

React Study 4강

목 차

1. 리액트 이벤트
2. 리액트 라이프사이클

2018. 11. 15

1. 리액트 이벤트

- 이벤트 : 유저가 웹브라우저에서 DOM요소들과 상호작용하는것
- 리액트이벤트 : 기존웹브라우저 HTML 이벤트와 거의 유사
- 틀린점 기존이벤트와 유사하지만
- 이벤트 이름이 camelCase : onclick이 아니라 onClick
- 이벤트에 실행할 자바스크립트코드를 전달하는게 아니라 함수형태 값을 전달
- DOM요소에만 이벤트 설정가능 div, button, input
- 사용자가 만든 컴포넌트의 경우 props를 전달하게 됨
- 이벤트 핸들링 첫번째 예제 만들어보기
- 이벤트를 처리할 부분을 만듭니다.(핸들러라고 합니다.)
- testEvent1() 알럿으로 메시지만 출력해줍니다.
- <h1 태그에서 이벤트를 만들고 핸들러를 호출해줍니다.

```
testEvent1() {  
  alert("React Event 11")  
}
```

```
render() {  
  return (  
    <div>  
      <h1 onClick={this.testEvent1}>My Movie Chart </h1>  
    </div>  
  );  
}
```

- 지난시간에 연습한 스테이트 값을 세팅하는것을 복습해봅시다.
-

1. 리액트이벤트 – 이벤트호출부에 바로 화살표 함수로 구현

- 텍스트 버튼을 만듭니다.
- 텍스트 버튼 입력부분이 변경될때 이벤트를 호출하도록 합니다. onChange
- 다른메서드 없이 바로 화살표 함수로 구현합니다.

```
<input type="text" name="message" placeholder="메세지입력하세요" className="col-xs-3 m
onChange={
  (e) => {
    alert("React Event 33")
  }
}/>
```

```
onChange={
  (e) => {
    //alert("React Event 33")

    this.setState(
      { userName : '아이콘',
        message : e.target.value
      }
    )
  }
}/>
```

1. 리액트이벤트 – 이벤트호출부에 바로 화살표 함수로 구현

- 텍스트항목 onChange이벤트 화살표함수 연습 결과 참조

```
1  import React, { Component } from "react";
2
3  class Lec02_arrowfunc extends Component {
4    render() {
5      return (
6        <div>
7          react study event : 화살표함수로
8          <br />
9          <br />
10         <input
11           type="text"
12           onChange={e => {
13             alert("리액트이벤트연습 화살표 함수로" + e.target.value);
14           }}
15         />
16       </div>
17     );
18   }
19 }
20 export default Lec02_arrowfunc;
```



1. 리액트 이벤트 - 화살표함수로 임의메서드 만들어 보시죠

- 화살표 함수가 직접 들어가니 메인소스가 좀 복잡해지는것 같으니 메서드로 빼보죠.
- 버튼을 만들어서 연습해봅시다.
- 버튼에 testEvent2가 버튼클릭시 호출됩니다.
- 이벤트 핸들러를 화살표함수로 만들어봅시다.

```
testEvent2 = () => {  
  alert("React Event 22")  
  this.setState( {userName: '트와이스'})  
}
```

```
render() {  
  return (  
    <div>  
      <h1 onClick={this.testEvent1} >My Movie Chart </h1>  
      <span>{this.state.userName} {this.state.message} </span>  
      <br/>  
      <button type="button" onClick={this.testEvent2} class="btn btn-primary">I  
    </div>  
  );  
}
```

1. 리액트 이벤트 - 화살표함수로 임의메서드 만들어 보시죠

- 화살표 함수가 직접 들어가는 부분을 위쪽에 메서드 만들어 주고 호출

```
import React, { Component } from "react";

class Lec03_method extends Component {
  handleClick = () => {
    alert('임의메서드가 잘 호출되나. ');
  }

  render() {
    return (
      <div>
        react study event : 임의메서드로 빼보죠
        <br />
        <br />
        <input
          type="text"
        />
        <button onClick={this.handleClick}> 이벤트세번째 </button>
      </div>
    );
  }
}

export default Lec03_method;
```



1. 리액트 이벤트 - 화살표함수로 임의메서드 만들어 보시죠

- 조금더 업그레이드 해서 텍스트박스도 넣고
- 지난시간에 공부했던 스테이트도 넣어서
- 텍스트 박스 입력값을 onChange 이벤트로 임의메서드 하나 더 만듭니다.
- 버튼을 클릭하면 스테이트값을 업데이트 해보죠
- 맨 밑에 스테이트 값과 텍스트 값을 디스플레이 해봅니다.

```
1  import React, { Component } from "react";
2
3  class Lec03_method extends Component {
4    constructor(props) {
5      super(props);
6      this.state = {
7        userName: "--",
8        message: "--"
9      };
10   }
11
12   handleBtnClick = () => {
13     alert("임의메서드가 잘 호출되나.");
14     this.setState({
15       userName: "트와이스"
16     });
17   };
18
19   handleTextboxChange = e => {
20     //alert(e.target.value);
21     this.setState({
22       message: e.target.value
23     });
24   };
25
26   render() {
27     return (
28       <div>
29         react study event : 임의메서드로 빼보죠
30         <br />
31         <br />
32         <input type="text" onChange={this.handleTextboxChange} />
33         <button onClick={this.handleBtnClick}> 이벤트세번째 </button>
34         <br />
35         userName : {this.state.userName} <br />
36         message : {this.state.message}
37       </div>
38     );
39   }
40 }
```



1. 리액트이벤트 – 인풋값이 여러 개 key를 동적으로 할당

- this.setState({
- [e.target.value] : e.target.value
- 인풋값이 여러 개인 경우 함수한개를 재활용해서 key와 value를 동적으로 할당

이메일: 핸드폰:

- 위처럼 입력박스 두개를 추가하고 아래쪽에 입력된 값을 보여주는 형태로 연습
- state두개 추가해서 데이터 연결항목으로 사용합니다.
- 같은이벤트이므로 이벤트핸들러 한 개로 재사용해봅니다.

```
<input type="text" name="email_msg" placeholder="이메일입력하세요." className="col-xs-3 mt-2 mr-2"
  value={this.state.email_msg}
  onChange={this.handleChange4}
/>
<input type="text" name="hpno_msg" placeholder="휴대폰입력하세요." className="col-xs-3 mt-2"
  value={this.state.hpno_msg}
  onChange={this.handleChange4}
/>
<br/>
<span>이메일: {this.state.email_msg}    핸드폰: {this.state.hpno_msg}</span>
```


1. 리액트이벤트 – 인풋값이 여러 개 key를 동적으로 할당

- 객체의 키값과 밸류값을 동적으로 할당 이벤트 만들어 봅니다. 최종결과

```
1 import React, { Component } from "react";
2
3 class Lec04_input extends Component {
4   constructor(props) {
5     super(props);
6     this.state = {
7       userName: "-",
8       message: "--"
9     };
10  }
11
12  handleTextboxChange = e => {
13    this.setState({
14      [e.target.name]: e.target.value
15    });
16  };
17
18  render() {
19    return (
20      <div>
21        react study event : 인풋값2개 state key 동적사용
22        <br />
23        <br />
24        <input
25          type="text"
26          name="userName"
27          placeholder="사용자명 입력하세요."
28          onChange={this.handleTextboxChange}
29        />
30        <br />
31        <input
32          type="text"
33          name="message"
34          placeholder="메세지 입력하세요."
35          onChange={this.handleTextboxChange}
36        />

```

https://50qj

Hello React Study 4th

react study event : 인풋값2개 state key 동적사용

| |
|----|
| 11 |
| 22 |

userName : 11
message : 22

1. 리액트이벤트 입력박스에서 엔터이벤트를 연습해봅니다.

- 입력박스에 키이벤트를 받는 방법을 확인합니다.
- onKeyPress 이벤트를 가진 입력박스를 추가합니다.
- 유저명과 입력한 값을 알럿메세지로 보여주는 UI를 연습합니다.

트와이스

키입력이벤트연습

```
<br/>
<input type="text" name="userName" placeholder="유저명" className="col-xs-3 mt-2 mr-2"
  value={this.state.userName}
  onChange={this.handleChange4}
/>
<input type="text" name="keyin" placeholder="키입력이벤트연습" className="col-xs-3 mt-2 mr-2"
  value={this.state.keyin}
  onChange={this.handleChange4}
  onKeyPress={this.testEvent5}
/>
```

- 기존 html에서는 키코드 13 이런식 이었는데 'Enter' 이런식으로 가능하네요.

```
testEvent5 = (e) => {
  if(e.key === 'Enter') {
    this.handleClick()
  }
}

handleClick = (e) => {
  alert(this.state.userName + this.state.keyin);
}
```

1. 리액트이벤트 입력박스에서 엔터이벤트를 연습해봅니다.

- 입력박스에 키이벤트를 받는 연습 - 최종결과코드

lec05_keyevent

엔터키입력됨==> 트와이스반가워요

확인

```
15  });
16  };
17
18  handleKeyin = e => {
19    if (e.key === "Enter") {
20      this.processOne();
21    }
22  };
23
24  processOne = e => {
25    alert("엔터키입력됨==> " + this.state.userName + this.state.message);
26  };
27
28  render() {
29    return (
30      <div>
31        react study event : key event 연습
32        <br />
33        <br />
34        <input
35          type="text"
36          name="userName"
37          placeholder="사용자명 입력하세요."
38          onChange={this.handleTextboxChange}
39          onKeyDown={this.handleKeyin}
40        />
41        <br />
```

https://50qj

Hello React Study 4th

react study event : key event 연습

트와이스

반가워요

userName : 트와이스

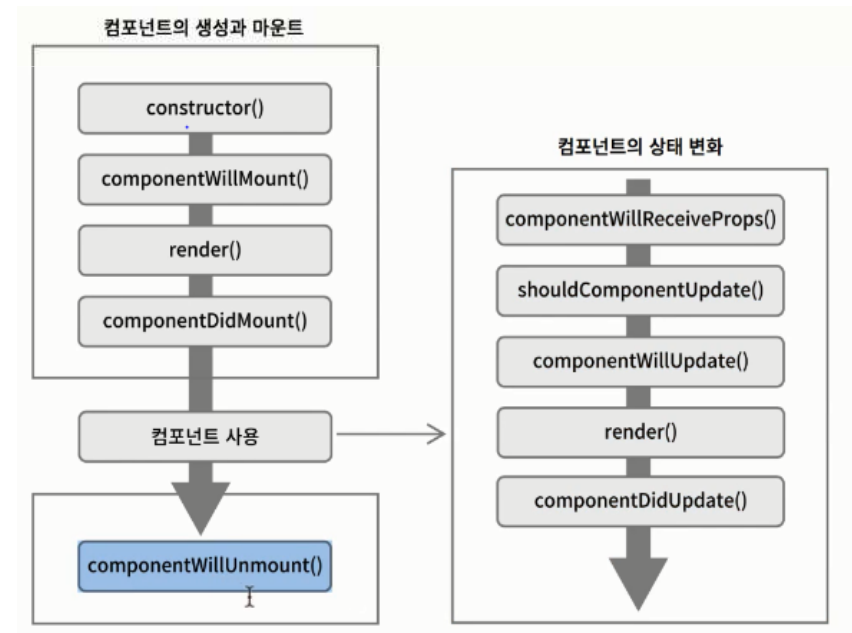
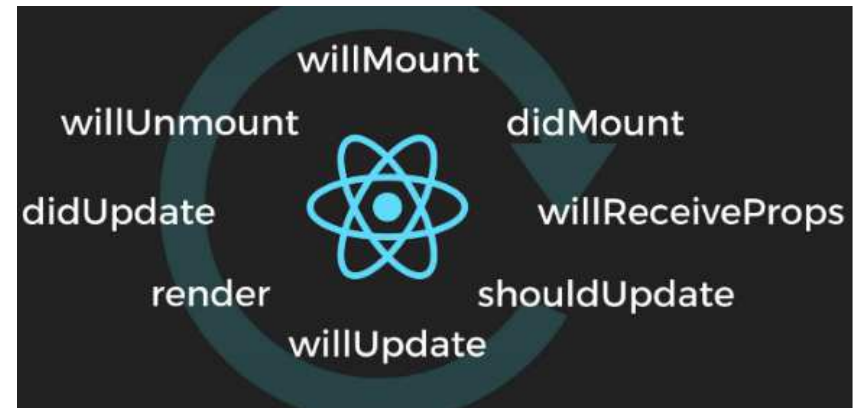
message : 반가워요

- 리액트

- 한템포 쉬어 갑니다.

2. 리액트 라이프사이클

- 리액트 UI컴포넌트는 생성 갱신 소멸
 - 컴포넌트를 세밀하게 제어
 - 생성 : 마운팅 이벤트
 - 갱신 : 속성, 상태 변경
 - 소멸 : 언마운팅 이벤트
-
- 마운팅 이벤트
 - constructor()
 - componentWillMount()
 - render()
 - componentDidMount()
-
- 갱신 이벤트
 - componentWillReceiveProps() - deprecated
 - shouldComponentUpdate()
 - componentWillUpdate()
 - render()
 - componentDidUpdate()
-
- 언마운팅
 - componentWillUnmount



2. 리액트라이프사이클을 콘솔로그에 찍어보며 익숙해지자

- 라이프사이클이란 개념이 초급자에게 조금 어려운 개념이다.
- 재정의된 API들에 익숙해지면서 사용예를 조금씩 넓혀 가는 전략이 좋아 보입니다.W
- 간단하게 클래스에 해당 API를 호출하고 콘솔로그를 찍어보겠습니다.
- MovieCard300Tail.js에 해당 API를 순서대로 찍어보았습니다.

react-dom.development.js:20733
Download the React DevTools for a better development experience: <https://fb.me/react-devtools>

| | |
|-------------------------------|--|
| 001 constructor() 생성자 | MovieCard300Tail.js:8 |
| 002 componentWillMount() 마운트전 | MovieCard300Tail.js:17 |
| 003 render() | MovieCard300Tail.js:48 |
| 004 componentDidMount() 마운트후 | MovieCard300Tail.js:21 |

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

class MovieCard300Tail extends Component {
  //마운트
  constructor(props) {
    super(props)
    console.log('001 constructor() 생성자')
    this.state = { userName: '' }
  }

  componentWillMount() {
    console.log('002 componentWillMount() 마운트전')
  }

  componentDidMount() {
    console.log('004 componentDidMount() 마운트후')
  }
}
```

- <https://jaeyeophan.github.io/2018/01/01/React-4-Component-Life-Cycle/>
-

2. 리액트라이프사이클을 콘솔로그에 찍어보며 익숙해지자

- 컴포넌트업데이트하기위해 인풋박스에 값을 입력한후 state에 세팅하도록 구성

| | | |
|-----------------------------|-------|------------------------|
| 006 shouldComponentUpdate() | . | MovieCard300Tail.js:26 |
| 007 componentWillUpdate() | 업데이트전 | MovieCard300Tail.js:31 |
| 003 render() | | MovieCard300Tail.js:48 |
| 008 componentDidUpdate() | 업데이트후 | MovieCard300Tail.js:35 |

```
handleChange(e) {
  this.setState({ userName: e.target.value })
}

render() {
  console.log('003 render() ')

  const setStateHandler = e => {
    console.log('call setState')
    this.setState({ r: Math.random() })
  }

  return (
    <section className="container mt-5 float-right">
      <div>
        <h6 name="{this.state.userName}">
          Copyright 2018 lion all rights reserved{' '}
        </h6>
      </div>
      <input
        type="text"
        className="text text-primary mt-2"
        value={this.state.userName}
        onChange={this.handleChange.bind(this)}
      />
    </section>
  )
}
```

2. 리액트 컴포넌트 라이프사이클 API

- 컴포넌트가 처음으로 렌더링할때 어떤작업이 필요한 경우
 - 컴포넌트를 업데이트 하기 전후로 어떤작업이 필요한 경우
 - 불필요한 업데이트를 방지하여야 하는 경우
 - 10가지 API
 - Will 점두사가 붙는거는 어떤작업 전에 한다는 의미
 - Did 점두사가 붙는거는 어떤작업 후에 한다는 의미
 - 라이프사이클은 3가지 카테고리 : 마운트, 업데이트, 언마운트
 - 마운트 : DOM이 생성되고 웹브라우저상에 나타나는것을 Mount
 - constructor() : 컴포넌트 초기화시 할일
 - getDerivedStateFromProps() : 프랍스 값을 스테이트에 동기화
 - render() : UI에 보여주는 것
 - componetDidMount() : 컴포넌트가 마운트된후에 호출되는 API
-

2. 리액트 컴포넌트 라이프사이클 API

- 업데이트
 - props가 바뀔때 호출
 - state가 변경될때 호출
 - 부모컴포넌트가 리렌더링될때 호출
 - forceUpdate로 강제렌더링될때 호출

 - `getDerivedStateFromProps` : 프롭스가 바뀌어서 업데이트할때 호출
 - `shouldComponentUpdate` : 리렌더링여부결정 `false`면리렌더링안함
 - `render`
 - `getSnapshotBeforeUpdate` : 컴포넌트 변화를 DOM에 반영하기전에 호출
 - `componentDidUpdate` : 컴포넌트 업데이트 작업이 끝난후

 - 언마운트 : 컴포넌트를 DOM에서 제거하는것을 `ummount`
 - `componentWillUnmount` : 컴포넌트가 웹브라우저에서 사라지기 전
 -
-

2. 리액트 컴포넌트 라이프사이클 API

- 스톱워치 예제
- stopwatch.js 파일을 생성후 rcc로 틀을 만듭니다.
- 라이프사이클중 willMount와 willUnmount 사용연습을 위한입니다.
- <h1>태그로 틀에 오류가 없는지 app.js에 렌더링 해봅니다.
- constructor에 isLive, curTime, startTime을 생성합니다.
- isLive는 스톱워치가 동작여부를 알려줍니다.
- curTime은 현재시간, startTime은 시작시간 입니다.
- willMount에 매초마다 갱신내역을 보여주도록 합니다.
- tick()함수를 구현하여 현재시각을 매초마다 변수에 담습니다.
- 스타트 버튼이 눌리면 스타트 시간을 변수에 담습니다.
- 정지 버튼이 눌리면 동작정지 isLive를 업데이트합니다.

```
constructor(props) {  
  super(props)  
  this.state = {  
    isLive: false,  
    curTime: 0,  
    startTime: 0  
  }  
  this.timerId = 0  
}  
  
componentWillMount() {  
  //마운트했을때  
  this.timerId = setInterval(e => {  
    this.tick()  
  }, 1000)  
}  
  
componentWillUnmount() {  
  clearInterval(this.timerId)  
}  
  
//매초마다 실행  
tick() {  
  if( this.state.isLive) {  
    const v = new Date().getTime()  
    this.setState({curTime: v})  
  }  
}
```

2. 리액트 컴포넌트 라이프사이클 API

- 스톱워치 최종결과물입니다.

```
// 시작중지버튼 클릭시
clickHandler = () => {
  // 중지
  if (this.state.isLive) {
    this.setState({ isLive: false });
    return;
  }

  // 시작
  const v = new Date().getTime();
  this.setState({
    curTime: v,
    startTime: v,
    isLive: true
  });
};

render() {
  let label = "START";
  if (this.state.isLive) {
    label = "STOP";
  }

  return (
    <div>
      react study event : 스톱워치
      <br />
      <div> {this.state.curTime - this.state.startTime} </div>
      <button onClick={this.clickHandler}> {label} </button>
    </div>
  );
}

export default Lec08_stopwatch;
```

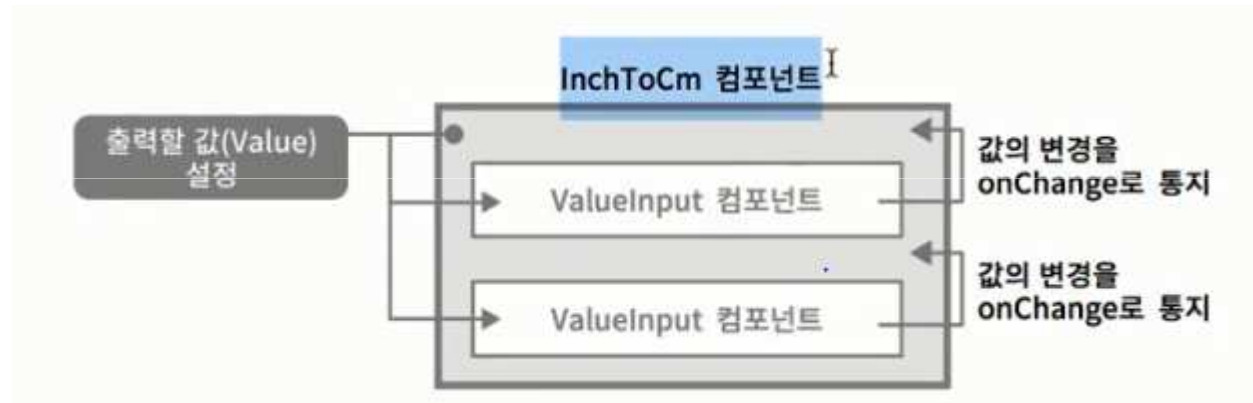


2. 리액트 컴포넌트 라이프사이클 API

- 인치를 센티미터로 변환기
- 컴포넌트간 데이터 전달
- 부모가 자식에겐 props로 전달
- 자식이 부모에게 값의변경을 이벤트로 통지

inch :
0.39370078740157477

cm :
1

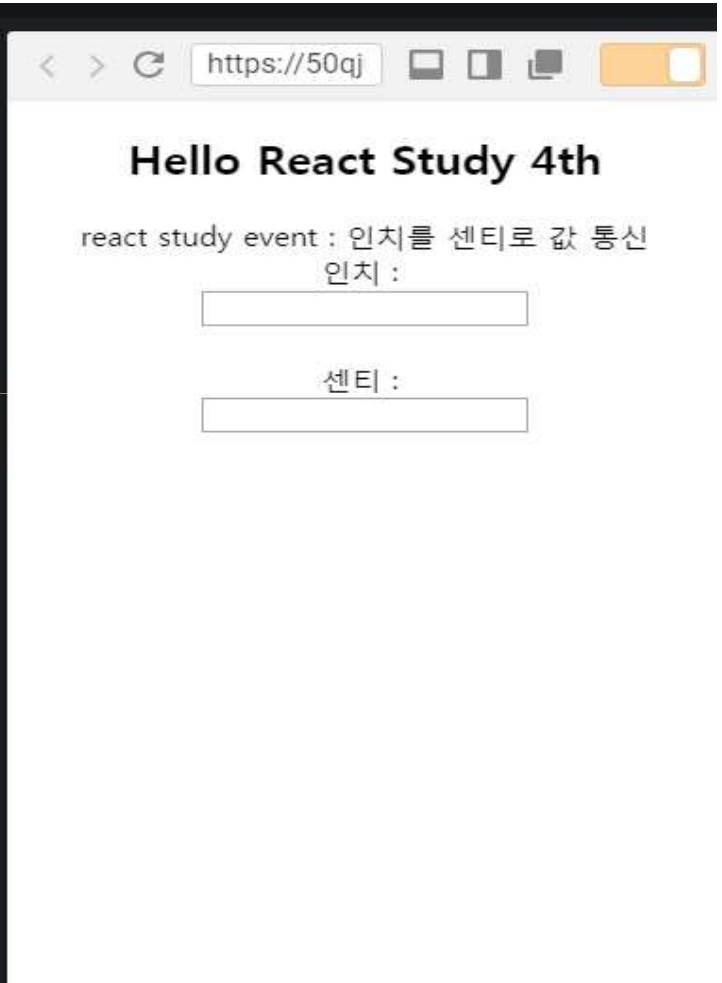


2. 리액트 컴포넌트 라이프사이클 API

- 인치를 센티미터로 최종결과물입니다.

-

```
1  import React, { Component } from "react";
2  import Lec09_ValueInput from "../lec09_ValueInput";
3
4  class Lec09_InchToCm extends Component {
5    constructor(props) {
6      super(props);
7      // 하위 ValueInput 컴포넌트에 출력할 값 저장 state
8      this.state = {
9        inch: 0,
10       cm: 0
11     };
12   }
13
14   render() {
15     return (
16       <div>
17         react study event : 인치를 센티로 값 통신
18         <br />
19         <Lec09_ValueInput title="인치" />
20         <br />
21         <Lec09_ValueInput title="센티" />
22       </div>
23     );
24   }
25 }
26 export default Lec09_InchToCm;
27
```



리액트

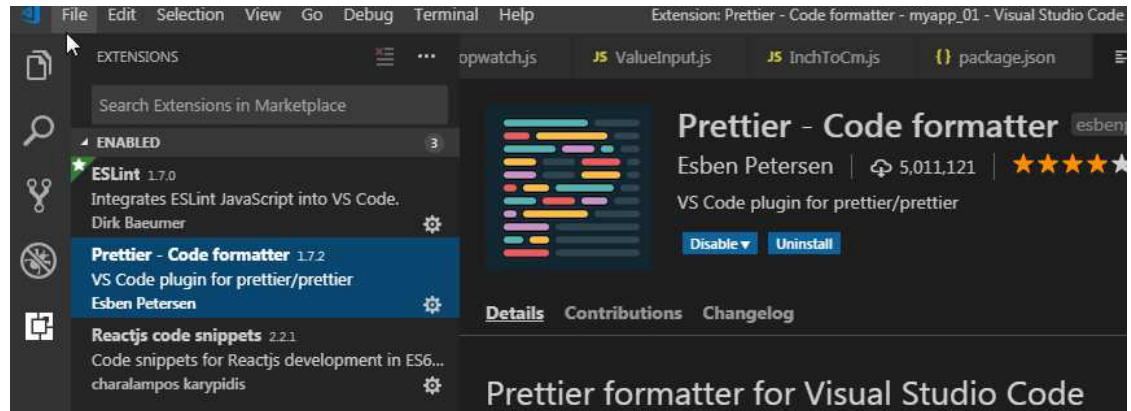
– 잠시쉬어갑니다.

3. 기타 공부해본 내용 – eslint 와 프리티어 같이 쓰기

- 둘다 코드 정렬 및 헬핑 기능인데 각각 특화장점이 있어 같이 쓰면 좋으네여.
- 그런데 두개가 겹치는 부분이 있어서 그부분을 좀 설정해주면 가끔 렉걸리는걸 방지
-

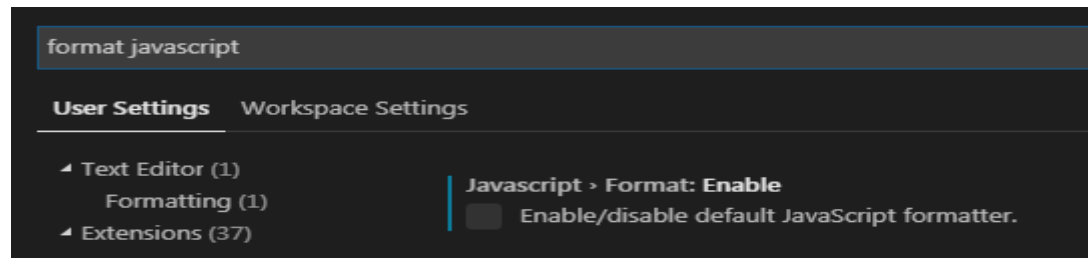
- eslint prettier extension 설치
- 패키지json 수정

```
"eslintConfig": {
  "extends": [
    "prettier"
  ],
  "rules": {
    "react/prefer-stateless-function": 0,
    "react/jsx-filename-extension": 0,
    "react/jsx-one-expression-per-line": 0
  }
},
```



- 코드포맷 Vscode꺼 꺼야 프리티어것 적용됨 설정에서 enabled시킴
- yarn add eslint-config-prettier
- .prettierrc 파일생성

```
{
  "singleQuote": true,
  "semi": false,
  "useTabs": false,
  "tabWidth": 2,
  "trailingComma": "all",
  "printWidth": 80
}
```



3. 기타 공부해본 내용 - 부트스트랩

- 공부하다 아무래도 개발내용이 넘 지저분해지면 별루이니
- 트위터에서도 비슷한 생각을 아주 예전에 했군여..
- 개발자가 적용가능하게 CSS를 만들어서 공짜로 배포해주네여 함 적용해봅니다.

- 관련사이트 getbootstrap.com
- 인덱스Html 수정후
- className적용

```
<!-- Bootstrap CSS -->
<link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.1.3/css/bootstrap.min.css">

<title>React App</title>
</head>
<body>
  <noscript>
    You need to enable JavaScript to run this app.
  </noscript>
  <!-- Optional JavaScript -->
  <!-- jQuery first, then Popper.js, then Bootstrap JS -->
  <script src="https://code.jquery.com/jquery-3.3.1.slim.min.js"></script>
  <script src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.14.3/umd/popper.min.js"></script>
  <script src="https://stackpath.bootstrapcdn.com/bootstrap/4.1.3/js/bootstrap.min.js"></script>
```

My Movie Chart

유저명:user 메시지:

React Event 2

메시지입력하세요

이메일입력하세요. 휴대폰입력하세요.

이메일: 핸드폰:

user 키입력이벤트연습

| | | |
|---|--|--|
| 랭킹1위 | 랭킹2위 | 랭킹3위 |
| 퍼스트맨 라이언고슬링 달에 가는 역경을 이겨내는 영화인가.. | 해리포터 다니엘 레드클리프 위압적인 숙부와 냉담한 이모, 욕심 많고 버릇없는 사촌 밑에서 갇은 구박을 견디며 계단 밑 벽장에서 생활하는 해리포터 | 보헤미안랩소디 벤하디 나는 스타가 아니다, 다만 전설이다. |
| | | 랭킹4위 |
| | | 완벽한타인 조진웅 핸드폰 공유하면 이케됨 |