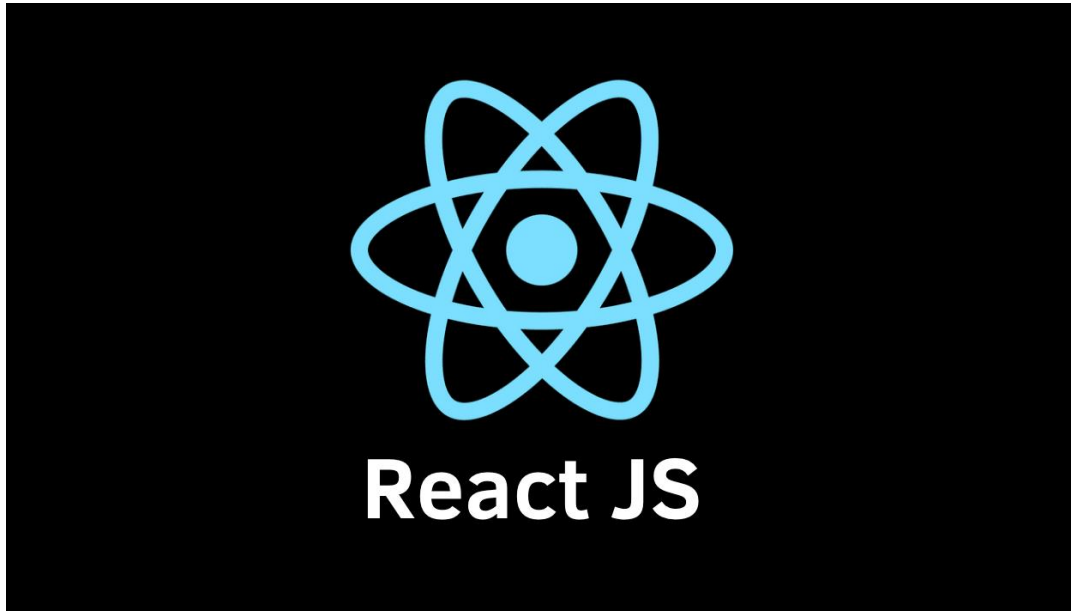
프론트 세미나 Week7
ReactJS (1)

Review

Last lecture: 백엔드와의 연동 – Form tag & Ajax

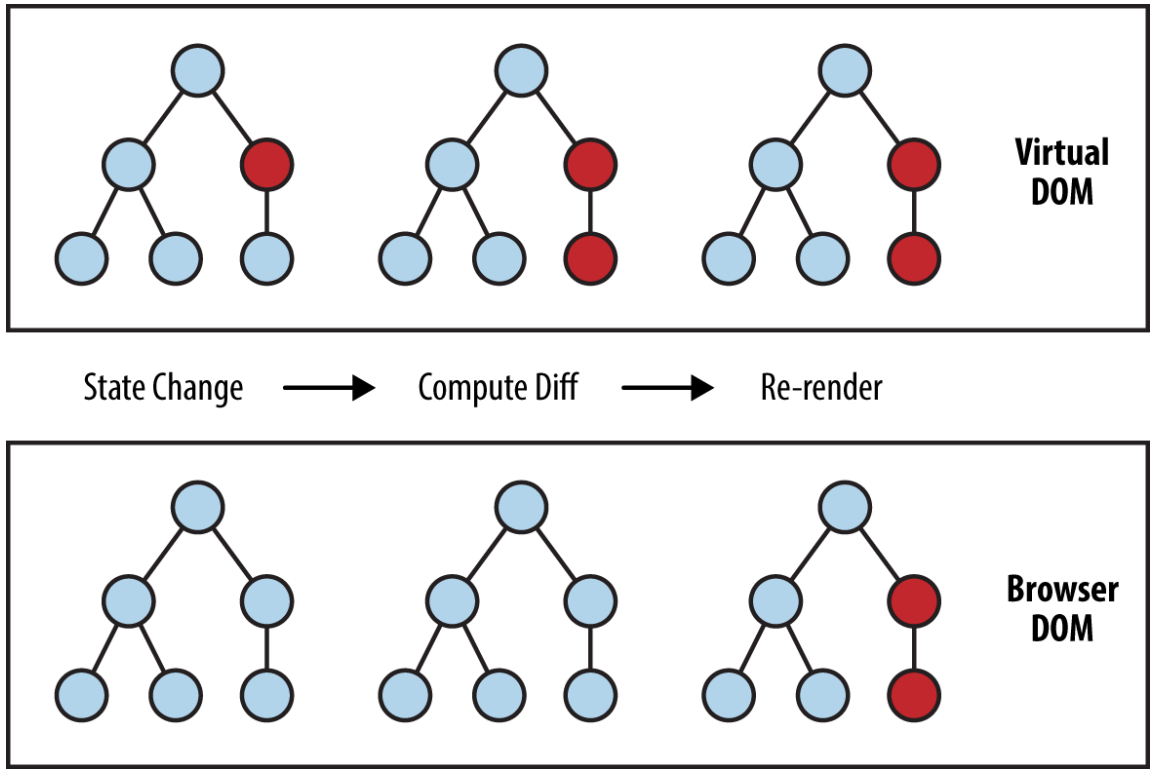
- 지금까지 배운 내용으로 우리는 HTML, CSS, JS를 활용하여 정적인 웹페이지를 구현할 수 있을 뿐만 아니라 백엔드와의 기본적인 협업도 가능하게 되었습니다!
- 하지만 지금까지 배운 기본적인 것으로는 프로그램의 유지보수가 어려울 수 있습니다!
- 마지막 2주차 동안은 프론트 라이브러리 중 하나인 ReactJS에 대해서 간략하게 배워볼 것입니다!

ReactJS



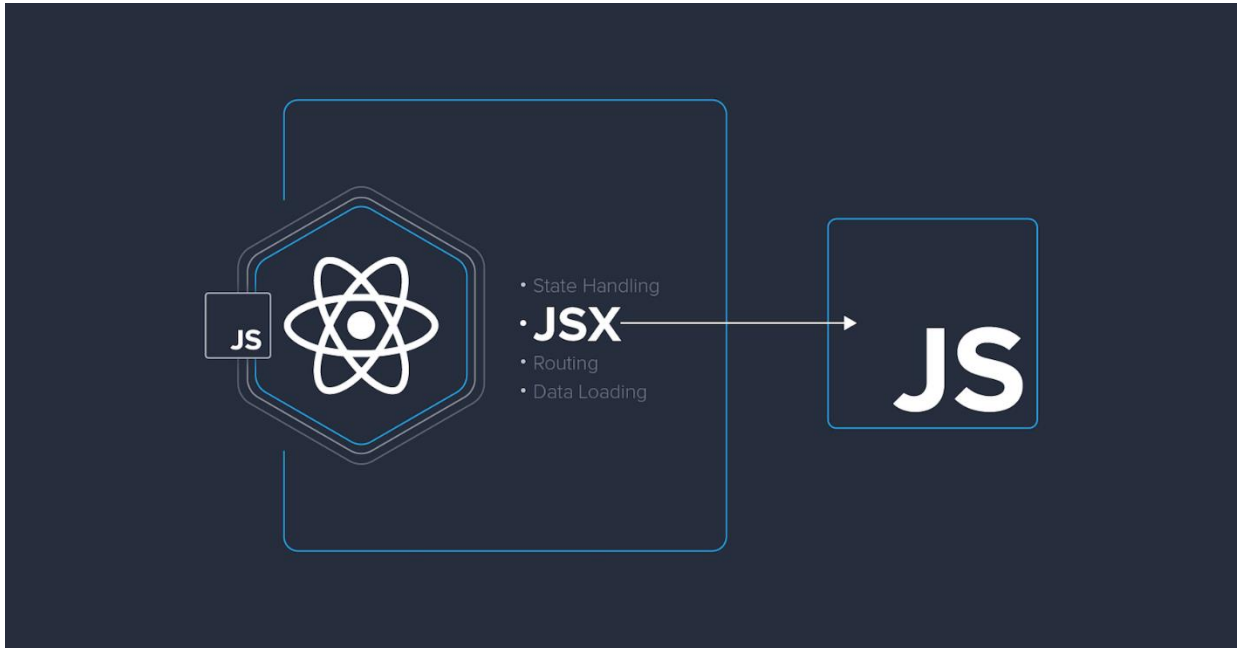
- Facebook에서 개발한 프론트엔드(JS) 라이브러리
- Component를 기본 단위로 하여 웹페이지 생성 → 레고 블록과 유사한 것!
- 코드의 가독성, 재사용성 증가, 유지보수에 편리

Virtual DOM



- React는 Virtual DOM 기반: 브라우저 상의 DOM을 직접 제어하는 대신, 메모리에 가상의 DOM을 생성하고 이를 사용
- DOM을 직접 제어하려면 우리는 이전에 JS를 사용했을 때처럼 바뀐 부분만 우리가 직접 찾아내야 함! → 바뀐 부분을 알아서 찾아 자동 렌더링 해주기 위한 것이 virtual DOM
- JSX 문법을 기반으로 함(JS와 유사하지만 다른 부분도 존재)

JSX



- JSX: Javascript XML
- React는 JSX 문법을 기반으로 함(JS와 유사하지만 다른 부분도 존재)
- React 내에서 HTML 쓸 수 있게 해줌

JSX

```
const myelement = <h1>React is {5 + 5} times better with JSX</h1>;
```

```
const myelement = <input type="text" />;
```

- { } 안에 expression을 적어줌 ex) {5+5} → 10
- XML 문법을 따름 → empty element를 / 로 끝냄

Render

```
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';
```

```
25  export default App;
```

- 각 컴포넌트를 렌더링 해주는 방식으로 동작
- 컴포넌트를 다른 JS파일에서 참조하기 위해서는 export/import 필요

Render

```
7  ReactDOM.render(
8    <React.StrictMode>
9      <App />
10   </React.StrictMode>,
11   document.getElementById('root')
12 );
```

- ReactDOM.render → (HTML code, HTML element)를 인자로 받음
- 인자로 받은 code를 해당 element로 렌더링해줌(위 예시에서는 root, 즉 문서 전체로 하는 것)
- 이 때 HTML code를 컴포넌트로 구성 가능

Routes

```
6   class Routes extends Component{
7       render(){
8           return(
9               <Router>
10                  <Switch>
11                      <Route exact path='/' component={MainPage}/>
12                      <Route path='/movies' component={MoviePage}/>
13                      <Redirect path="*" to ="/"/>
14                  </Switch>
15              </Router>
16          )
17      }
18  }
19
20  export default Routes;
```

```
7   ReactDOM.render(
8     <React.StrictMode>
9       <Routes />
10    </React.StrictMode>,
11    document.getElementById('root')
12  );
```

- Routes -> 주소에 따라 다른 컴포넌트를 렌더링 가능
- exact path → 정확히 일치할 때, path → 뒤에 다른 것까지 포함
- Redirect path="*" to "/"→ 이외에 다른 path가 오면 기본 경로로 리다이렉트

Components

```
class Car extends React.Component {
  constructor() {
    super();
    this.state = {color: "red"};
  }
  render() {
    return <h2>I am a Car!</h2>;
  }
}
```

- extends : Python에서의 상속(inheritance) 키워드
- constructor: 생성자 → component의 property를 초기화하는 부분
- super(); → parent component의 constructor를 실행, parent component에서 정의된 function들에 대한 access 가능
- props와 states를 가짐 (component의 property)→ 다음 slide에서 설명

Components

```
import React from 'react';
import ReactDOM from 'react-dom';

class Car extends React.Component {
  render() {
    return <h2>Hi, I am a Car!</h2>;
  }
}

export default Car;
```

```
import React from 'react';
import ReactDOM from 'react-dom';
import Car from './App.js';

ReactDOM.render(<Car />, document.getElementById('root'));
```

- export default Car → 다른 파일에서 Car component 인식 가능
- 코드의 재사용성이 획기적으로 증가! (여러 개의 다른 파일에서 해당 component 사용)
- 웹페이지가 변경되어야 할 경우 해당 componen만 바꿔주면 다른 파일에서도 똑같이 적용 → 유지보수의 획기적 변화!

Props

```
const myelement = <Car brand="Ford" />;
```

```
class Car extends React.Component {
  render() {
    return <h2>I am a {this.props.brand}!</h2>;
  }
}
```

- props: function argument와 같은 역할
- props를 전달하기 위해서는 component 생성 시 값을 넣어 줌 ex) brand=“Ford”
- argument로 받은 props를 component 내부에서 활용

Props

```
class Car extends React.Component {
  render() {
    return <h2>I am a {this.props.brand}!</h2>;
  }
}

class Garage extends React.Component {
  render() {
    const carname = "Ford";
    return (
      <div>
      <h1>Who lives in my garage?</h1>
      <Car brand={carname} />
      </div>
    );
  }
}

ReactDOM.render(<Garage />, document.getElementById('root'));
```

- component 내부에서 또 다른 component를 부를 때 props 활용 가능
- props 값을 변수 이름으로 적어주는 것도 가능
- props 값이 object가 될 수도 있음
  ex) brand={name:"Ford", model:"Mustang}
  this.props.brand.model값을 내부에서 사용
- component에 constructor가 존재할 때는 항상 super(props)를 호출해주어야 함!!

→ 따라서, constructor 작성 시에는 무조건

```
constructor(props){
        super(props);
        (other lines...)
}
```

와 같은 형태를 지켜줄 것!

States

```
class Car extends React.Component {
  constructor(props) {
    super(props);
    this.state = {
      brand: "Ford",
      model: "Mustang",
      color: "red",
      year: 1964
    };
  }
  render() {
    return (
      <div>
        <h1>My {this.state.brand}</h1
        <p>
          It is a {this.state.color}
          {this.state.model}
          from {this.state.year}.
        </p>
      </div>
    );
  }
}
```

- state: component 내부의 property value
- props와 마찬가지로 object를 쓸 수도 있음

States

```
class Car extends React.Component {
  constructor(props) {
    super(props);
    this.state = {
      brand: "Ford",
      model: "Mustang",
      color: "red",
      year: 1964
    };
  }
  render() {
    return (
      <div>
        <h1>My {this.state.brand}</h1
        <p>
          It is a {this.state.color}
          {this.state.model}
          from {this.state.year}.
        </p>
      </div>
    );
  }
}
```

- state 값을 바꿀 때는 setState라는 함수 이용!
- 절대로 state 값을 direct로 바꾸지 말 것!
- setState 함수를 호출하는 순간 페이지가 자동으로 re-render 됨!

- setState함수 호출 시 자동 re-render→ setState 함수가 계속 호출된다면 무한루프에 걸리는 문제 발생하므로 주의

Events

```
<button onClick={shoot}>Take the Shot!</button>
```

```
class Football extends React.Component {
  shoot = (a) => {
    alert(a);
  }
  render() {
    return (
      <button onClick={() => this.shoot("Goal")}>Take the shot!</button>
    );
  }
}

ReactDOM.render(<Football />, document.getElementById('root'));
```

- onClick method 시에 함수를 위와 같이 호출 가능
- Note) React component 내부에서 this는 component 자기 자신을 가리킴!

Events

```
class Football extends React.Component {
  shoot(a) {
    alert(a);
  }
  render() {
    return (
      <button onClick={this.shoot.bind(this, "Goal")}>Take the shot!</button>
    );
  }
}

ReactDOM.render(<Football />, document.getElementById('root'));
```

- 화살표 함수를 사용하지 않을 경우 위와 같이 bind를 통해 this 를 같이 넘겨줘야 함!

Reference

- https://www.w3schools.com/react/
- https://opentutorials.org/module/4058
- https://ko.reactjs.org/tutorial/tutorial.html

과제

- 화면에는 처음에 숫자 0이 표시됨, 숫자 증가/감소 버튼 2개가 존재함
- 숫자 증가 버튼 클릭 시 숫자 값이 1 증가, 감소 버튼 클릭 시 1 감소
- hint) onclick 시 state 값을 적절히 변화시킬 것