```

PhoneInfo.js

React Project - Demo

- 데이터 수정

```
> const mytag = [
    { id: 0, text: 'hello', tag: 'a' },
    { id: 1, text: 'world' , tag: 'b' },
    { id: 2, text: 'bye', tag: 'c' }
  ];

> const modifiedArray = mytag.map(item => item.id === 1 ? ({ ...item, text: 'Korea' }) : item);

> modifiedArray
<- ▼(3) [{…}, {…}, {…}]
    ▶0: {id: 0, text: "hello", tag: "a"}
    ▶1: {id: 1, text: "Korea", tag: "b"}
    ▶2: {id: 2, text: "bye", tag: "c"}
     length: 3
    ▶__proto__: Array(0)
```

React Project - Demo

- 데이터 수정

```
handleUpdate = (id, data) => {
  const { contacts } = this.state;
  this.setState({
    contacts: contacts.map(
      info => id === info.id
        ? { ...info, ...data } // 새 객체를 만들어서 기존의 값과 전달받은 data 을 덮어씀
        : info // 기존의 값을 그대로 유지
    )
  })
}

render() {
  return (
    <div>
      <PhoneForm
        onCreate={this.handleCreate} />
      <PhoneInfoList
        data={this.state.contacts}
        onRemove={this.handleRemove}
        onUpdate={this.handleUpdate}
      />
    </div>
  );
```

App.js

React Project - Demo

- 데이터 수정

```
class PhoneInfoList extends Component {
  static defaultProps = {
    data: [],
    onRemove: () => console.warn('onRemove not defined'),
    onUpdate: () => console.warn('onUpdate not defined'),
  }

  render() {
    const { data, onRemove, onUpdate } = this.props;
    const list = data.map(
      info => (
        <PhoneInfo
          key={info.id}
          info={info}
          onRemove={onRemove}
          onUpdate={onUpdate}
        />)
    );
```

PhoneInfoList.js

React Project - Demo

- 데이터 수정

```
12  state = {
13    editing: false,
14    name: '',
15    phone: '',
16  }
17
18  handleToggleEdit = () => {
19    const { editing } = this.state;
20    this.setState({ editing: !editing });
21  }
22
23  handleChange = (e) => {
24    const { name, value } = e.target;
25    this.setState({
26      [name]: value
27    });
28  }
```

```
30  componentDidUpdate(prevProps, prevState) {
31    const { info, onUpdate } = this.props;
32    if(!prevState.editing && this.state.editing) {
33      this.setState({
34        name: info.name,
35        phone: info.phone
36      })
37    }
38
39    if (prevState.editing && !this.state.editing) {
40      onUpdate(info.id, {
41        name: this.state.name,
42        phone: this.state.phone
43      });
44    }
45  }
```

PhoneInfo.js

React Project - Demo

데이터 수정

```
60      const { editing } = this.state;
61
62      if (editing) { // 수정모드
63        return (
64          <div style={style}>
65            <div>
66              <input
67                value={this.state.name}
68                name="name"
69                placeholder="이름"
70                onChange={this.handleChange}
71              />
72            </div>
73            <div>
74              <input
75                value={this.state.phone}
76                name="phone"
77                placeholder="전화번호"
78                onChange={this.handleChange}
79              />
80            </div>
81            <button onClick={this.handleToggleEdit}>적용</button>
82            <button onClick={this.handleRemove}>삭제</button>
83          </div>
84        );
85      }
```

PhoneInfo.js

React Project - Demo

- 데이터 필터링
  - App.js
    - input 태그 렌더링 → state에 추가 (keyword)
    - 이벤트 핸들러 추가

```
 8    state = {
 9      contacts: [
```

```
21      keyword: ''
22    }
23
24    handleChange = (e) => {
25      this.setState({
26        keyword: e.target.value,
27      });
28    }
```

App.js

```
55    render() {
56      return (
57        <div>
58          <PhoneForm
59            onCreate={this.handleCreate} />
60          <p>
61            <input
62              placeholder="검색 할 이름을 입력하세요.."
63              onChange={this.handleChange}
64              value={this.state.keyword}
65            />
66          </p>
```

React Project - Demo

- 데이터 필터링
  - 부모 컴포넌트가 렌더링 되면, 자식 컴포넌트도 렌더링 됨
  - 데이터(내용)이 변경될 경우에만 렌더링 하도록 수정

```
11  shouldComponentUpdate(nextProps, nextState) {
12    return nextProps.data !== this.props.data;
13  }
```

PhoneInfoList.js

React Project - Demo

- 데이터 필터링
  - 필터링 데이터 처리

```
55    render() {
56      const { contacts, keyword } = this.state;
57      const filteredList = contacts.filter(
58        info => info.name.indexOf(keyword) !== -1
59      );
60
61      return (
62        <div>
63          <PhoneForm
64            onCreate={this.handleCreate} />
65          <p>
66            <input
67              placeholder="검색 할 이름을 입력하세요.."
68              onChange={this.handleChange}
69              value={this.state.keyword}
70            />
71          </p>
```

```
72            <PhoneInfoList
73              data={filteredList}
74              onRemove={this.handleRemove}
75              onUpdate={this.handleUpdate}
76            />
77          </div>
78        );
79      }
```

App.js

React Project - Demo

- 렌더링 최적화
  - PhoneInfo.js

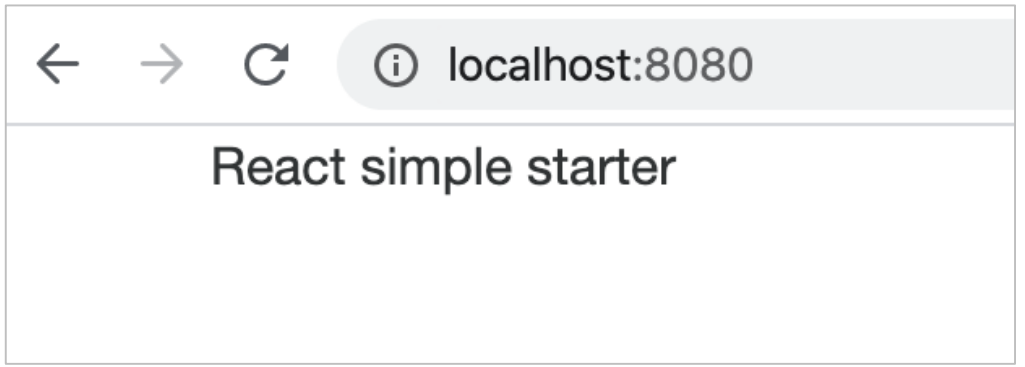
```
69  shouldComponentUpdate(nextProps, nextState) {
70    // 수정 상태가 아니고, info 값이 같다면 리렌더링 안함
71    if (!this.state.editing
72        && !nextState.editing
73        && nextProps.info === this.props.info) {
74      return false;
75    }
76    // 나머지 경우엔 리렌더링함
77    return true;
78  }
```

PhoneInfo.js

Simple Youtube App

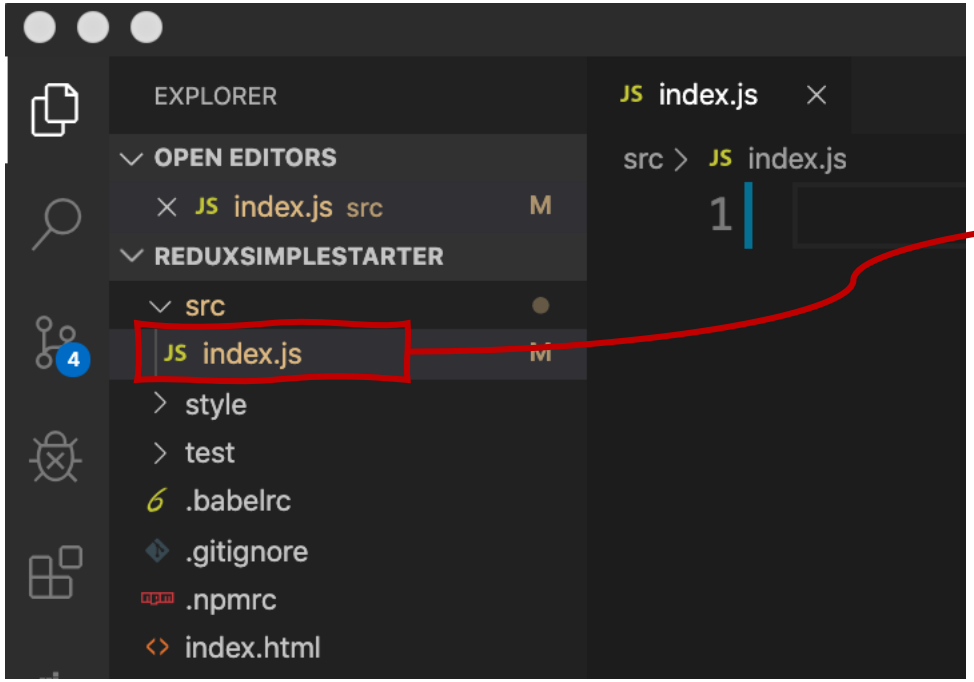
- git clone ~~
- npm install
- npm start

```
npm start

> redux-simple-starter@1.0.0 start /Users/dowon/Desktop/Work/ReduxSimpleStarter
> node ./node_modules/webpack-dev-server/bin/webpack-dev-server.js

 http://localhost:8080/webpack-dev-server/
webpack result is served from /
content is served from ./
404s will fallback to /index.html
Hash: 5f9768af15edaafa1b31
Version: webpack 1.15.0
Time: 1530ms
    Asset     Size  Chunks             Chunk Names
bundle.js  917 kB       0  [emitted]  main
chunk    {0} bundle.js (main) 890 kB [rendered]
    [0] multi main 28 bytes {0} [built]
    [1] ./src/index.js 854 bytes {0} [built]
    [2] ./~/react/index.js 190 bytes {0} [built]
    [3] ./~/process/browser.js 5.42 kB {0} [built]
```



Simple Youtube App

- 프로젝트 파일 초기화, index.js 생성



Simple Youtube App

- index.js 실행

index.js index.html

src > index.js > ...

```
1  import React from 'react';
2  import ReactDOM from 'react-dom';
3
4  const App = function() {
5    return <div>Hi!</div>
6  }
7
8  ReactDOM.render(<App />, document.querySelector('.container'));
```

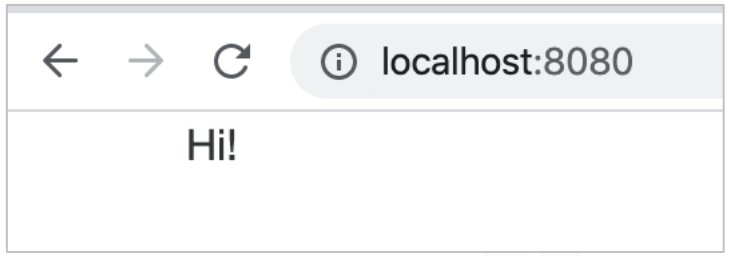
index.js

index.js index.html

index.html > html > script

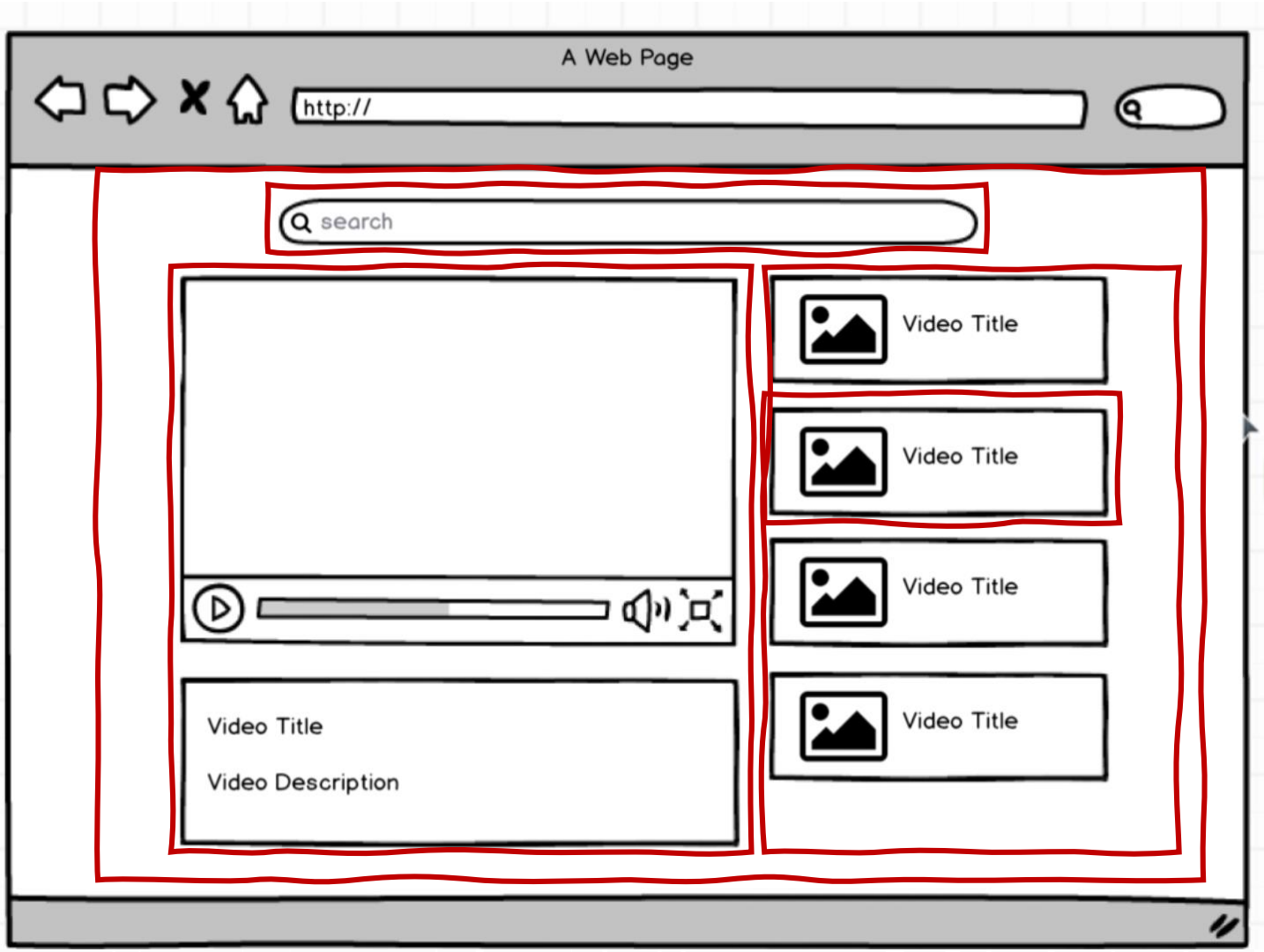
```
1  <!DOCTYPE html>
2  <html>
3    <head>
9    <body>
10     <div class="container"></div>
11   </body>
12   <script src="/bundle.js"></script>
13 </html>
```

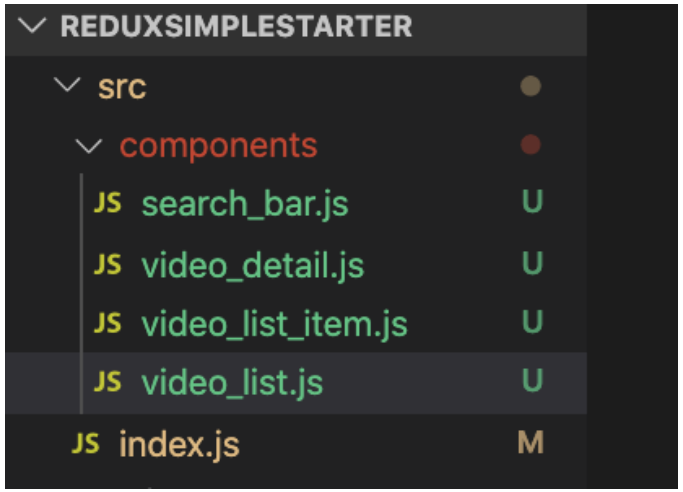
index.html



Simple Youtube App

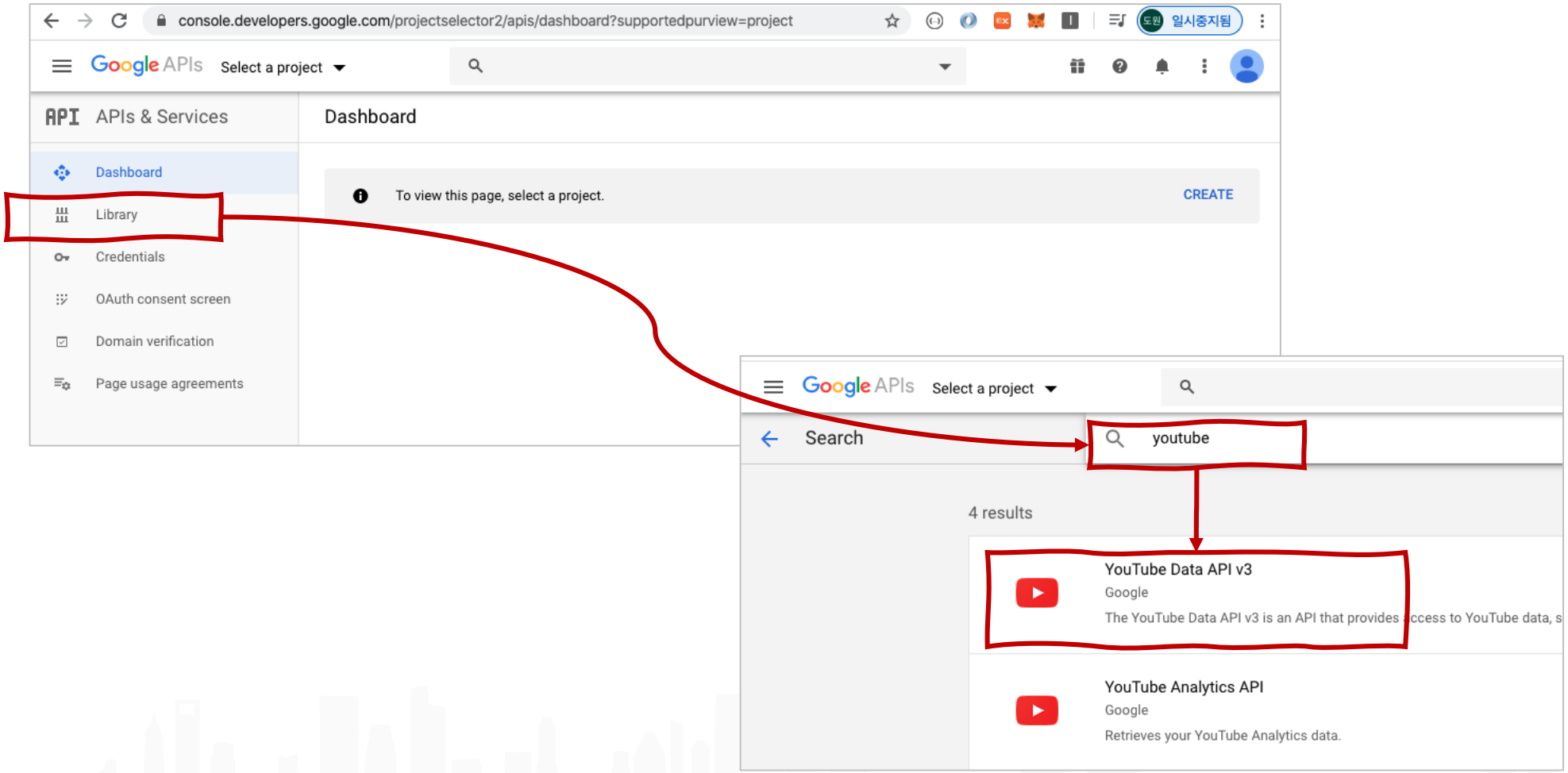
- 전체 컴포넌트 구조





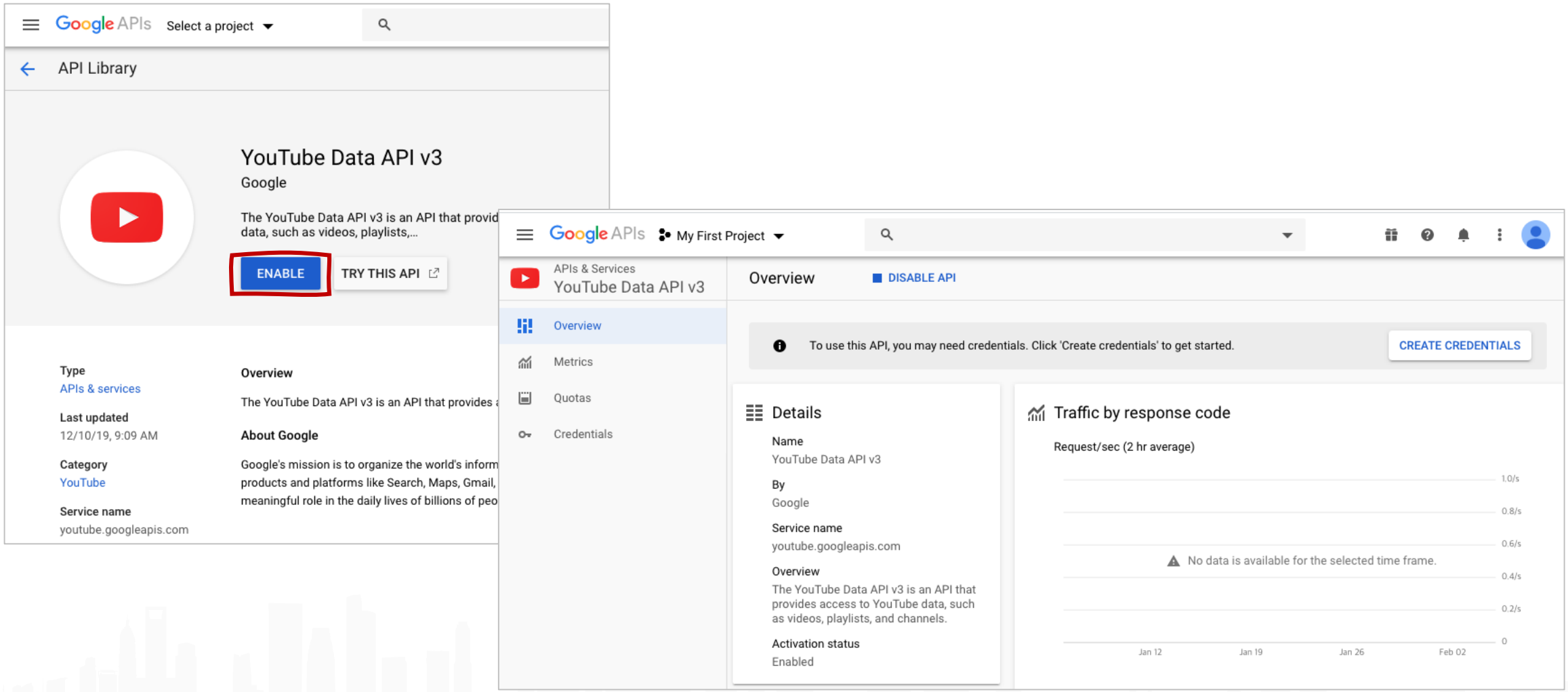
Simple Youtube App

- 구글 개발자 회원 로그인 (https://console.developers.google.com/)



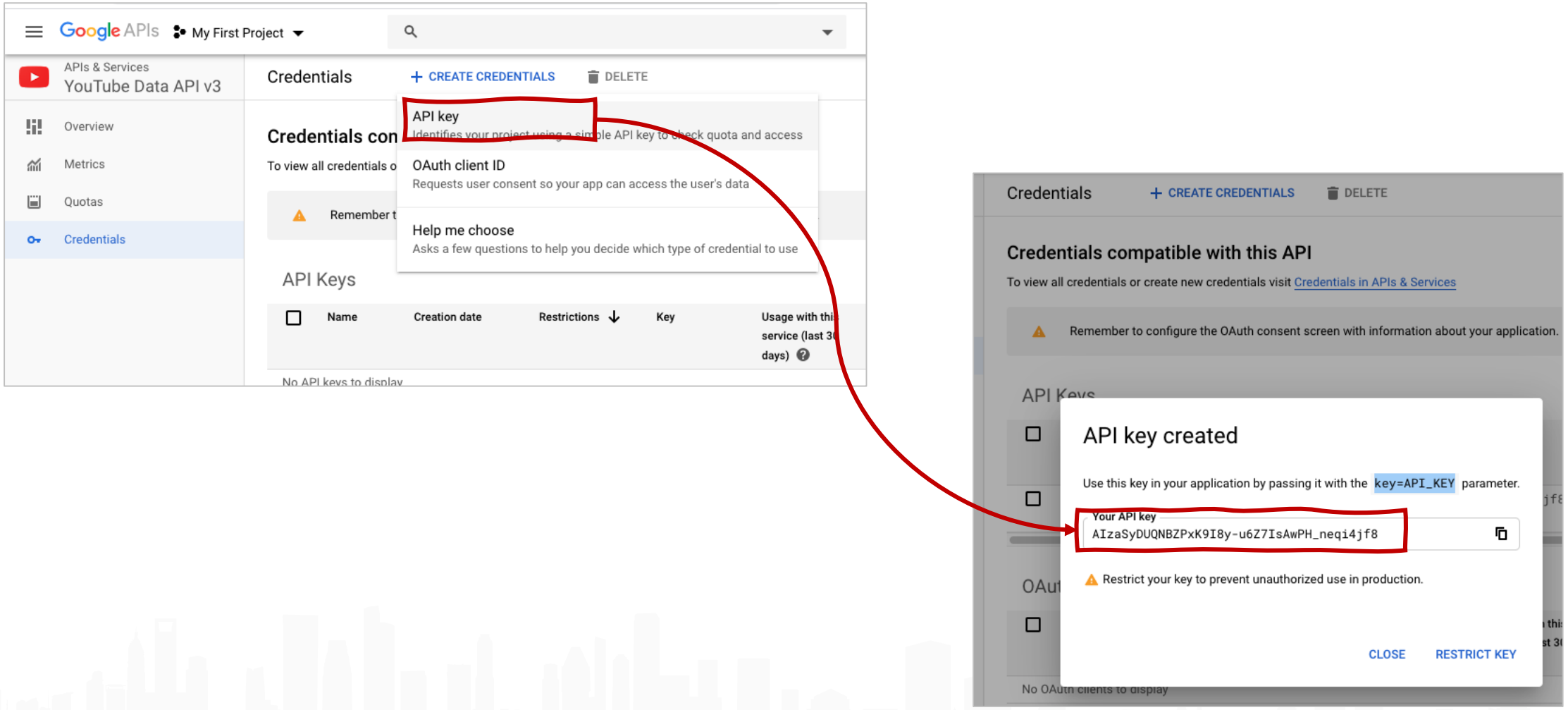
Simple Youtube App

- YouTube Data API v3



Simple Youtube App

- YouTube Data API v3



Simple Youtube App

- index.js → API_KEY 설정

```
index.js  search_bar.js  video_detail.js  video_list_item.js  video_list.js  index.html
src > index.js > ...
1  import React from 'react';
2  import ReactDOM from 'react-dom';
3
4  const API_KEY = 'AIzaSyDUQNBZPxK9I8y-u6Z7IsAwPH_neqi4jf8';
5
6  const App = () => {
7    return <div>Hi!</div>
8  }
9
10 ReactDOM.render(<App />, document.querySelector('.container'));
```

index.js

- YouTube search library 설치

```
dowon@DOWON-MacBook ~/Desktop/Work/ReduxSimpleStarter master
npm install --save youtube-api-search
+ youtube-api-search@0.0.5
added 2 packages in 3.368s
dowon@DOWON-MacBook ~/Desktop/Work/ReduxSimpleStarter master
```

Simple Youtube App

- search_bar.js

index.js | search_bar.js | video_detail.js

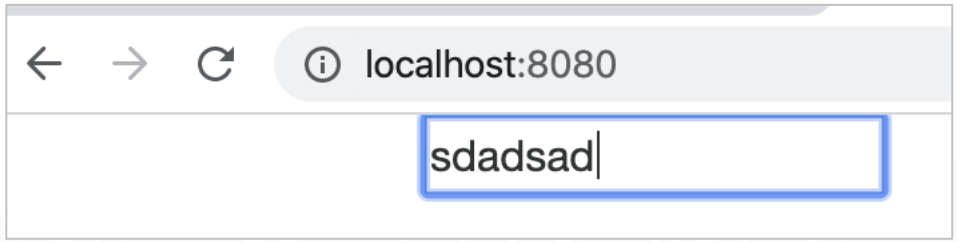
src > components > search_bar.js > ...

```
1  import React from 'react';
2
3  const SearchBar = () => {
4      return <input />;
5  }
6
7  export default SearchBar;
8
```

search_bar.js

```
4  import SearchBar from './components/search_bar';
5
6  const API_KEY = 'AIzaSyDUQNBZPxK9I8y-u6Z7IsAwPH_neqi4jf8';
7
8  const App = () => {
9    return (
10     <div>
11       <SearchBar />
12     </div>
13   );
14 }
```

index.js



Simple Youtube App

- search_bar.js → class component로 변경

```
1  import React, { Component } from 'react';
2
3  class SearchBar extends Component {
4    render() {
5      return (
6        <input />
7      )
8    }
9  }
10
11 export default SearchBar;
```

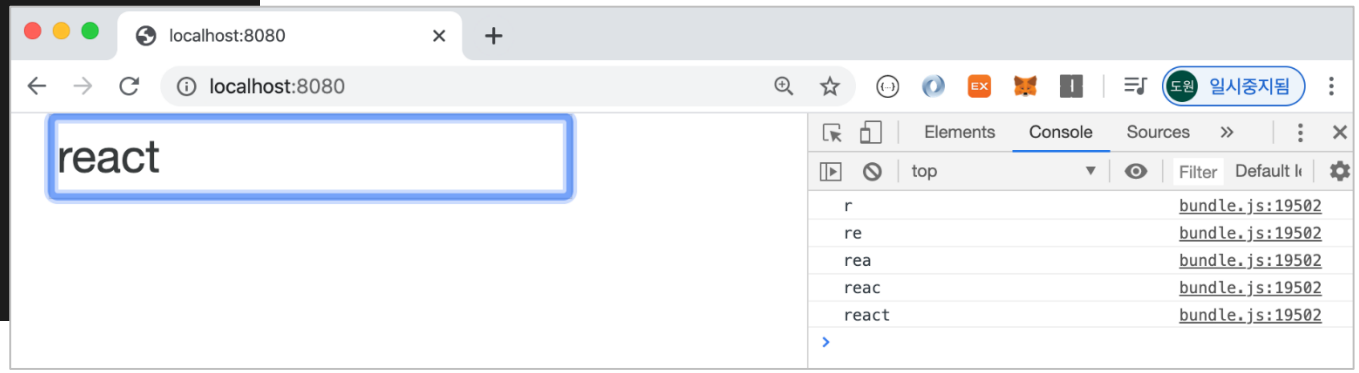
search_bar.js

Simple Youtube App

- search_bar.js → event handler 추가

```
3  class SearchBar extends Component {
4    render() {
5      return (
6        <input onChange={this.onInputChange} />
7      )
8    }
9
10   onInputChange(event) {
11     console.log(event.target.value);
12   }
13 }
```

search_bar.js




```
render() {
  return (
    <input onChange={event => console.log(event.target.value)} />
  )
}
```

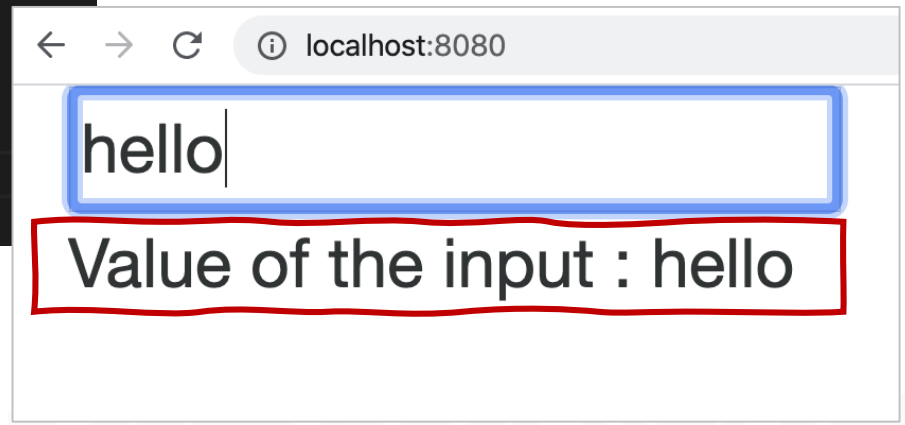
→ event handler를 간단하게

Simple Youtube App

- state를 이용한 검색기능

```
class SearchBar extends Component {
  constructor(props) {
    super(props);

    this.state = { term: '' };
  }

  render() {
      return (
        <div>
          <input
            value={this.state.term}
            onChange={event => this.setState({ term: event.target.value })} />
        </div>
      )
  }

}
```

search_bar.js

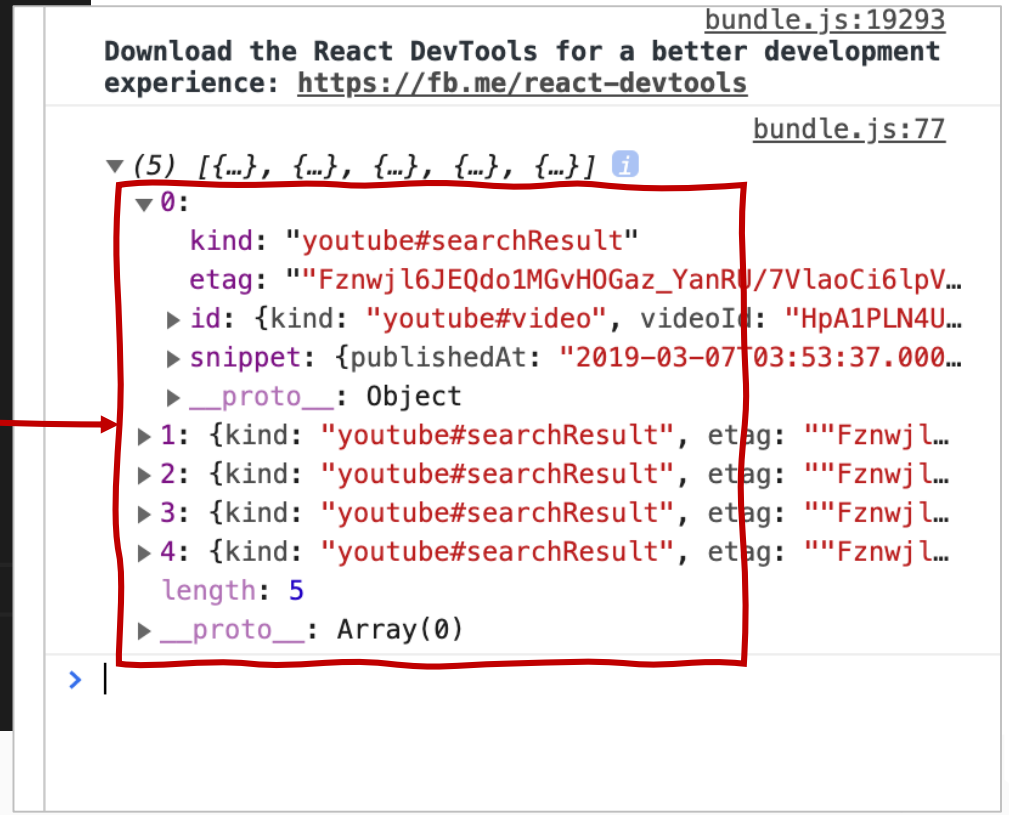


Simple Youtube App

- YouTube 검색 반환

```
1  import React from 'react';
2  import ReactDOM from 'react-dom';
3  import YTSearch from 'youtube-api-search';
4
5  import SearchBar from './components/search_bar';
6
7  const API_KEY = 'AIzaSyDUQNBZPxK9I8y-u6Z7IsAwPH_neqi4jf8';
8
9  YTSearch({key: API_KEY, term: 'surfboards'}, function(data) {
10   console.log(data);
11 });
12
13 const App = () => {
14   return (
15     <div>
16       <SearchBar />
17     </div>
18   );
19 }
```

index.js



Simple Youtube App

- index.js → class component

```
13  class App extends Component {
14    render() {
15      return (
16        <div>
17          <SearchBar />
18        </div>
19      );
20    }
21  }
```

index.js

- TO-DO
  - 생성자 함수 추가
    - videos: [Array] 을 state에서 초기화
  - YTSearch 함수를 생성자 함수로 이동
  - YTSearch 함수에서 검색 된 videos의 목록을 state에 업데이트

Simple Youtube App

- index.js → class component

```
 9  class App extends Component {
10    constructor(pros) {
11      super(pros);
12      this.state = {
13          videos: [],
14      };
15
16      YTSearch({key: API_KEY, term: 'surfboards'}, (videos) => {
17        this.setState({
18          videos: videos
19        });
20      });
21    }
```

index.js

Simple Youtube App

- video_list.js

```
1  import React from 'react';
2
3  const VideoList = () => {
4    return (
5      <ul className="col-md-4 list-group">
6      </ul>
7    )
8  }
9
10 export default VideoList;
```

video_list.js

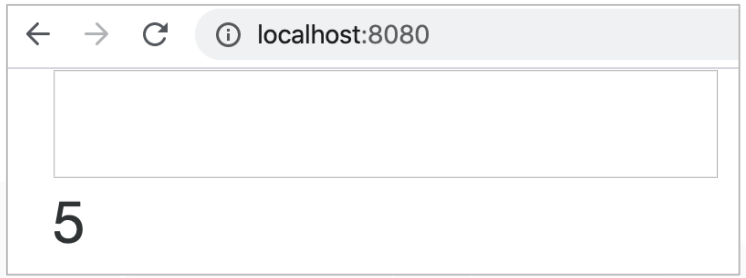
```
6  import VideoList from './components/video_list';
7
8  const API_KEY = 'AIzaSyDUQNBZPxK9I8y-u6Z7IsAwPH_neqi4jf8';
9
10 class App extends Component {
11   constructor(pros) {
```

```
24   render() {
25     return (
26       <div>
27         <SearchBar />
28         <VideoList videos={this.state.videos} />
29       </div>
30     );
31   }
```

index.js

video 개수 출력?

```
3  const VideoList = (props) => {
4    return (
5      <ul className="col-md-4 list-group">
6        {props.videos.length}
7      </ul>
8    )
9  }
```

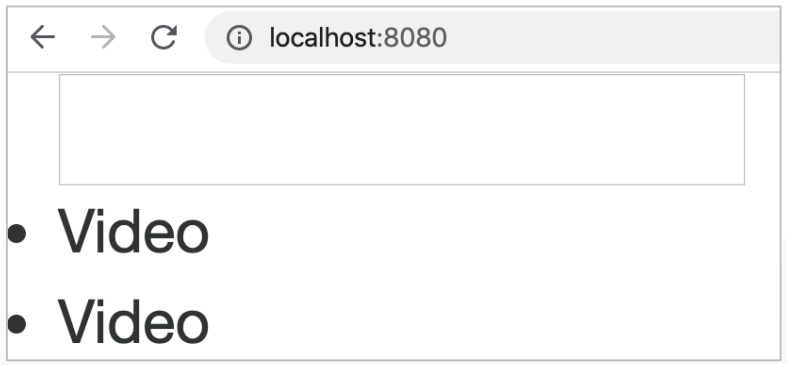


Simple Youtube App

- video_list_item.js

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

video_list_item.js

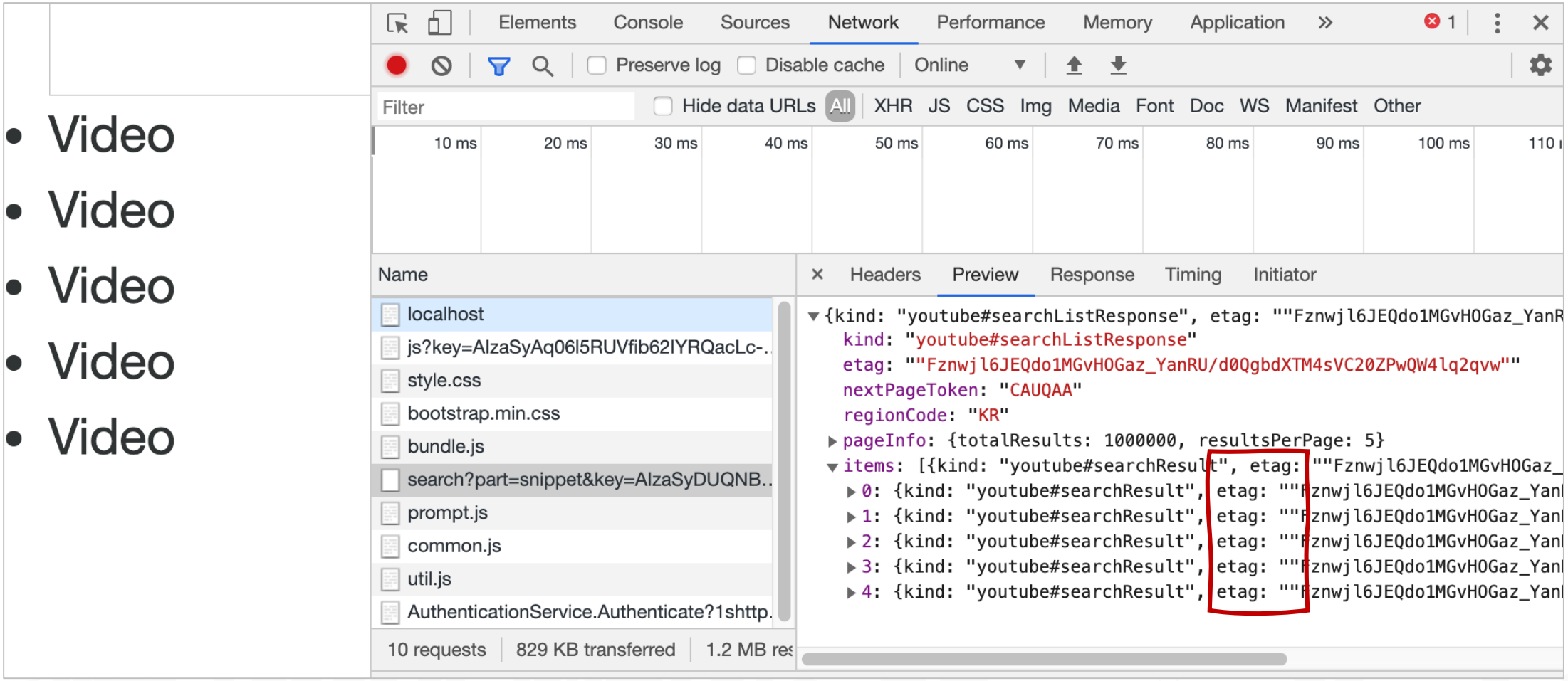



```
2  import VideoListItem from './video_list_item';
3
4  const VideoList = (props) => {
5    const videoItems = props.videos.map((video) => {
6      return (
7          <VideoListItem
8              video={video} />
9      );
10   });
11
12   return (
13     <ul className="col-md-4 list-group">
14       {videoItems}
15     </ul>
16   )
17 }
```

video_list.js

Simple Youtube App

- video_list.js → 각 item에 key 설정, 각 video의 etag를 key로 사용

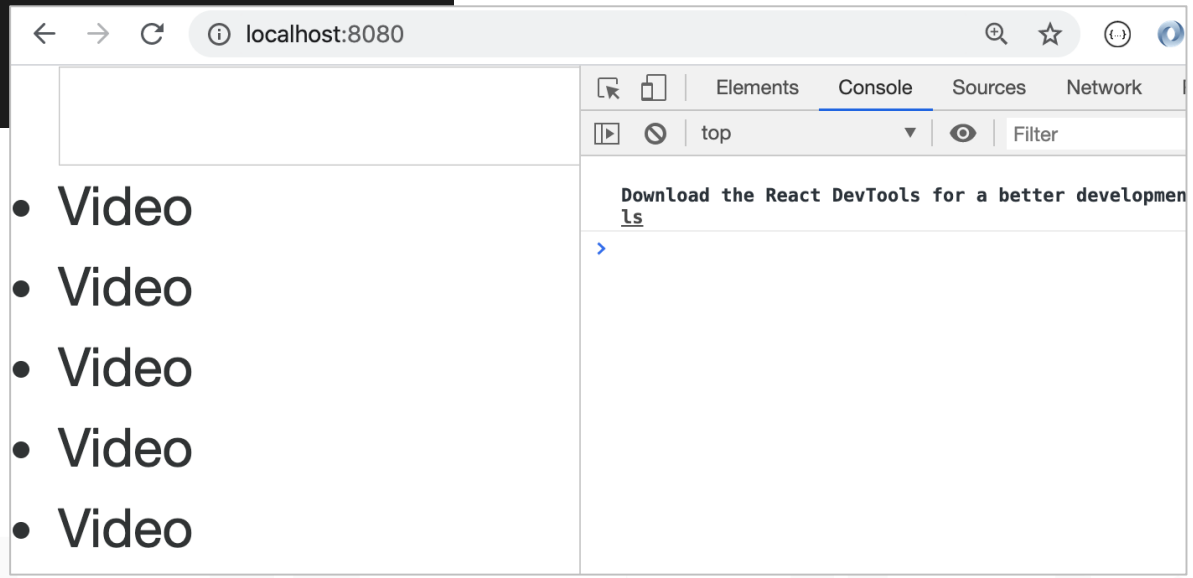


Simple Youtube App

- video_list.js → 각 item에 key 설정, 각 video의 etag를 key로 사용

```
4  const VideoList = (props) => {
5    const videoItems = props.videos.map((video) => {
6      return (
7          <VideoListItem
8              key={video.etag}
9              video={video} />
10     );
11   });
```

video_list.js

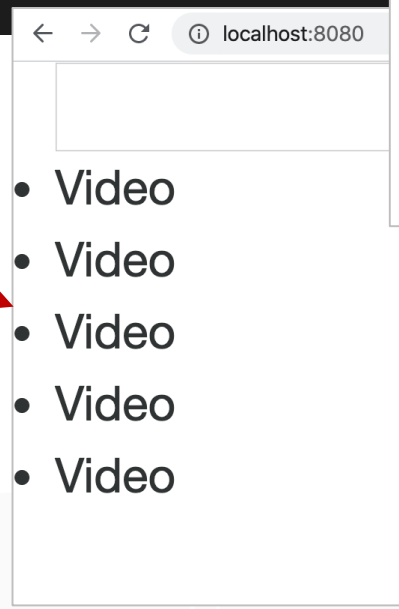


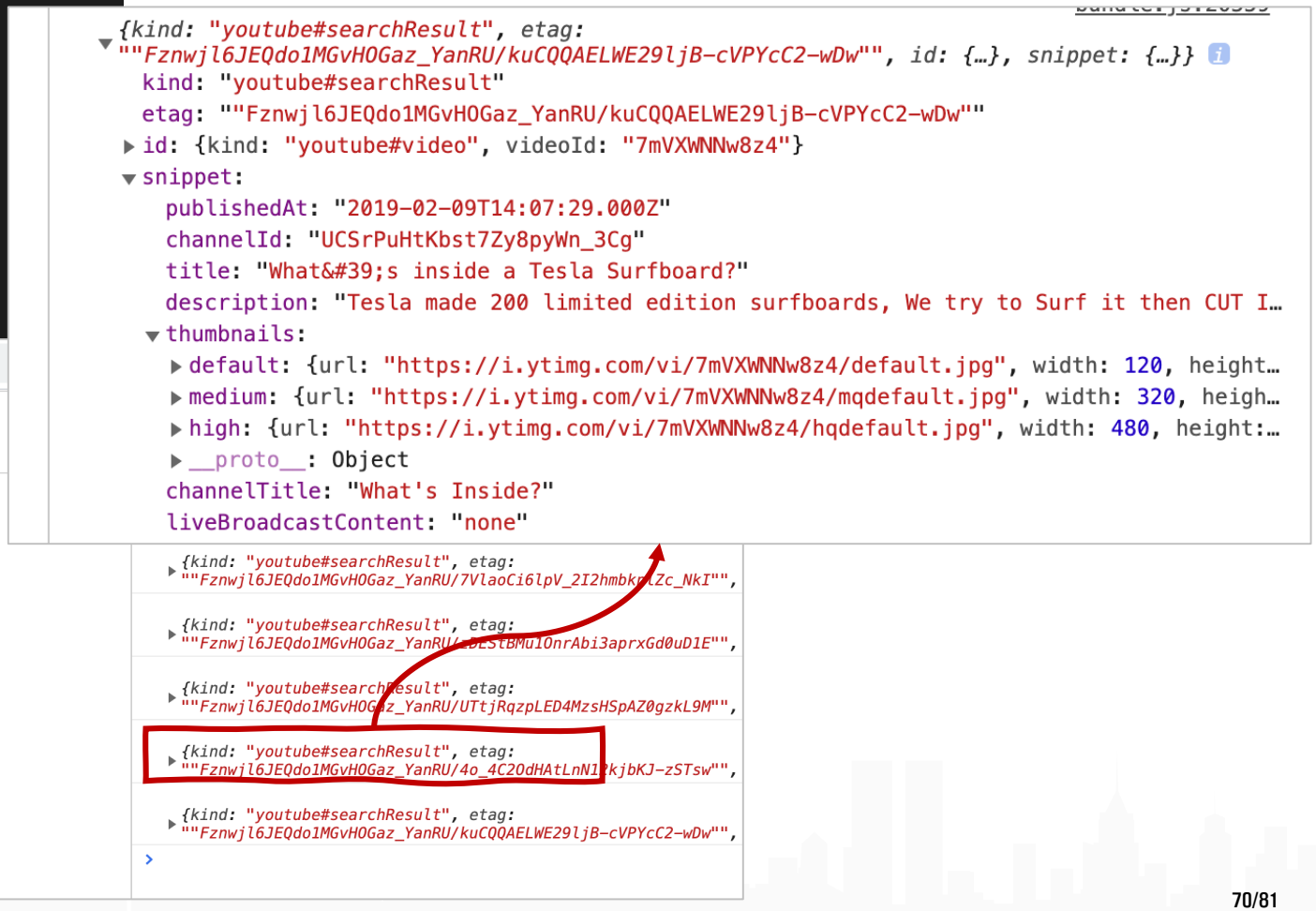
Simple Youtube App

- video_list_item.js → 전달되는 video의 정보 상세 표시

```
3  const VideoListItem = ({video}) => {
4    console.log(video);
5      return (
6          <li>
7            Video
8          </li>
9      )
10 };
```

video_list_item.js



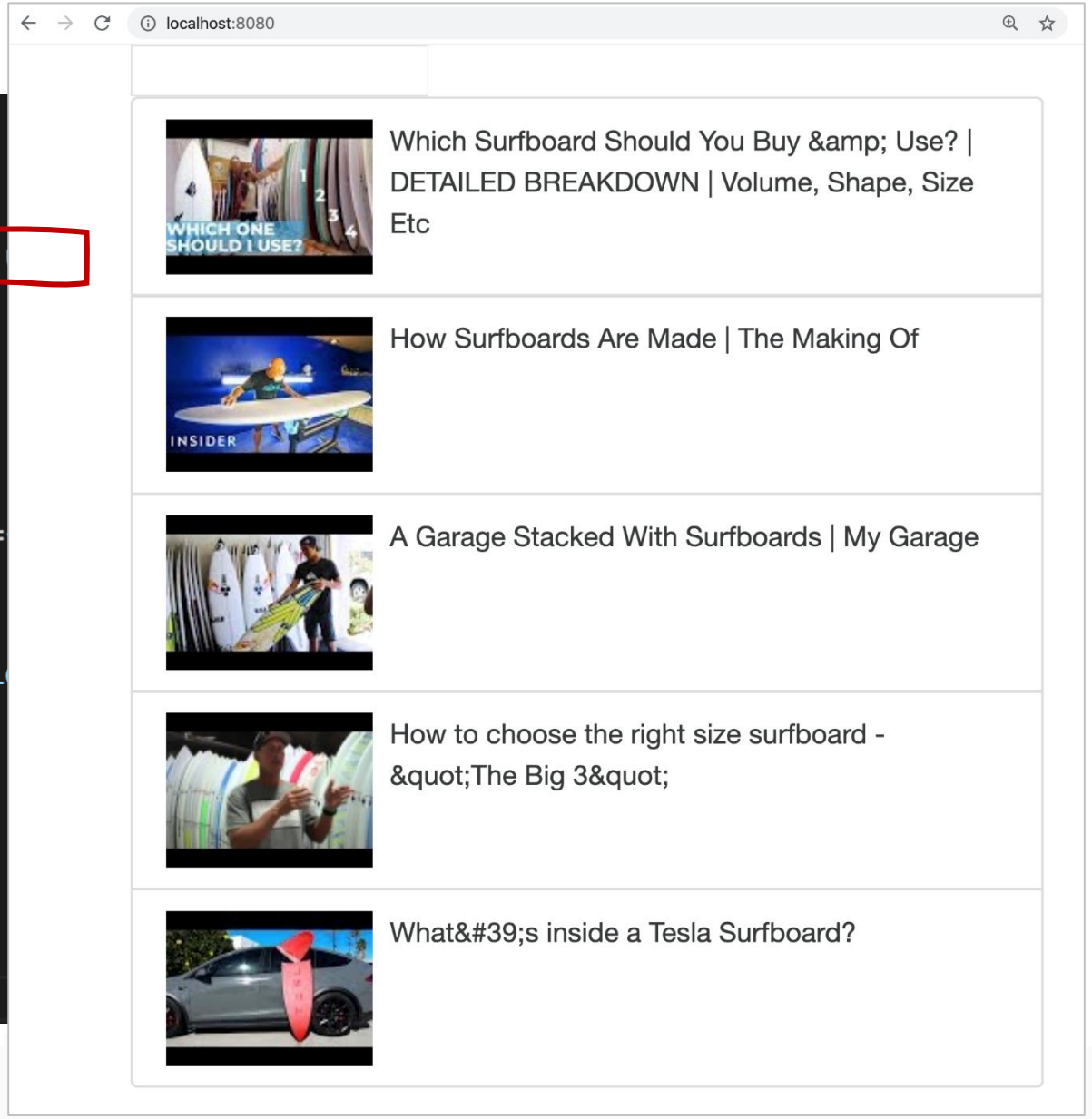


Simple Youtube App

- video_list_item.js → 스타일 적용

```
1  import React from 'react';
2
3  const VideoListItem = ({video}) => {
4    const imageUrl = video.snippet.thumbnails.default.
5
6    return (
7      <li className="list-group-item">
8        <div className="video-list media">
9          <div className="media-left">
10           <img className="media-object" src=
11         </div>
12         <div className="media-body">
13           <div className="media-heading">{vi
14         </div>
15       </div>
16     </li>
17   )
18 };
19
20 export default VideoListItem;
```

video_list_item.js



Simple Youtube App

- video_detail.js

```
1  import React from 'react';
2
3  const VideoDetail = ({video}) => {
4      const videoId = video.id.videoId;
5      // const url = 'https://www.youtube.com/embed/' + videoId;
6      const url = `https://www.youtube.com/embed/${videoId}`;
7
8      return (
9          <div className="video-detail col-md-8">
10             <div className="embed-responsive embed-responsive-16by9">
11                 <iframe className="embed-reponsive-item" src={url}></iframe>
12             </div>
13             <div className="details">
14                 <div>{video.snippet.title}</div>
15                 <div>{video.snippet.description}</div>
16             </div>
17         </div>
18     );
19 };
20
21 export default VideoDetail;
```

video_detail.js

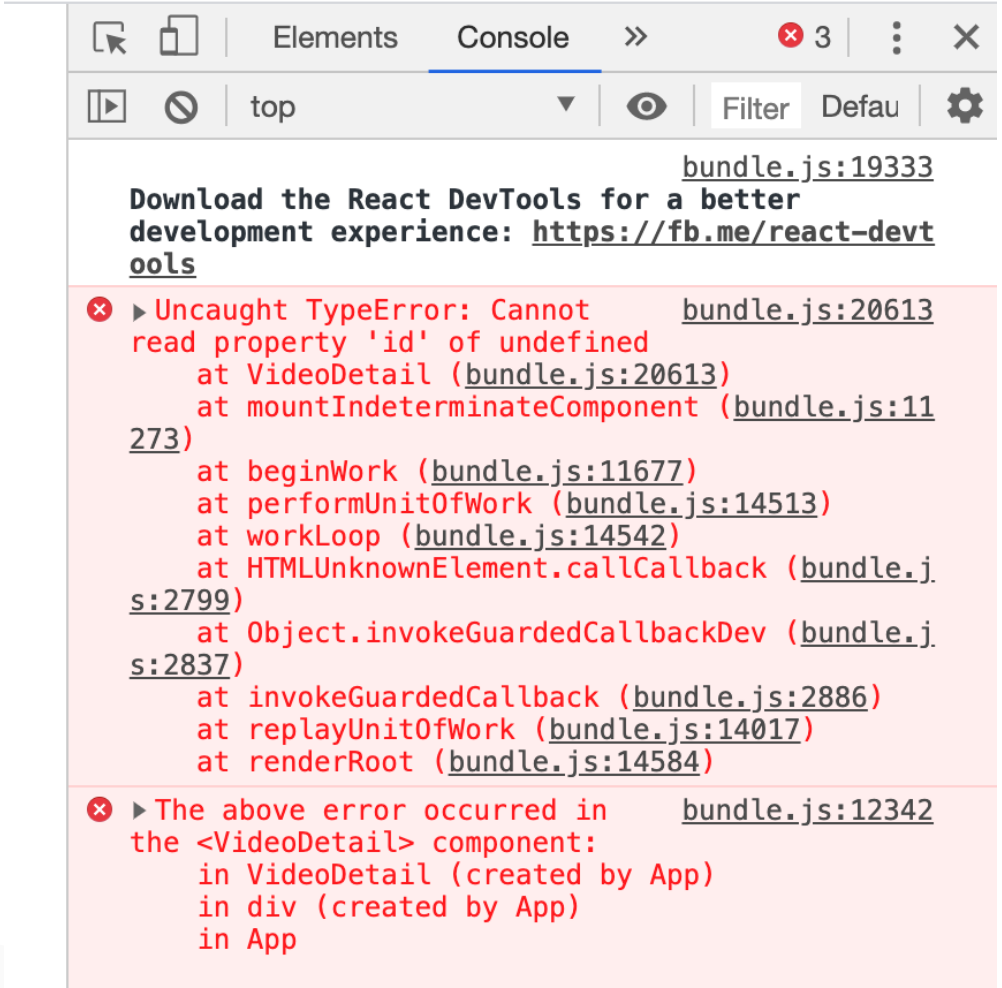
Simple Youtube App

- video_detail.js

```
5  import SearchBar from './components/search_bar';
6  import VideoList from './components/video_list';
7  import VideoDetail from './components/video_detail';
8
9  const API_KEY = 'AIzaSyDUQNBZPxK9I8y-u6Z7IsAwPH_neqi4jf8';
10
11 class App extends Component {
```

```
25   render() {
26     return (
27       <div>
28         <SearchBar />
29         <VideoDetail video={this.state.videos[0]} />
30         <VideoList videos={this.state.videos} />
31       </div>
32     );
33   }
34 }
```

index.js

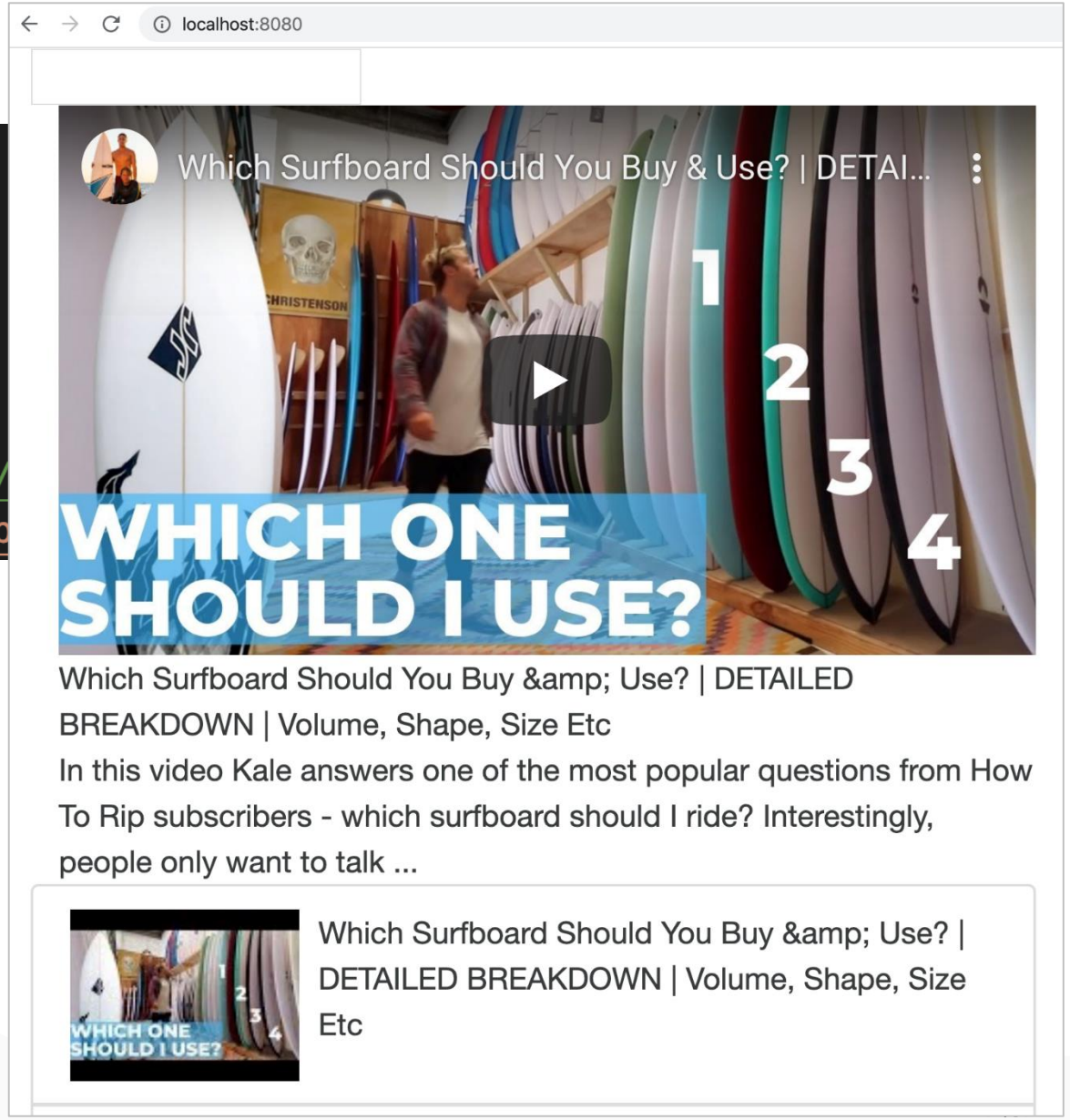


Simple Youtube App

- video_detail.js → 초기 검색어 처리

```
3  const VideoDetail = ({video}) => {
4      if (!video) {
5          return <div>Loading...</div>;
6      }
7
8      const videoId = video.id.videoId;
9      // const url = 'https://www.youtube.com/
10     const url = `https://www.youtube.com/emb
```

video_detail.js



Simple Youtube App

- index.js → 선택 된 video item을 detail에 전달

```
11  class App extends Component {
12    constructor(pros) {
13      super(pros);
14      this.state = {
15          videos: [],
16          [redacted]
17      };
18
19      YTSearch({key: API_KEY, term: 'surfboards'}, (videos) => {
20        this.setState({
21          videos: videos,
22          [redacted]
23        });
24      });
25    }
26
27    render() {
28      return (
29        <div>
30            <SearchBar />
31            <VideoDetail video=[redacted] />
32            <VideoList videos={this.state.videos} />
33        </div>
```

index.js

Simple Youtube App

- index.js → 선택 된 video item을 detail에 전달

```
4  const VideoList = (props) => {
5    const videoItems = props.videos.map((video) => {
6      return (
7          <VideoListItem
8              onVideoSelect={props.onVideoSelect}
9              key={video.etag}
10             video={video} />
11     );
12   });
```

video_list.js

```
3  const VideoListItem = ({video, onVideoSelect}) => {
4    const imageUrl = video.snippet.thumbnails.default.url;
5
6    return (
7        <li onClick={() => onVideoSelect(video)} className="list-group-item">
8            <div className="video-list media">
9                <div className="media-left">
10                   <img className="media-object" src={imageUrl} />
```

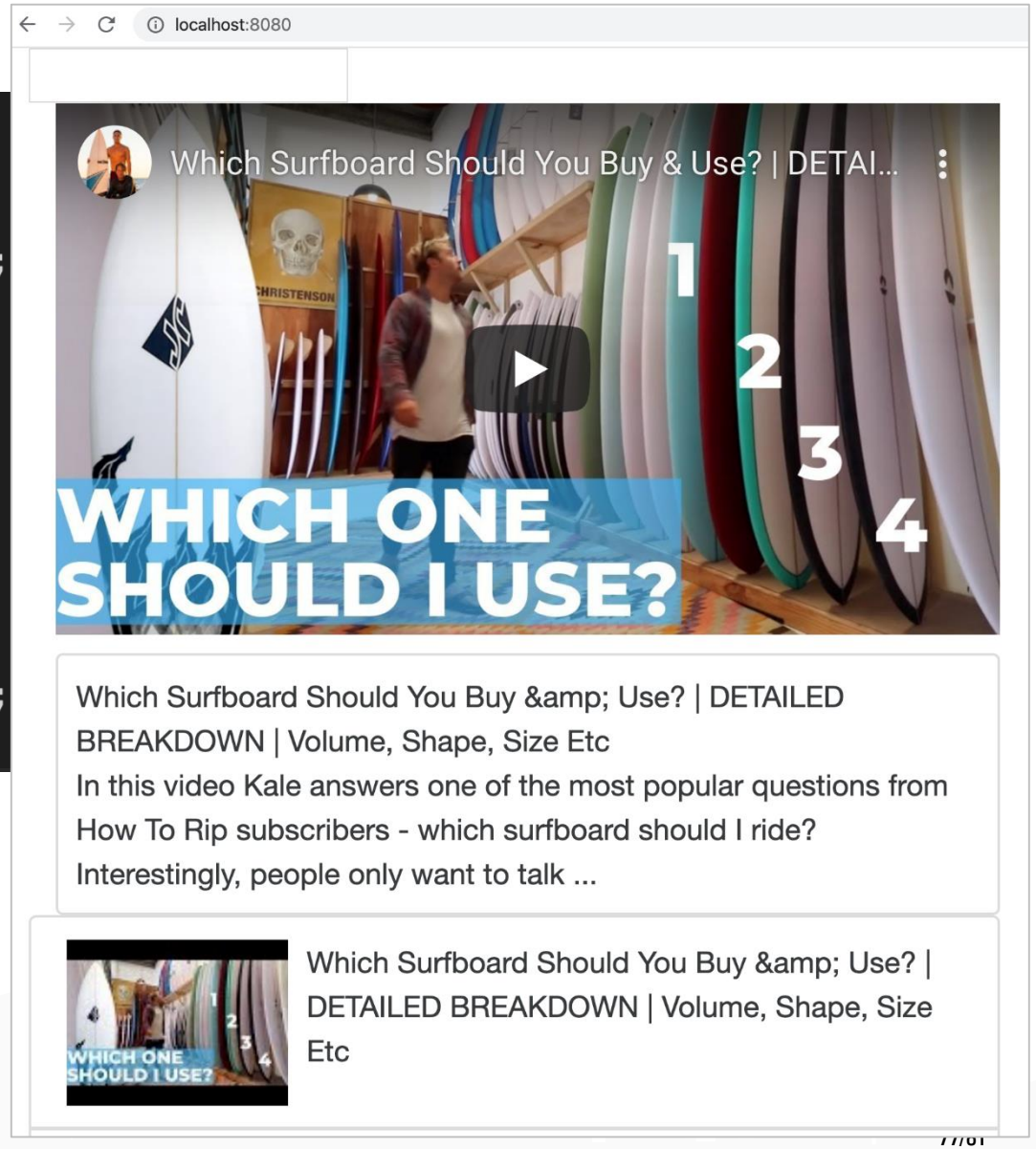
video_list_item.js

Simple Youtube App

- css 생성

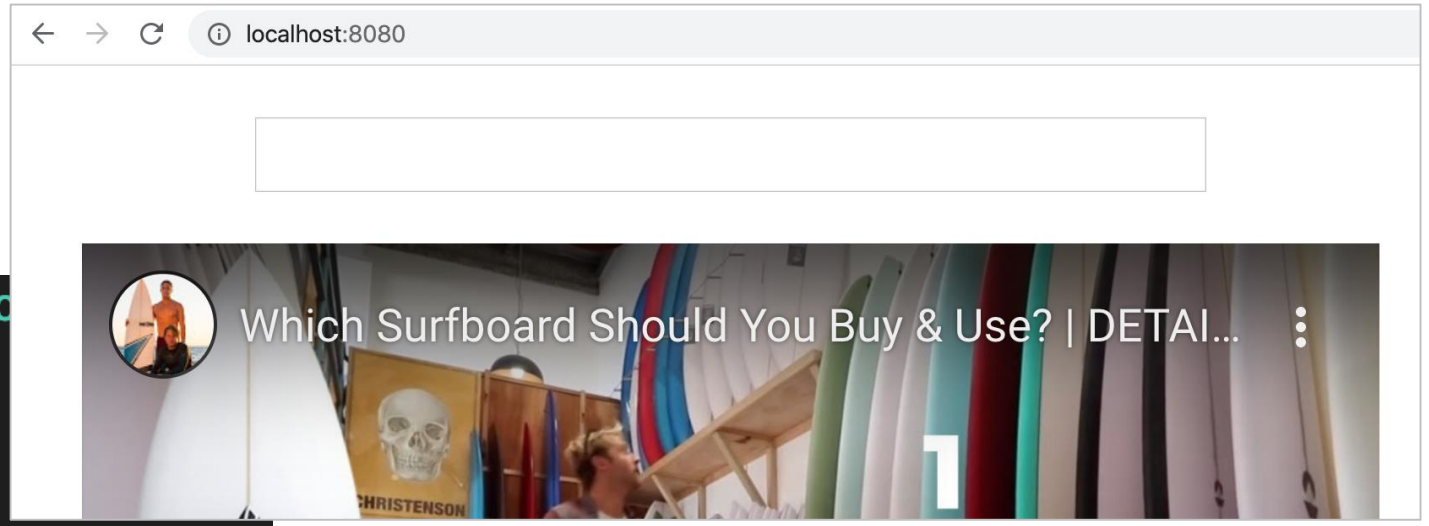
```css
.search-bar {
  margin: 20px;
  text-align: center;
}

.search-bar input {
  width: 75%;
}

.video-item img {
  max-width: 64px;
}

.video-detail .details {
  margin-top: 10px;
  padding: 10px;
  border: 1px solid #ddd;
  border-radius: 4px;
}

.list-group-item {
  cursor: pointer;
}

.list-group-item:hover {
  background-color: #eee;
}
```

style.css



Simple Youtube App

- search-bar.js -→ style 적용




```
3  class SearchBar extends Compo
4    constructor(props) {
5      super(props);
6
7      this.state = { term: '' };
8    }
9
10   render() {
11       return (
12         <div [redacted]>
13           <input
```

search_bar.js

Simple Youtube App

- 비디오 검색 → index.js 수정

```
11 class App extends Component {
12   constructor(pros) {
13     super(pros);
14     this.state = {
15         videos: [],
16         selectedVideo: null
17     };
18
19
20   }
21
22   videoSearch(term) {
23     YTSearch({key: API_KEY, term: term}, (videos) => {
24         this.setState({
25             videos: videos,
26             selectedVideo: videos[0]
27         });
28     });
29   }
```

index.js

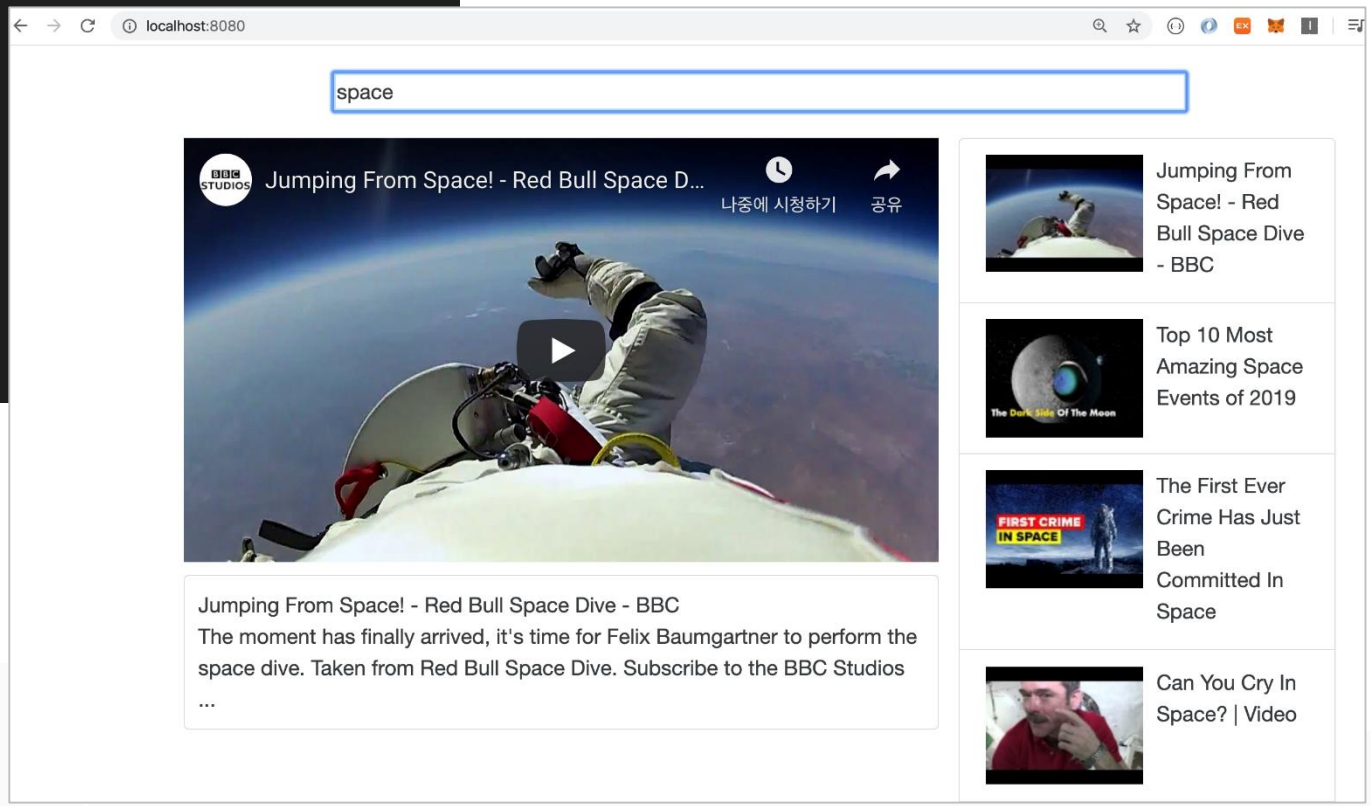
```
31   render() {
32     return (
33       <div>
34         <SearchBar onSearchTermChange={term => this.videoSearch(term)}
35         <VideoDetail video={this.state.selectedVideo} />
36         <VideoList
37           onVideoSelect={selectedVideo => this.setState({selectedVideo}
38           videos={this.state.videos} />
39       </div>
```

Simple Youtube App

- 비디오 검색 → search_bar.js 수정

```
10    render() {
11        return (
12          <div className='search-bar'>
13            <input
14              value={this.state.term}
15              onChange={event => this.onInputChange(event.target.value)} />
16          </div>
17        )
18    }
19
20    onInputChange(term) {
21      this.setState({term});
22      this.props.onSearchTermChange(term);
23    }
24  }
```

search_bar.js



Simple Youtube App

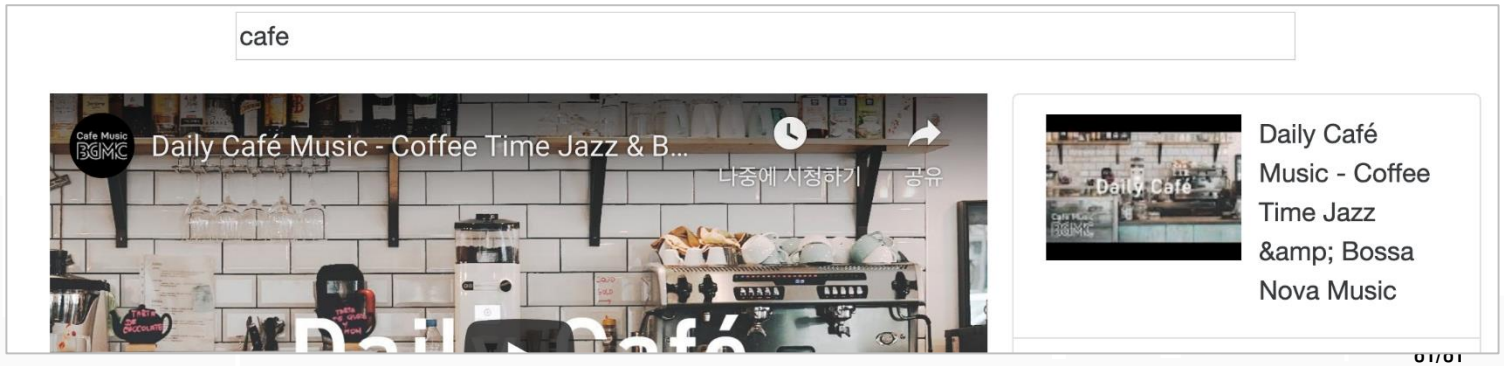
- 검색 시간 조절 → lodash, debounce

```
npm install --save lodash
+ lodash@3.10.1
updated 1 package in 2.944s
 dowon@DOWON-MacBook ~/Desktop/Work/ReduxSimpleStarter master
```

```
1  import _ from 'lodash';
2
3  import React, { Component } from 'react';
```

```
33    render() {
34      const videoSearch = _.debounce((term) => {this.videoSearch(term)}, 300);
35
36      return (
37        <div>
38          <SearchBar onSearchTermChange={videoSearch} />
```

search_bar.js



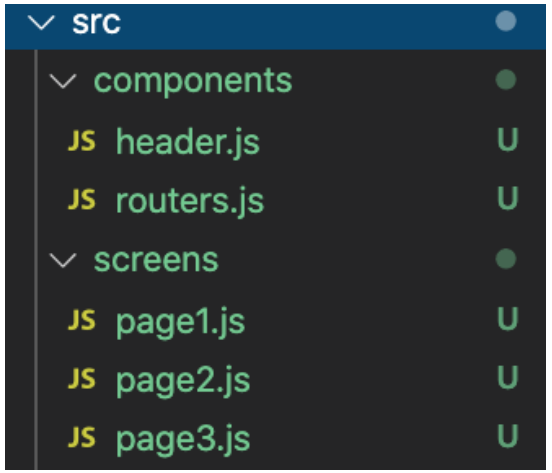
React Router

- SPA에서의 라우팅 문제를 해결하기 위해 사용되는 네비게이션 라이브러리
- 브라우저 내장 객체 사용 가능
  - location
  - history

- React Router
  - Web
  - Native
  - react-router-dom 라이브러리 필요 (React Router V4부터 필요)

React Router

- npx create-react-app react-router
  - src/components, src/routes 폴더 생성
  - src/components/routers.js, src/components/header.js 생성
  - src/routes/page1.js, src/routes/page2.js, src/routes/page3.js 생성



React Router

- components/header.js

```
1  import React from 'react';
2
3  const Header = () => (
4      <div>
5          <ul>
6              <li>Page1</li>
7              <li>Page2</li>
8              <li>Page3</li>
9          </ul>
10     </div>
11 )
12
13 export default Header;
```

- screens/page1.js

```
1  export default () => "Page1!";
```

- screens/page2.js

```
1  export default () => "Page2!";
```

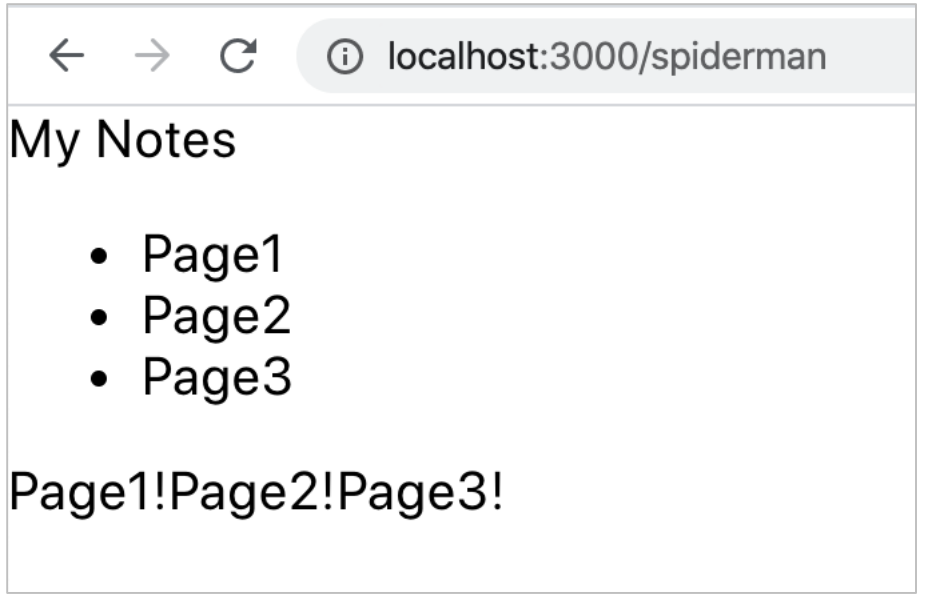
- screens/page3.js

```
1  export default () => "Page3!";
```

React Router

- /App.js

```
1  import React from 'react';
2  import Header from "./components/header";
3  import Page1 from "./screens/page1";
4  import Page2 from "./screens/page2";
5  import Page3 from "./screens/page3";
6
7  function App() {
8    return (
9      <div>
10       My Notes
11       <Header />
12       <Page1 />
13       <Page2 />
14       <Page3 />
15     </div>
16   );
17 }
18
19 export default App;
```

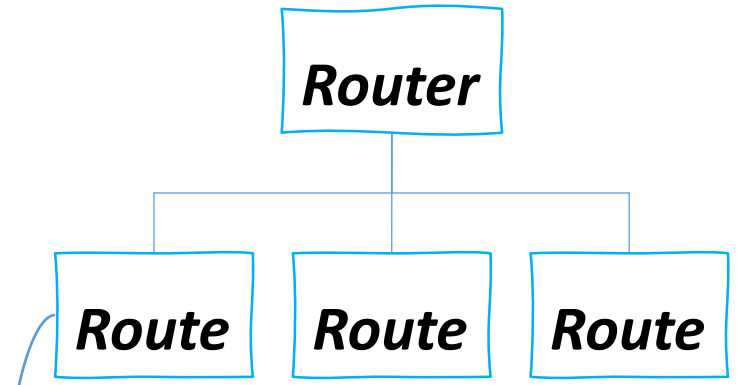


React Router

- react-router-dom 적용
  - npm install react-router-dom

- components/routers.js

```
1  import React from 'react';
2  import { BrowserRouter as Router, Route } from 'react-router-dom';
3  import Page1 from '../screens/page1';
4  import Page2 from '../screens/page2';
5  import Page3 from '../screens/page3';
6
7  export default () => {
8    return (
9      <Router>
10       <Route path="/my_note1" component={Page1} />
11       <Route path="/my_note2" component={Page2} />
12       <Route path="/my_note3" component={Page3} />
13     </Router>
14   )
15 }
```



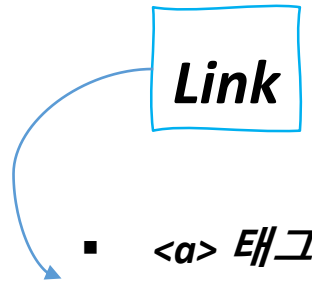
- *path*
- *component*

- *현재 주소창의 경로와 매칭될 컴포넌트 지정*

React Router

- components/header.js 파일 수정

```
1  import React from 'react';
2  import { Link } from "react-router-dom";
3
4  const Header = () => (
5      <div>
6          <li>
7              <Link to="/my_note1">My Note1</Link>
8          </li>
9          <li>
10             <Link to="/my_note2">My Note2</Link>
11         </li>
12         <li>
13             <Link to="/my_note3">My Note3</Link>
14         </li>
15     </div>
16 )
17
18 export default Header;
```



- *<a> 태그와 유사한 기능의 컴포넌트*
- *href의 속성의 의미로 to 속성 사용*

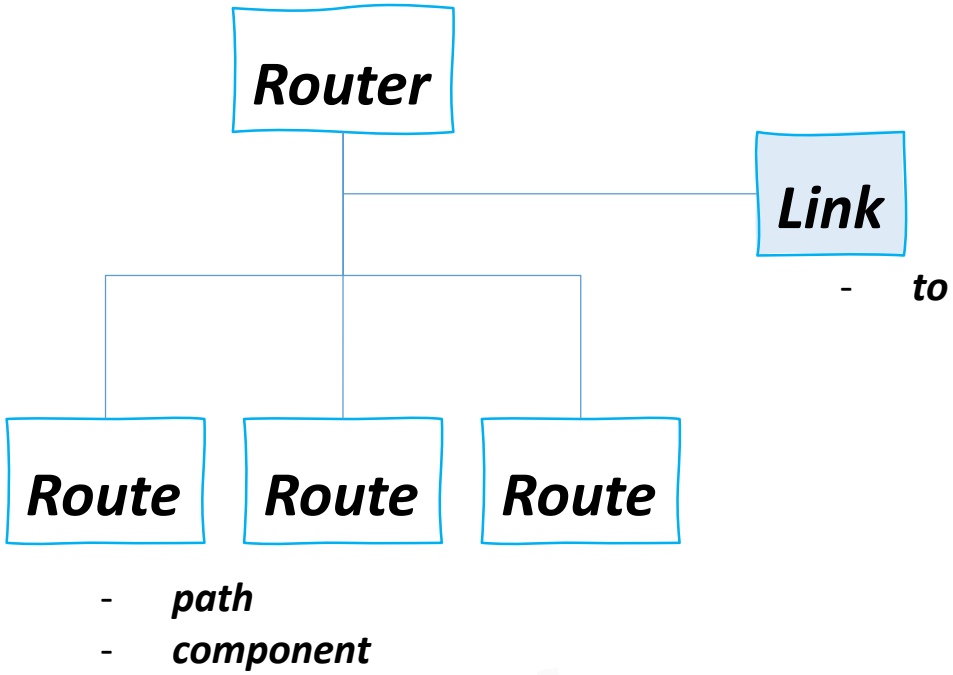
React Router

- App.js 파일 수정

```
1  import React from 'react';
2  import Header from "./components/header";
3  import Routes from "./components/routers";
4
5  function App() {
6    return (
7      <div>
8        My Notes
9        <Header />
10       <Routes />
11     </div>
12   );
13 }
14
15 export default App;
```

React Router

- App.js 파일 수정
  - <Header />를 routers.js 페이지로 이동




```
1  import React from 'react';
2  import { BrowserRouter as Router, Route } from 'react-router-dom';
3  import Page1 from '../screens/page1';
4  import Page2 from '../screens/page2';
5  import Page3 from '../screens/page3';
6  import Header from './header';
7
8  export default () => {
9    return (
10     <Router>
11       <Header />
12       <Route path="/my_note1" component={Page1} />
13       <Route path="/my_note2" component={Page2} />
14       <Route path="/my_note3" component={Page3} />
15     </Router>
16   )
17 }
```

React Router

- 결과