7/26 월요일 실습강의 리액트 보조자료

```
작성자      : 코치 김성일
최종 수정일 : 2021.07.26
```

React란?

- 페이스북에서 개발한 **UI를 구축하기 위한 자바스크립트 라이브러리**입니다.
- **컴포넌트라는 요소를 이용하여 복잡한 UI를 독립적인 단위로 쪼개어 구현**합니다.

등장배경

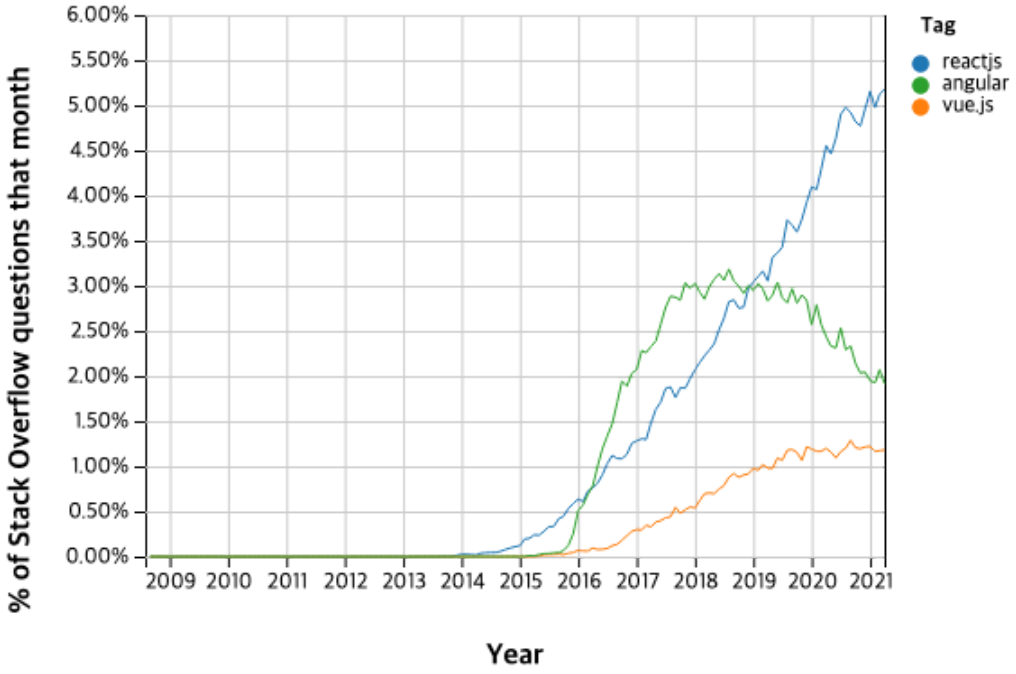
- 현대로 오면서, **성능이 좋은 동적 UI를 구축해야하는 상황**이 생겼습니다.
- 페이지가 늘어나고 구조가 복잡해지면서 다양한 라이브러리와 프레임워크의 중요성이 대두되었습니다.

React(UI 라이브러리 혹은 프레임워크)를 사용해야하는 이유

- 각 페이지마다 값을 관리해야하고, 사용자의 응답에 따라 인터페이스가 지속적으로 변해야 합니다.
- 기존의 순수한 html과 javascript로 이것을 구현하고 관리하기 어렵습니다.
- UI 라이브러리 혹은 프레임워크는 이 어려움을 해결해줍니다.

여러 UI 프레임워크와 라이브러리 중 React를 사용해야하는 이유?

- 현재 **가장 점유율이 높은 UI 라이브러리**입니다.
    - 네카라쿠배당토 모두 React를 사용하고 있습니다.
- **StackOverflow** 질문 갯수 비교 (vs Vue.js && Angular)



- 관련 **Github Repository** 수
  - **React : 220만**
  - Angular : 77만
  - Vue : 55만

React의 특징

- **Virtual DOM**을 사용하여, 개발자의 작업속도를 높여줍니다.
  - **Virtual DOM**이란 가상적인 표현을 메모리에 저장하고 **ReactDOM**과 같은 라이브러리에 의해 실제 **HTML DOM**과 동기화하는 프로그래밍 개념입니다.
- **컴포넌트** 단위로 개발을 하기 때문에, 재사용이 가능하여 개발 시간을 절약시켜줍니다.
- **단방향 데이터 흐름을** 사용하기 때문에 안정적인 코드 작성을 가능하게 합니다.
  - 각 데이터는 항상 일정한 곳에 존재하고, 그곳에서만 변경이 가능한 것을 의미합니다.
- **Hooks**라는 기술을 이용하여, 컴포넌트의 상태(여러 값들)를 쉽게 관리할 수 있습니다.
- **오픈소스**입니다. 여러 커뮤니티에 React의 정보를 쉽게 얻을 수 있습니다.

- 오픈소스이기때문에, 여러 개발 도구가 개발되어 있습니다.

React와 자바스크립트의 차이점

React를 사용하면 React의 방식에 따라 코드를 작성해야하기 때문에, 더 관리하기 쉽고 유지보수하기 쉬운 코드를 작성할 수 있습니다.

사용자 인터페이스를 만드는법 비교

- **순수 Javascript만 사용했을 때**
  - UI는 보통 HTML만 이용하여 구현합니다.
  - 따로 Javascript가 추가적으로 필요하지 않습니다.

```
<div>
    <h1>Grocery List</h1>
    <ul>
        <li>Milk</li>
        <li>Bread</li>
        <li>Eggs</li>
    </ul>
</div>
```

- **React를 사용했을 때**
  - React는 JSX를 이용하여 UI를 정의합니다.
    - JSX란?
      - **J**ava **S**cript e**X**tension
      - Reaxt에서 Element를 생성하는 Javascript 확장 문법
      - Javascript 안에서 HTML 문법을 사용해서 UI를 구성할 수 있게 도와주는 Javascript 문법
  - `GroceryList` 컴포넌트는 `ReactDOM`을 통해 UI로 렌더링되어 화면에 출력됩니다.

```
function GroceryList(props) {
    return (
        <div>
            <h1>Grocery List</h1>
            <ul>
                <li>Milk</li>
                <li>Bread</li>
                <li>Eggs</li>
            </ul>
        </div>
    )
};
```

앱에서 기능이 분할되는 방식

- **순수 Javascript만 사용했을 때**
  - HTML과 Javacript를 따로 작성합니다.
  - 화면과 그에따른 기능이 서로 다른 곳에 위치하기에, 유지보수하기 어렵습니다.

```
<!-- item.html -->

<script src="item.js"></src>
<div>
    <h1>Grocery List</h1>
    <ul id="grocery-list">
        <li>Milk</li>
        <li>Bread</li>
        <li>Eggs</li>
    </ul>
</div>
```

```
// item.js

function addItem() {
  // todo : add item at grocery-list
}
```

- **React를 사용했을 때**

- UI에 필요한 코드와 UI 요소를 같이 정의하여 하나의 파일로 유지할 수 있습니다.
- 코드가 복잡해지고 많아지더라도, 유지보수가 어렵지 않습니다.

```
// GroceryList.js

function GroceryList(props) {
    const [items, setItems] = useState(["Milk", "Bread", "Eggs"]);
    const [itemName, setItemName] = useState('');

    function addItem() {
       // todo : add item at grocery-list
    }

    return (
        <div>
            <h1>Grocery List</h1>
            <ul>
                <li>Milk</li>
                <li>Bread</li>
                <li>Eggs</li>
            </ul>
        </div>
    )
};
```

데이터를 저장하는 방식

- **순수 Javascript를 사용했을 때**
  - 사용자의 데이터가 DOM에 저장됩니다.
  - 사용자가 데이터를 입력하면, DOM을 통해 수동으로 데이터를 가져옵니다.

```
<!-- item.html -->

<script src="item.js"></src>
<div>
    <h1>Grocery List</h1>
    <ul id="grocery-list">
        <li>Milk</li>
        <li>Bread</li>
        <li>Eggs</li>
    </ul>
    <input type="text" name="item" />
    <button onclick="addItem('test')">Add Test</button>
</div>
```

```
// item.js

function addItem() {
  let list = document.getElementById('grocery-list');
  const itemName = document.getElementById('item').value;

  const newItem = document.createElement('li');
  newItem.innerText = itemName;
  list.appendChild(newItem);
}
```

- **React를 사용했을 때**
  - 사용자가 데이터를 입력하면, 컴포넌트가 입력을 기반으로 자신의 상태를 관리하고 업데이트합니다.
  - 하지만, 미리 상태를 정의하고 상태가 바뀌었을 때 실행 될 함수를 정의해놓아야합니다.
  - 이렇게 DOM에 의존하지않고, 데이터를 Javascript를 통해 관리할 수 있다는 장점이 있습니다.

```
// GroceryList.js

function GroceryList(props) {
    const [items, setItems] = useState(["Milk", "Bread", "Eggs"]);
    const [itemName, setItemName] = useState('');

    function addItem() {
       // todo : add item at grocery-list
    }

    return (
        <div>
            <h1>Grocery List</h1>
            <ul>
                {items.map(item => (
                  <li key={item}>{item}</li>
                ))}
            </ul>
            <input type="text" onChange="{e => setItemName(e.target.value)}" />
            <button onClick={addItem}>Add React</button>
        </div>
```

```
    )
};
```

UI 업데이트 방법

- **순수 Javascript를 사용했을 때**
  - DOM을 통해 찾은 Button Element에 EventListener를 추가합니다.
  - 그 후 직접 DOM을 통해 새로운 항목을 생성하고, 목록 끝에 추가합니다.

```
<!-- item.html -->

<script src="item.js"></src>
<div>
    <h1>Grocery List</h1>
    <ul id="grocery-list">
        <li>Milk</li>
        <li>Bread</li>
        <li>Eggs</li>
    </ul>
    <input type="text" name="item" />
    <button onclick="addItem()">Add Test</button>
</div>
```

```
// item.js

function addItem() {
  let list = document.getElementById('grocery-list');
  const itemName = document.getElementById('item').value;

  const newItem = document.createElement('li');
  newItem.innerText = itemName;
  list.appendChild(newItem);
}
```

- **React를 사용했을 때**
  - 데이터의 상태를 유지하는 React의 특징 때문에, 각 요소는 함수를 통해 JSX로 렌더링할 수 있습니다.
  - 버튼을 정의하고, 버튼을 눌렀을 때 작동할 함수를 정의합니다.

- 이 함수를 통해, items state에 데이터를 추가하면 별도의 DOM 조작없이 화면에 추가 된 요소가 렌더링 됩니다.

```
// GroceryList.js

function GroceryList(props) {
    const [items, setItems] = useState(["Milk", "Bread", "Eggs"]);
    const [itemName, setItemName] = useState('');

    function addItem() {
       setItems([...items, itemName]);
    }

    return (
        <div>
            <h1>Grocery List</h1>
            <ul>
                {items.map(item => (
                  <li key={item}>{item}</li>
                ))}
            </ul>
            <input type="text" onChange="{e => setItemName(e.target.value)}" />
            <button onClick={addItem}>Add React</button>
        </div>
    )
};
```

로컬에서 React 실습하기

이제 저희는 **CRA(Create-React-App)**를 사용하여 React 앱을 구축할 것입니다.

CRA는 **리액트 프로젝트를 시작하는데 필요한 개발 환경을 세팅 해주는 도구**입니다.

기본적인 **React는 UI를 만드는 기능만 제공**하고, 웹 애플리케이션을 띄우는 등의 다른 기능은 직접 개발자가 구축해야한다는 단점이 있습니다.

이러한 단점을 보완하기위해 나온 것이 CRA로, **CRA를 이용하면 명령어로 리액트 개발환경을 전부 구축해줍니다**.

- CRA 설치하기

  패키지 매니저인 yarn을 통해 CRA을 설치합니다.

```
yarn global add create-react-app
```

- 설치되었는지 확인하기

```
create-react-app --version
```

- CRA로 새로운 프로젝트를 하나 생성해보겠습니다.

```
create-react-app hello-world
```

- CRA로 생성한 리액트 App 실행하기

```
yarn start
```