

백엔드 개발자가 허우적대는 리액트 이야기 천하제일 스타리그

김대희

발표자 소개

- 비상교육 Back-End 개발자
- Spring, Classic ASP, ETC...

목차

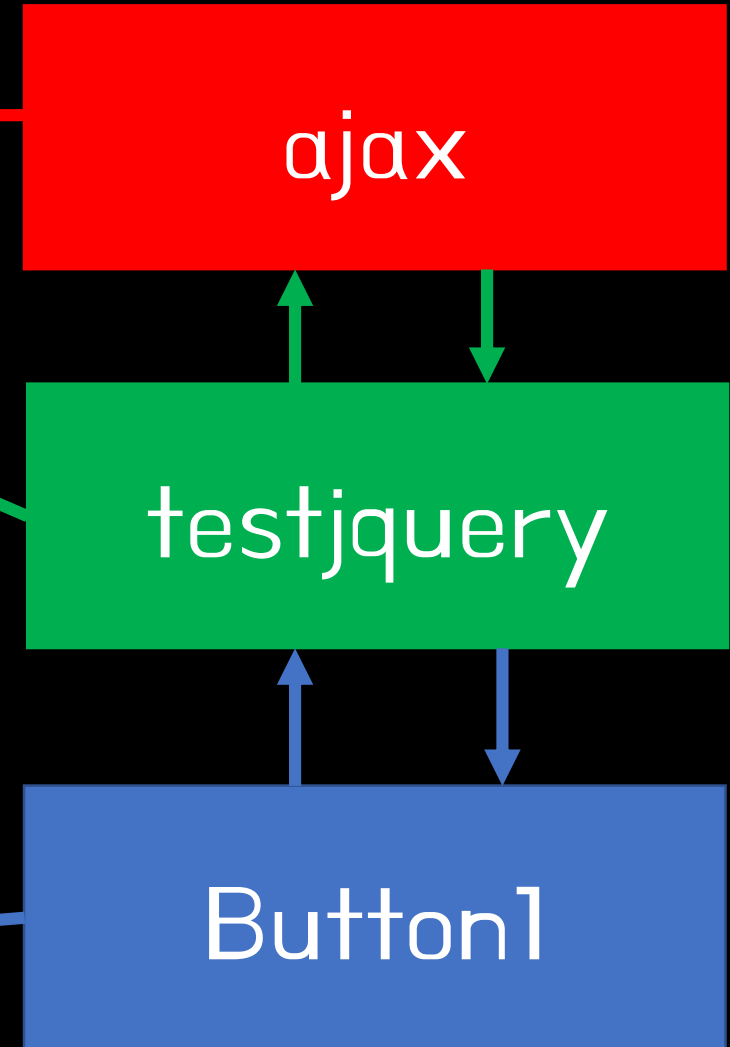
- React를 하기 전
- React를 하면서 겪은 사례
- React를 하면서 느낀 점
- React를 하면서 어려웠던 점
- React를 통해 추후에 해보고 싶은 것들
- 결론

React를 하기 전

- 필요한 요소만 그때그때 끼워 맞춤
- DOM은 무엇인가
- Server 작업을 표시하기 위한 도구일 뿐...

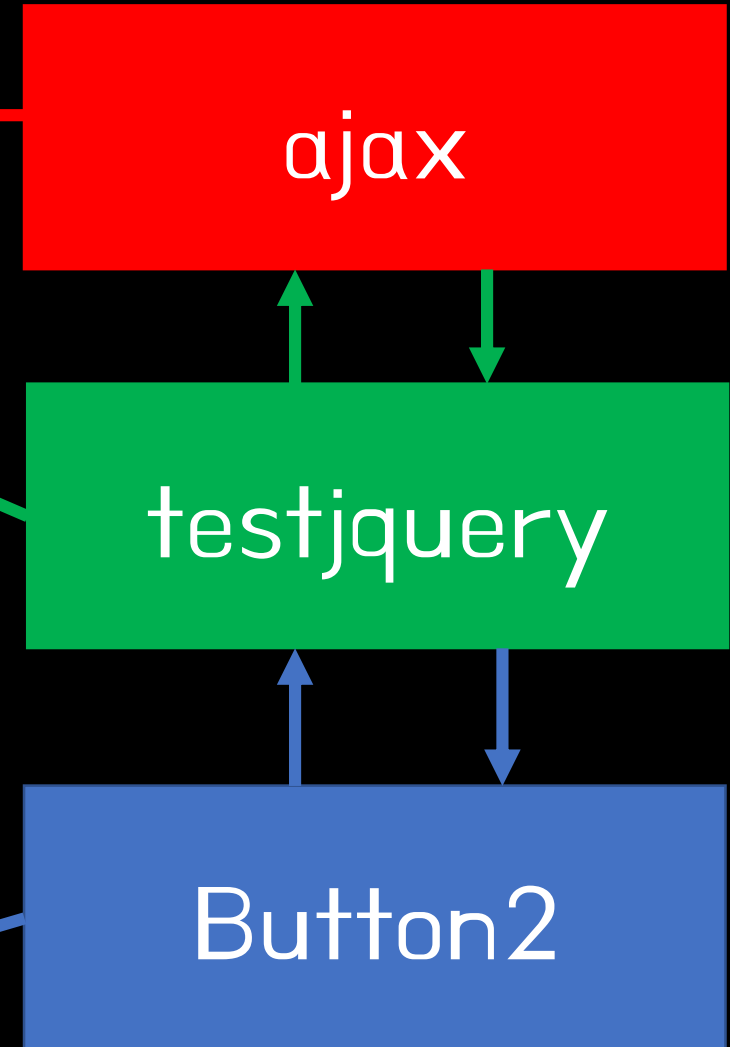
React를 하기 전

```
function testjquery() {  
  
  $.ajax({  
    url: "test/test",  
    type: 'POST',  
    contentType: "application/x-www-form-urlencoded; charset=UTF-8",  
    data : {test : "test"},  
    success: function (data) {  
      console.log(data);  
    },  
    error: function (jqXHR, textStatus, errorThrown) {  
      alert("실패하였습니다. 다시 시도해주세요.");  
    }  
  });  
}  
  
//버튼 클릭시에 함수가 실행  
$("#button1").click(function(){  
  console.log("testjquery start");  
  testjquery();  
});  
  
<html>  
<input type="button" id="button1"></input>  
<input type="button" id="button2" onclick="javascript:testjquery();"></input>  
</html>
```



React를 하기 전

```
function testjquery() {  
  
  $.ajax({  
    url: "test/test",  
    type: 'POST',  
    contentType: "application/x-www-form-urlencoded; charset=UTF-8",  
    data : {test : "test"},  
    success: function (data) {  
      console.log(data);  
    },  
    error: function (jqXHR, textStatus, errorThrown) {  
      alert("실패하였습니다. 다시 시도해주세요.");  
    }  
  });  
}  
  
//버튼 클릭시에 함수가 실행  
$("#button1").click(function(){  
  console.log("testjquery start");  
  testjquery();  
});  
  
<html>  
<input type="button" id="button1"></input>  
<input type="button" id="button2" onclick="javascript:testjquery();"></input>  
</html>
```



Front-End를
계속 그렇게만 사용할 줄 알았다.

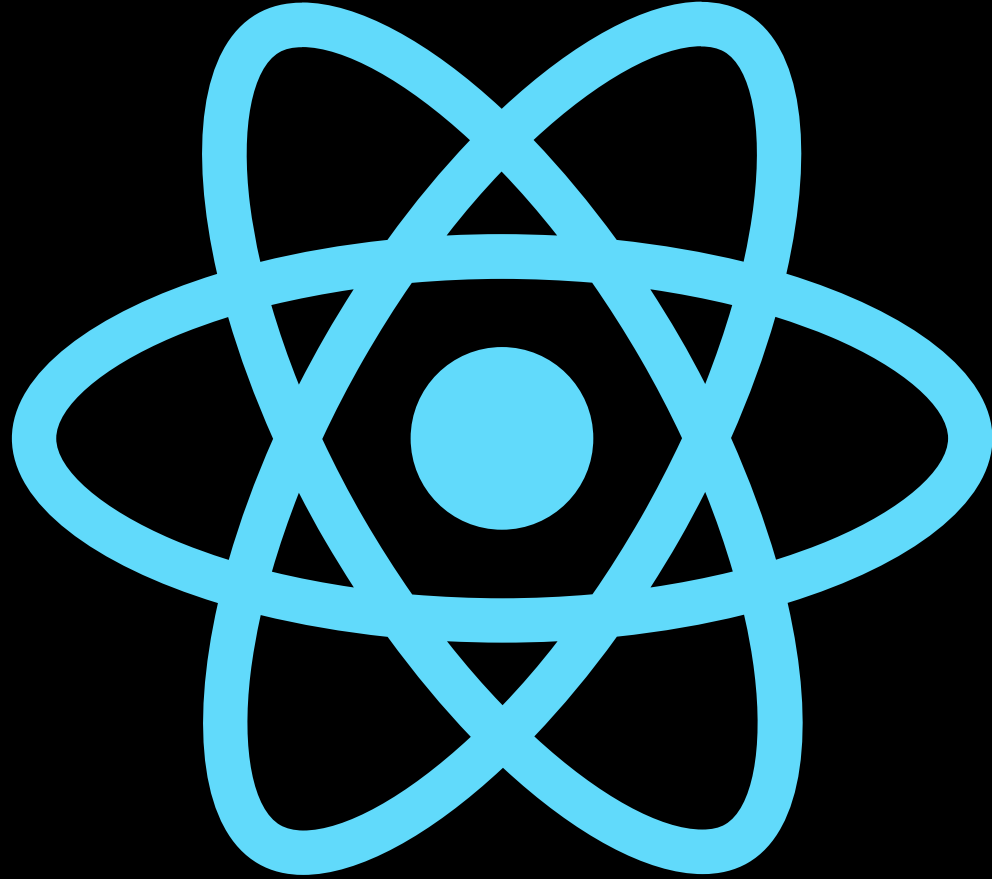
하지만 React를 하게되었...?

React를 하면서

- 수없이 많은 개념들
- Component? JSX ?
- Virtual DOM?
- Life Cycle?
- 좋은 라이브러리들
- ETC...

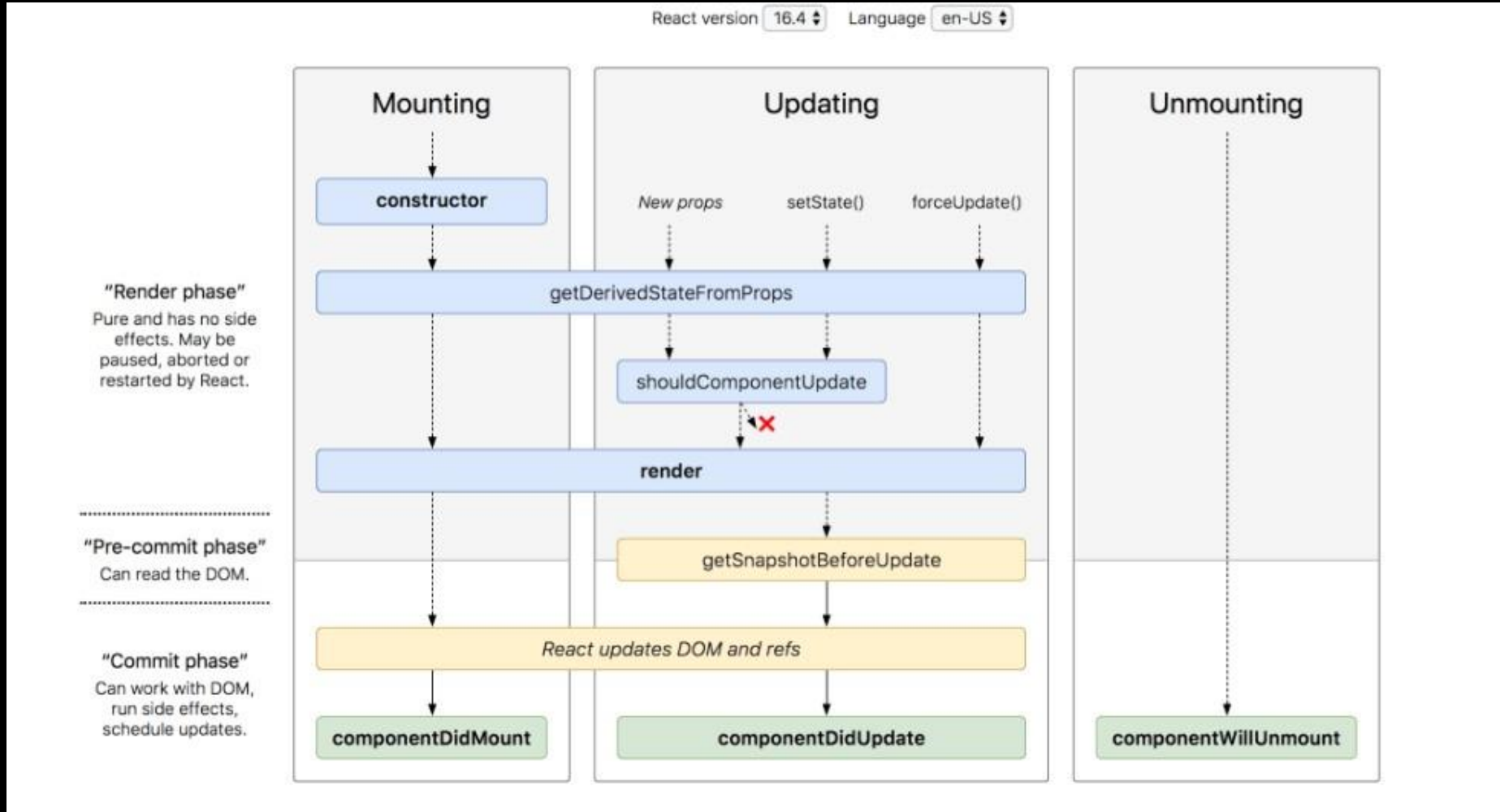
????????????????

React란?



사용자 인터페이스를 만들기 위해 사용되는 JavaScript 라이브러리

Lifecycle이란?



React Component들의 수명주기로, 페이지가 시작할 때부터 사라질 때까지 관리

React를 하면서 겪은 사례(1)

- 변수를 Render, state 에 각각 집어넣음
- Render가 되고 나서 실행되어야 할 함수를 Lifecycle를 이용하지 않고, Render에 집어넣음
- 그 결과 Render 부분이 코드가 지저분해 지고, 코드를 짜는 것이 더욱 복잡해짐
- Class 내부에서 일어나는 함수들은 전부 Class 내부로, Render 부분에서 변수 선언하는 부분은 모두 제거

React를 하면서 겪은 사례(1)

```
function test3(){  
  // api call 함수  
}  
  
class Test extends Component{  
  constructor(props){  
    super(props);  
    this.state = {  
      test4 : 0  
    };  
  }  
  
  render(){  
    let test = 1;  
  
    return(  
      <div>  
        <h2>{test}</h2>  
        <button type="button" onClick={test3(test)}>테스트</button>  
        {test3(test)}  
      </div>  
    )  
  }  
}
```

```
class Test extends Component{  
  constructor(props){  
    super(props);  
    this.state = {  
      test1 : 0,  
      test4 : 0  
    };  
  }  
  
  test3 = () => {  
    // this.state.test1 ..  
    // api call  
  }  
  
  componentDidMount(){  
    // api call  
  }  
  
  render(){  
    return(  
      <div>  
        <h2>{this.state.test1}</h2>  
        <button type="button" onClick={this.test3}>테스트</button>  
      </div>  
    )  
  }  
}
```

React를 하면서 겪은 사례(2)

- 사용자의 입력 문제
- Undefined 및 NULL 고려를 안 했음
- 세상에는 바른 사용자만 있지 않다.
- 맨 처음에는 생성자에서 관리 ->
Redirect Component로 변경

React를 하면서 겪은 사례(2)

```
import { Route, Redirect } from 'react-router'
class Test extends Component{

  constructor(props){
    this.state = {
      testvariable : this.props.testvariable
    };
  }

  componentDidMount(){

  }

  render(){
    return (
      testvariable == null ? ( <Redirect to="/" /> ) : (<TestPage />)
    )
  }
}
```

Redirect Component

React를 하면서 느낀 점

- 한 페이지를 체계적으로 관리할 수 있었음
(state, lifecycle)
- React로 한 페이지에 state를 통해
값들을 편하게 다룰 수 있었음

React를 하면서 느낀 점

```
function test1(){
  var test2 = [0,0,0];
  //.....
  var test3 = 0;
  var test4 = 0;
  // .....
}

function test6(){
  var test5 = 0;
}

//...

function test40(){
  var test41 = 'test';
}

// api call은 어디에?
```



```
class Test extends Component{

  constructor(props){
    this.state = {
      A1 : 0,
      A2 : 0,
      A3 : 0,
      A4 : [0,0,0],
      A5 : 0,
      /// ETC..
      A40 : 0
    };
  }

  componentDidMount(){
    // api call
  }

  render(){
    return (
      <h2></h2>
    )
  }
}
```

React를 하면서 느낀 점

- 항상 최신의 값을 보여줄 필요가 없거나 전체 페이지에 적용되는 정적인 값들을 서버 호출을 통해 관리했지만, Redux-Store 등을 통해 서버 호출을 줄일 수 있었음

React를 하면서 느낀 점



**코드를 조금이나마 쉽게 유지보수하고,
그름을 읽을 수 있었습니다.**

React를 하며 어려웠던 것들

- 어떤 작업을 할 때 좋은 라이브러리 고르기
ex) 불변성 데이터를 다루는 라이브러리 (immutable / immer)
- 빠르게 변화하는 버전들의 변화를 확인하기
ex) react의 경우 15 / 16 차이 이외에 16.X 간에도 변화가
빠르게 일어나고 있다. (16.8에 추가된 Hook API 등 ...)
- Component 스타일링 , Redux,
Redux-saga, React 문법...

추후에 해보고 싶은 작업들

- React 최신 버전과 라이브러리 최신버전을 이용하는 프로젝트
- 다른 Back-End 언어와 React 연동
- Angular, Vue와 같은 다른 Front-End 프레임워크나 라이브러리 사용

결론

- React는 결코 쉽지 않습니다.
- React, Vue, Angular를 하게 된다면, 해당 되는 공식 문서들과 커뮤니티들을 많이 보는 것이 좋습니다. (좋은 분들이 많습니다.)
- Front-End는 사용자와 가장 가깝기 때문에 사용자들을 고려하여야 합니다.

디테일을 챙겨야 합니다.

**프론트 개발자분들
존경합니다.**

감사합니다.