00.

나는 비전공자이며, 프론트엔드로 방향을 정한지 얼마되지 않아 이것저것 알아보았을 때, 리액트를 많이 쓴다는 것을 알게 되었다. 그래서 '무엇을 이용해서 지식을 얻어야할까?' 하고 고민을 했다. 그러다 한 가지 책을 추천받았는데 그 책은 바로`리액트를 다루는 기술`이다.

책에 대한 내용 정리는 벨로그에서 작성하고, 깃허브에 올릴 예정이다.

아직은 초반이라 그런지는 몰라도 책에 대한 거부감도 없을 뿐더러 오히려 책이 재밌어서 공부하기에 괜찮은 것 같다.

> ### 나한테 리액트에 대한 개념 이해도가 왜 필요할까?

처음에 말했듯이 나는 비전공자라 나는 먼저 내가 이것을 왜 쓰는지, 어떻게 쓰는지 전혀 모른다.

내가 이것을 사용하는 이유를 알아야 더 재밌어지고, 사용하는 데에 있어서 거부감이 들지 않기 때문에 가장 먼저 리액트를 사용하는 이유를 알아야할 것 같다.

01.

> ### 리액트

- 최대한 성능을 아끼고 편안한 사용자 경험을 제공하면서 구현하고자 개발

- 자바스크립트 라이브러리로 사용자 인터페이스를 만드는데 사용

- \*\*오직 뷰(View)만 신경 쓰는 라이브러리\*\*이다.

(기타 기능은 직접 구현하여 사용해야 한다.)

- 따라서, \*\*라우팅에는 리액트 라우터\*\*, \*\*Ajax 처리는 axios나 fetch\*\*, \*\*상태 관리는 리덕스, MobX를 사용\*\*하여 빈자리를 채우면 된다.

- 여러 라이브러리를 접해야 한다는 단점이 있지만, 자신의 취향대로 스택을 설정할 수 있다는 장점과 다른 웹 프레임워크나 라이브러리와 혼용할 수도 있다.

\*\*주의 할 점\*\*

- 리액트를 사용하기 전에 작업이 매우 간단할 때는 사용하지 않는 편이 더 나은 성능을 보이기도 한다.

---

리액트 프로젝트에서 특정 부분이 어떻게 생길지 정하는 선언체가 있는데, 이를 \*\*컴포넌트\*\*라고 한다.

<br>

> ### \*\* 컴포넌트 \*\*

- 재사용이 가능한 API로 수많은 기능들을 내장

- 컴포넌트 하나에서 해당 컴포넌트의 생김새와 작동 방식을 정의.

데이터가 변할 때마다 새롭게 리렌더링하면서 성능을 아끼고, 최적의 사용자 경험을 제공할 수 있는 이유는 무엇일까?

이에 대한 답은 \*\*'초기 렌더링'\*\*과 \*\*'리렌더링'\*\* 개념을 이해해야 한다고 한다.

UI관련 프레임워크, 라이브러리를 사용하든지 간에 맨 처음 보일지를 정하는 초기 렌더링이 필요한데 리액트에서는 \*\*render 함수\*\*가 있다.

>### \*\*render 함수\*\*

- 컴포넌트가 어떻게 생격는지 정의

- html 형식의 문자열을 반환하지 않음

- 뷰가 어떻게 생겼고 어떻게 작동하는지에 대한 정보를 지닌 객체를 반환

<br>

컴포넌트는 데이터를 업데이트했을 때 단순히 업데이트한 값을 수정하는 것이 아니라, 새로운 데이터를 가지고 render 함수를 또 다시 호출한다.

이때 데이터를 지닌 뷰를 생성하게 되는데 render 함수는 반환하는 결과를 곧바로 DOM에 반영하지 않고, 이전에 만들었던 컴포넌트 정보와 비교한 후 둘의 차이를 알아내어 최소한의 연산으로 DOM 트리를 업데이트 하는 것이다.

전체 컴포넌트를 다시 렌더링하는 것처럼 보이지만, 최적의 사원을 사용하여 이를 수행하는 것이다.

---

컴포넌트 내부에는 또 다른 컴포넌트들이 들어갈 수 있다. 이때 render 함수를 실행하면 내부에 있는 컴포넌트 들도 재귀적으로 렌더링한다.

이렇게 최상위 컴포넌트의 렌더링 작업이 끝나면 지니고 있는 정보들을 사용하여 HTML 마크업을 만들고 내가 정하고자 하는 실제 페이지의 DOM 요소 안에 주입한다.

>렌더링 절차(컴포넌트를 실제 페이지에 렌더링 할 때)

1. 문자열 형태의 HTML 코드를 생성

2. 특정 DOM에 해당 내용을 주입하면 이벤트가 적용됨.

---

구체적으로 이해하기 위해서는 Virtual DOM을 알아야 한다고 해서 그 다음장에 있는 내용을 읽게 되었다.

> ### DOM

- Document Object Model의 약어로 \*\*객체로 문서 구조를 표현하는 방법\*\*

(방법으로는 XML이나 HTML로 작성한다.)

- 트리 형태라서 특정 노드를 찾거나 제거하거나 원하는 곳에 삽입할 수 있다.<br>

DOM API를 수많은 플랫폼과 웹 브라우저에서 사용하는 편이지만,

DOM의 치명적인 단점으로 \*\*종적 UI에 최적화되어 있지 않다는 것\*\*이다.

HTML은 자체적으로는 정적이지만, 자바스크립트를 사용하여 이를 동적으로 만들 수 있다.

규모가 큰 퀩 애플리케이션에서 DOM에 직접 접근하여 변화를 주다보면 느려지기 시작한다.

이를 두고 'DOM은 느리다' 라고 하지만, DOM 자체는 빠르다 대신 웹 브라우저가 CSS를 다시 연산하고, 레이아웃을 구성하고, 페이지를 리페인트 하는 과정에서 시간을 허비하게 되기 때문에 느리다고 표현을 하는 것이다.

DOM을 최소한으로 조작하여 작업을 처리하는 방식으로 개선할 수 있다고 한다.

여기서 리액트는 Virtual DOM 방식을 사용하여 DOM 업데이트를 추상화함으로써 DOM 처리 횟수를 최소하 하고 효율적으로 진행한다.

> ### Virtual DOM

리액트의 주요 특징 중 하나는 Virtual DOM을 사용하는 것이라고 한다.<br>

이것을 사용하면 실제 DOM에 접근하여 조작하는 대신, 이를 추상화한 자바스크립트 객체를 구성하여 사용한다고 한다.

리액트에서 데이터가 변하여 웹브라우저에 실제 DOM을 업데이트할 때는 세 가지 절차가 진행되는데 절차 과정은 아래와 같다.

1. 데이터를 업데이트하면 전체UI를 Virtual DOM에 리렌더링한다.

2. 이전 DOM에 있던 내용과 현재 내용을 비교한다

3. 바뀐 부분만 실제 DOM에 적용한다.

---

리액트와 Virtual DOM이 언제나 제공하는 것은 \*\*업데이트 처리 간결성\*\*이다.

2장

> ### 번들러 (웹팩)

- 리액트 프로젝트에서는 주로 \*\*웹팩\*\*을 사용하는 추세이다.

(편의성과 확장성이 다른 도구보다 뛰어나기 때문.)

- 모듈을 불러왔을 때 불러온 모듈을 모두 합쳐서 하나의 파일을 생성해준다.

- SVG 파일과 CSS 파일도 불러와서 사용 가능. - \*\*로더 기능 담당.\*\*

- \*\*css, file - loader\*\*는 각각 CSS 파일, 웹 폰트나 미디어 파일 등을 불러오며

- \*\*babel-loader\*\*는 자바스크립트 파일들을 불러오면서 최신 JS 문법으로 작성된 코드를 바벨이라는 도구를 사용하여 ES5 문법으로 변환해줌

(구 버전 웹 브라우저와 호환하기 위해서 ES5로 변환해주어야 한다.)

종류로는 웹팩, Parcel, browserify라는 도구가 있음. (도구마다 특성이 다르다.)

---

> ### JSX

- 자바스크립트의 확장 문법, XML과 매우 비슷하게 생겼다.

- JSX 안에서는 자바스크립트 표현식을 쓸 수 있다. 내부에서 코드를 { }로 감싸면 된다.

- 보기 쉽고 익숙하며, 높은 활용도를 가지고 있음.

- JSX를 여러 줄로 작성할 때 괄호로 감싸고, 한 줄로 표현할 수 있는 JSX는 감싸지 않는다.

### JSX 문법

1. 컴포넌트에 여러 요소가 있다면 반드시 부모 요소 하나로 감싸야한다.

- Virtual DOM에서 컴포넌트 변화를 감지해 낼 때 효율적으로 비교할 수 있도록 \*\*컴포넌트 내부는 하나의 DOM 트리 구조로 이루어져야 한다\*\*는 규칙이 있기 때문.

>```

function App() {

return (

<div> or <Fragment> or <>

<h1> 리액트 ! </h1>

<h2>작동하나요?</h2>

</div> or </Fragment> or </>

);

}

exprot default App;

```

`여기서 <div>와 같이 감싸져 있어야한다. `

`(div 요소를 사용하고 싶지 않다면 Fragment, <>을 사용해도 됨.)`

2. if 문 대신 조건부 연산자(삼항 연산자)를 사용해야 한다.

- 조건에 따라 다른 내용을 렌더링해야 할 때

2-1. JSX 밖에서 if 문을 사용하여 사전에 값을 설정

2-2. { } 안에 조건부 연산자를 사용

`(삼항 연산자 : 조건문 ? 참 값 : 거짓 값)`

>```

function App() {

const name = '리액트';

return(

<div>

{name === '리액트' ? (<h1>리액트 </h1>) : (<h2> 리액트가 아님 </h2>)}

</div>

);

}

exprot default App;

```

2-3. 특정 조건을 만족할 때만 렌더링할 때는 조건부 연산자에 AND 연산자(&&)를 사용한다.

`조건문 && 참 값`

> ```

function App() {

const name = 'React';

return(

<div>

{name === '리액트' && <h1>리액트 </h1>}

</div>

);

}

exprot default App;

```

`여기서 name은 한글로 리액트가 아닌 React 이므로 결과는 출력되지 않는다.`

3. 함수에서 undefined만 반환하여 렌더링하는 상황을 만들면 안된다.

> ```

import './App.css';

function App() {

const name = undefeined;

return name;

}

exprot default App;

```

`이러한 상황은 오류를 발생한다.`

따라서,

```

import './App.css';

function App() {

const name = undefeined;

return name || '값이 undefined입니다.';

}

exprot default App;

```

` OR 연산자를 사용하여 || '값이 undefined입니다.' 와 같이 값을 지정하면 오류를 방지할 수 있다.`

```

return <div>{name}</div>

```

`또한, return 값을 위 코드 처럼 JSX 내부에서 undefined를 렌더링 하는 것은 괜찮다.`

4. 리액트에서 DOM 요소에 스타일을 적용할 때는 문자열 형태로 넣는 것이 아니라 객체 형태로 넣어 주어야 한다.

>```

function App() {

const name = '리액트';

const style = {

backgroundColor: 'black'

color: 'aqua',

fontSize: '48px',

fontWeight: 'bold',

padding: 16 //단위를 생략하면 px로 지정됨.

};

return <div style={style}> {name}</div>;

}

exprot default App;

```

`background-color는 backgorundColor와 같이 - 문자를 없애고 카멜 표기법으로 작성된 것을 볼 수 있다.`

`추가로 font-size, font-weight도 카멜 표기법으로 작성되었다.`

미리 선언하지 않고 바로 style 값을 지정하고 싶을 경우에는

>```

function App() {

const name = '리액트';

return <div style=

backgroundColor: 'black'

color: 'aqua',

fontSize: '48px',

fontWeight: 'bold',

padding: 16 >

{name}

</div>;

}

exprot default App;

```

`이렇게 표현할 수 있다.`

5. JSX에서는 className으로 설정해주어야 한다.

- 일반 HTML에서 CSS 클래스를 사용할 때는 ```<div class="myclass"></div>```라는 속성을 설정한다.

> css 클래스를 작성한 후 아래 코드를 입력하고 저장한다.

```

.react {

backgroundColor: 'black'

color: 'aqua',

fontSize: '48px',

fontWeight: 'bold',

padding: 16

}

```

이것을 js파일에 적용하려면

```

import './App.css';

function App() {

const name = '리액트';

return <div className ="react">{name}</div>;

}

export default App;

```

`className`처럼 작성해주면 된다.

`(class로 값을 설정해도 적용되긴 하지만 Console 탭에 경고 문구가 띄워진다.)`

6. 태그는 닫아주어야 한다.

> - 태그 사이에 별도의 내용이 들어가지 않는 경우 self-closing 태그를 이용하여 선언과 동시에 닫는 태그를 적용한다 `ex) <input />`

7. 주석

> - JSX의 주석은 일반 자바스크립트와는 다르다.

`{/\* 주석처리 할 내용 \*/}` 처럼 작성한다.

<br>

- `//`, `/\* \*/`을 사용하게 되면 페이지에 고스란히 나타난다.