리액트를 다루는 기술 정리 - 박정모

> 2022-05-04

01장 - 리액트 시작

1.1 왜 리액트인가?

- 자바스크립트만으로 대규모 애플리케이션을 만들 수 있게 관리하는 프레임 워크들
  - `Angular`(google개발, but 너무 어려움), `Backbone.js`, `Derby.js`, `Ember.js`, `Ext.js`, `Knockback.js`, `Sammy.js`, `PureMVC`, `Vue.js`(Svelte와 비슷, but 제작자 망언으로 안쓰임)
  - 최신 `Svelte`
- 상기한 프레임워크들은 주로 `MVC`, `MVVM`아키텍쳐를 사용
  - `MVC` - Model - View - Controller
  - `MVVM` - Model - View - View Model
  - MVC, MVVM, MVW등의 구조는 모델과 뷰가 존재
    - `Model` - Data 관리 영역
    - `View` - 사용자에게 보이는 영역
    - 사용자에게서 어떤 작업을 입력받을 때 -> Controller는 Model 데이터를 조회, 수정 -> View에 반영

1.1.1 리액트 이해

용어	설명
컴포넌트	특정 부분이 어떻게 생길지 정하는 선언체 재사용이 가능
렌더링	사용자 화면에 View를 보여주는 것

- React.js는 데이터가 변할 때 마다, 어떤 데이터를 변화를 주는 것이 아니라 기존 뷰를 날려버리고 새로 렌더링
- 오직 View만 신경쓰는 라이브러리

1.1.1.1 초기 렌더링

- 어떤 프레임워크를 사용하든, 맨 처음 어떻게 보일지를 정하는 초기 렌더링이 필요
- React에서는 `render()`가 이를 담당(컴포넌트의 생김을 정의)

1.1.1.2 조화 과정

- React는 화면갱신시 View를 변형하는 것이 아니라 새로운 요소로 갈아끼움(`render()`가 담당)
- 화면갱신 순서
  1. 컴포넌트 데이터가 업데이트 되었을 때, 새로운 데이터를 가지고 `render()`를 다시 호출

2. `render()`가 반환하는 결과를 바로 DOM에 적용하는 것이 아니라, 이전에 만들었던 컴포넌트 정보와 현재 만들어진 결과를 비교
3. 둘의 차이를 알아내 최소한의 연산으로 DOM을 업데이트(갈아끼움)

1.2 리액트의 특징

1.2.1 Virtual DOM

1.2.1.1 DOM이란?

- DOM - Document Object Model = Tag에 의한 구조표현
- DOM은 동적 UI에 최적화되어 있지 않음. (JS로 동적으로 만듦)
- DOM 자체는 빠르나, 동적으로 변화를 줄 때마다 웹브라우저단에서 CSS재연산, 레이아웃 재구성, 페이지 리페인트 하는 과정이 반복되어 느리게 느껴짐
- DOM을 조작할 때마다 엔진이 웹페이지를 새로 그리는 단점을 방지하기 위하여 DOM을 최소한으로 조작, 처리하는 방식으로 개선할 수 있는데, React는 이를 Virtual DOM 방식으로 DOM 업데이트를 추상화하여 효율적으로 진행함

1.2.1.2 Virtual DOM?

Virtual DOM을 사용하면 실제 DOM에 접근, 조작하는 것이 아니라 이를 추상화한 JS객체를 구성하여 사용함

1. 데이터 업데이트시 전체 UI를 Virtual DOM에 리렌더링
2. 이전 Virtual DOM과 비교
3. 바뀐 부분만 실제 DOM에 적용

하나를 수정하면 DOM 전체를 리로딩하는 것이 아니기 때문에 가볍고 빠름

다만, React가 무조건 빠른 것이 아니라 `지속적으로 데이터가 변화하는 대규모 어플리케이션`에 최적화되어 있음

소규모 어플리케이션이나 최적화가 매우 잘 된 코드를 사용할 때는 React를 사용하지 않는 경우가 더 나을 수 있다.

1.3 작업환경 설정

사전 작업 순서
Node.js / npm , yarn 설치
코드 에디터 설치
Git 설치
creat-react-app

1.3.1 node.js와 npm

`Node.js`가 직접적으로 필요한 것이 아니라 프로젝트를 진행하는데 필요한 주요 도구들을 사용하기 위해 설치

- `babel` - ES6로 작성 -> `Webpack`으로병합 -> `babel`로 ES5로 변환
- `Webpack` - 모듈화된 코드를 한 파일로 합치고(번들링), 코드를 수정할 때마다 웹브라우저를 리로딩하는 등의 여러 기능 포함

#02장 - JSX

2.1 코드 이해하기

```
import logo from './logo.svg';
import './App.css';
```

- `import` - 다른 js파일 불러와 사용하는 구문
  - 이렇게 모듈을 불러와서 사용하는 것은 브라우저에는 없던 기능(브라우저가 아닌 환경에서 JS를 실행할 수 있게 하는 Node.js에서 지원하는 기능)
  - 이러한 기능을 브라우저에서도 사용할 수 있게 `bundler`(React는 `Webpack`)를 사용(분할된 js를 import해서 조립)
  - 이렇게 조립된 js파일을 웹브라우저에서 사용

```
function App() {
  return (
    <div className="App">
      <header className="App-header">
        <img src={logo} className="App-logo" alt="logo" />
        <p>
          Edit <code>src/App.js</code> and save to reload.
        </p>
        <a
          className="App-link"
          href="https://reactjs.org"
          target="_blank"
          rel="noopener noreferrer"
        >
          Learn React
        </a>
      </header>
    </div>
  );
}
```

- Return하는 내용은 HTML이 아닌 JSX코드

2.2 JSX?

- JS의 확장문법

```
function App() {
    return (
        <div>
            Hello <b>react</b>
        </div>
    );
}
```

위 코드가 JSX인데, 이를 JS로 작성할 경우

```
function App() {
    return React.createElement("div", null, "Hello ", React.createElement("b",
null, "react"));
}
```

이렇게 된다. 매우 복잡하고 불편.

> 그러나, JSX는 공식적인 JS문법은 아님. 바벨을 통해 개발자들이 임의로 만든 문법을 사용할 수 있는 것

2.3 JSX의 장점

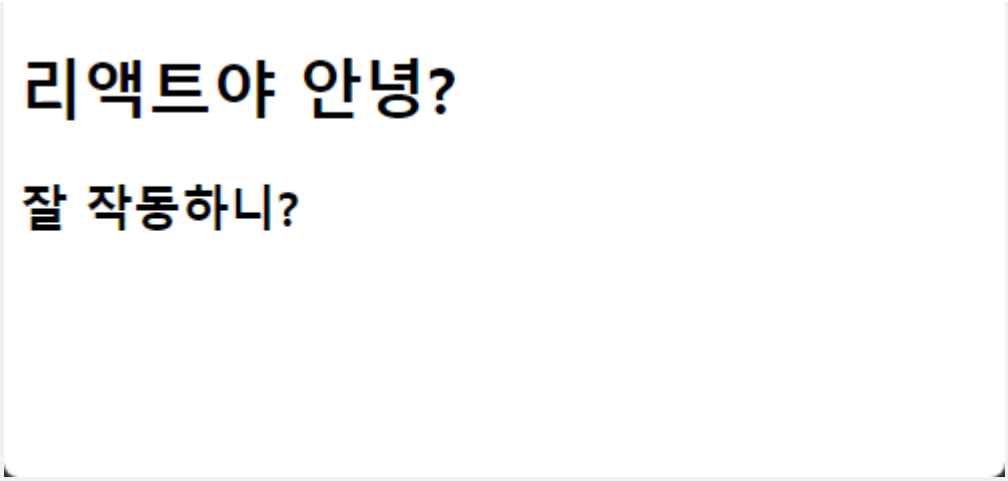
보기 쉽고 익숙함

2.4 JSX 문법

2.4.1 감싸인 요소

- 컴포넌트에 여러 요소가 있다면 반드시 하나의 부모 태그로 감싸야 함

> 실습 결과 (src/App.js)
>
> ```
> function App() {
>  return (
>    
>      <h1>리액트야 안녕?</h1>
>      <h2>잘 작동하니?</h2>
>    
>  );
> }
>
> export default App;
> ```



2.4.2 자바스크립트 표현

- JSX내부에서 {}로 감싼 후 JS표현식을 사용할 수 있다.

2.4.3 If문 대신 조건부 연산자

- JSX내부의 JS표현식에서는 if문 사용이 불가
- 조건부 연산자(삼항 연산자) 이용
- 조건이 참일 때만 렌더링을 할 경우 -> AND 연산자(&&)
- 조건이 거짓일 때만 렌더링을 할 경우 -> OR 연산자(||)
    - 여기서 false의 조건이 0일 경우, 0은 렌더링 됨

2.4.6 인라인 스타일링

React에서 DOM에 스타일 적용시 문자열 형태가 아닌 객체(json)형태로 주입

- 이 때 속성을 카멜 표기법으로 작성한다(backgroundColor)

실습 결과

```
function App() {
   const name = "리액트";
   return (
           style={{
               backgroundColor: "black",
               color: "#00ff00",
               fontSize: "48px",
               fontWeight: "bold",
               padding: "15",
           }}
       >
         {name}
       </div>
   );
}

export default App;
```

2.4.7 class 대신 className

JSX는 js의 응용언어이므로 기본으로 할당되어있는 class 대신 className을 사용한다.

2.4.8 꼭 닫아야 하는 태그

`<hr />`이나 `<input />`처럼 HTML에서는 닫지 않아도 정상 적용 되던 태그들도 JSX에서는 확실히 닫아야 compile오류가 발생하지 않는다.

2.4.9 주석

- JSX 안에서 주석을 작성하는 방법은 `{/* ... */}`이다.
- 태그 안에서 줄바꿈을 할 경우 주석 작성이 가능한데, 태그 안에서는 일반 JS처럼 `//`를 이용한다.
- 태그 밖에서 `//`이나 `/** */`등 JS 주석 문법을 작성할 경우 화면에 그대로 노출됨