React

<React의 특징(장/단점) 및 용어 정리>

<Vitural DOM>
먼저 DOM은 Document Object Model의 약자로, HTML, XML 문서의 프로그래밍 인터페이스를 의미합니다. DOM은 HTML, XML 문서의 구조와 콘텐츠를 객체로 표현하여, JavaScript를 사용하여 브라우저의 DOM에 접근하고 조작할 수 있도록 합니다.

Virtual DOM은 Document Object Model의 경량 JavaScript 표현입니다. 가상 DOM을 업데이트하는 것이 실제 DOM을 업데이트하는 것보다 비교적 빠릅니다. 실제 DOM보다 업데이트가 빠르기 때문에 웹사이트를 구축하는데 있어서 유용합니다.

<JSX>
.jsx는 JavaScript eXtends의 줄임말로 기존 JS에서 확장된 것

<Element Rendering>
Element = 가장 작은 블록(가장 작은 요소)들을 의미, DOM에서 사용하는 용어
불변성을 가지고 있다.
Element를 생성 후에는 children이나 attributes를 바꿀 수 없음.
붕어빵 틀에 반죽 넣고 구워져 나오면 완성된 붕어빵이 되고 완성된 붕어빵은 완성된 것으로 끝나는 격과 같음.

를 사용하여 Root DOM Node를 생성.
Root DOM Node는 DOM 트리의 최상위 노드를 말한다.
웹 애플리케이션 개발에서 중요한 개념이다.
DOM 트리를 조작할 때 Root DOM Node를 사용하여 DOM을 관리한다.

Element의 불변성으로 인해서 기존의 Element를 변경하기 어렵지만, 새로운 Element를 생성하여 Rendering된 Element 업데이트 할 수 있습니다.

<Component>
Component란 무엇인가?
구성요소란 뜻이며 마치 레고의 블록 하나하나를 조립하여 완성된 모양을 잡아나가듯 하나의 컴포넌트가 모이고 모여서 웹사이트를 만들기에 Component가 React에서는 굉장히 중요합니다.

Component에서는 함수형 컴퍼넌트와 클래스형 컴퍼넌트가 있습니다.

함수형 컴퍼넌트(Function Component)는 React에서 가장 많이 사용되는 Component의 한 종류
JavaScript 함수를 사용하여 작성되며, props를 인수로 받아 element를 반환하는 컴퍼넌트 형태.
간단한 컴포넌트를 작성할 때, 상태를 관리하지 않을 때, 성능이 중요할 때 주로 함수형 컴퍼넌트를 사용합니다.

클래스형 컴퍼넌트(Class Component)는 JavaScript class를 사용하여 작성되며, state를 관리할 수 있습니다.
복잡한 컴포넌트를 작성할 때, 상태를 관리할 때, 성능이 중요하지 않을 때 주로 클래스형 컴퍼넌트

를 사용합니다.

현재의 개발에 있어서는 함수형 컴퍼넌트 기반을 많이 사용하는 추세입니다.

컴퍼넌트 이름은 항상 대문자로 시작해야한다.
소문자로 시작하게 되면 컴퍼넌트를 DOM 태그로 인식합니다.

컴퍼넌트 추출을 통해서 큰 컴퍼넌트를 여러개의 컴퍼넌트로 추출할 수도 있습니다.
재사용성이 올라가고, 컴퍼넌트의 기능과 목적이 명확하고 props도 단순하여져서 가독성 등 여러면에서 유용합니다.

<재사용성>
재사용성은 객체 지향 프로그래밍에서 쓰이는 개념으로 소프트웨어 개발에 있어 중요함.
재사용성이란 말 그대로 다시 사용이 가능한 성질을 말합니다.
재사용성의 특징을 이용함으로 개발기간을 단축, 유지보수에 용이하기에 모듈 수정 및 버그 수정이 쉽습니다.

<React Native>
모바일 환경 UI FrameWork를 사용하여 모바일 앱을 개발 할 수 있는 UI 라이브러리

방대한 학습과 높은 상태 관리의 복잡도는 React의 단점입니다.

<Props, State>
props - 부모로부터 받아오는 값
props는 React에서 컴포넌트 사이에 데이터를 전달하는 데 사용되는 객체입니다.
부모 컴포넌트에서 자식 컴포넌트로 props를 전달하면 자식 컴포넌트에서 props의 속성을 사용하여 데이터를 사용한다.
props는 읽기 전용, props를 변경하려면 부모 컴포넌트에서 props의 값을 변경해야 합니다.

부모 컴포넌트에서 자식 컴포넌트에 데이터를 제공해야 하는 경우
자식 컴포넌트에서 부모 컴포넌트의 데이터를 사용할 수 있도록 해야 하는 경우 props를 사용을 합니다.

State는 현재 내 자신에서 받아오는 값으로 컴퍼넌트의 상태를 의미한다.
React Component의 변경 가능한 데이터를 State라고 합니다.
랜더링이나 데이터 흐름에 사용되는 값만 포함시켜야 합니다.
JS 객체 = State, 직접적인 변경이 불가능, State에서도 클래스 컴퍼넌트와 함수 컴퍼넌트로 구분이 됩니다.
클래스 컴퍼넌트는 생성자에서 모든 state를 한번에 정의 할 수 있습니다. state를 변경하고자 할 때는 setState() 함수를 사용해야 한다.
함수 컴퍼넌트는 useState()를 사용하여 각각 state를 정의하고 각 state별로 주어지는 set함수를 사용해 state 값을 변경하는 것이 특징이다.

<컴퍼넌트의 생명주기>
컴퍼넌트는 생성되고 사라지는 주기가 있습니다. 이를 컴퍼넌트의 생명주기라고 한다.

컴퍼넌트 생성(Mount)

컴퍼넌트가 랜더링 되고 컴퍼넌트 생성자(Constructor)가 실행되면서 state를 정의하게 되고, 컴퍼넌트 랜더링 이후 componentDidMount()가 호출된다.

컴퍼넌트 업데이트
컴퍼넌트는 생명 주기동안 여러번 업데이트가 되면서 랜더링이 됩니다.
컴퍼넌트의 props가 업데이트 되면서 setState() 함수 호출에 의해 state가 변경이 됩니다.
또는 forceUpdate()라는 강제 업데이트 함수가 호출되면서 componentDidUpdate()가 호출 됩니다.

컴퍼넌트 소멸(Unmount)
컴퍼넌트도 생명 주기가 다 되어 사라지게 됩니다.
상위 컴퍼넌트에서 현재 컴퍼넌트를 더 이상 화면에 표시하지 않게 되면 컴퍼넌트가 소멸이 되게 됩니다. 이를 Unmount라고 표현합니다.
Unmount가 되기 전 componentWillUnmount() 함수가 호출이 됩니다.

컴퍼넌트는 계속해서 존재하는 것이 아니라 시간의 흐름에 따라 생성되고 업데이트되고 사라지고 하는 과정을 반복하게 됩니다.
이것이 컴퍼넌트의 생명주기라고 합니다.

<훅(Hook)>
훅은 갈고리라는 의미이지만, 프로그래밍에서는 원래 존재하는 어떤 기능에 마치 갈고리를 거는 것처럼 같이 수행되는 것을 의미합니다.
React에서는 state와 컴퍼넌트 생명주기 기능에 갈고리를 걸어서 원하는 시점에 정해진 함수를 실행하도록 만든 것을 의미합니다.
훅의 이름은 모두 use로 시작

useState()
가장 많이 사용되는 훅의 종류로 state를 사용하기 위한 훅입니다. 함수형 컴퍼넌트에서는 기본적으로 state라는 것을 제공하지 않습니다. 그렇기에 함수형 컴퍼넌트에서 클래스 컴퍼넌트같이 state를 사용하기 위해서 useState()를 사용합니다.
const [변수명, set함수명] = useState(초깃값);의 형태로 사용
변수 각각에 대해 set함수가 따로 존재합니다.

useEffect()
useState()와 더불어 가장 많이 사용하는 훅의 형태
서버에서 데이터를 받아오거나 수동으로 DOM을 변경하는 등의 작업(사이드 이펙트)을 주로 수행하는 훅입니다. DOM이 변경된 이후에 해당 이펙트 함수를 실행하라는 의미이기에 useEffect() 훅만으로 클래스 컴퍼넌트의 생명주기 함수들과 동일한 기능을 수행합니다.
useEffect(이펙트 함수, 의존성 배열); 의 형태로 사용
의존성 배열안에 있는 변수 중 하나라도 값이 변경되었을 때 이펙트 함수가 실행된다.
의존성 배열에 빈 배열([])을 넣으면 생명주기에 단, 한 번만 실행이 됩니다.
의존성 배열을 생략시에는 컴퍼넌트가 업데이트 될 때마다 호출이 된다.
선언된 컴퍼넌트의 props와 state에 접근할 수 있습니다.
useEffect()에서 리턴하는 함수는 컴퍼넌트 마운트가 해제될 때 호출됨.

그 외에도 useMemo(), useCallback(), useRef()가 있다.

<Hook의 규칙>

1) 무조건 최상위 레벨에서만 호출해야 합니다.
반복문이나 조건문 또는 중첩된 함수들 안에서 훅을 호출하면 안 됩니다.
훅은 컴퍼넌트가 랜더링 될 때마다 매번 같은 순서로 호출되어야 합니다.

2) 함수 컴퍼넌트에서만 훅을 호출해야 합니다.
함수 컴퍼넌트나 커스텀 훅에서만 훅이 호출이 된다.
커스텀 훅 - 기본적으로 제공된 훅 이외에 추가적으로 필요한 기능을 맏르어서 사용하는 훅
이름이 use로 시작하고 내부에서 다른 훅을 호출하는 하나의 JS 함수, 중복되는 로직을 커스텀 훅을 이용하여 추출하여 재사용성을 높일 수 있음.

<이벤트 핸들링>
이벤트는 사용자가 버튼을 클릭하는 등에 의해 웹 브라우저에서 발생하는 사건을 의미

이벤트 처리 방법 중 Event Handler가 주로 많이 쓰이므로 중요함
어떤 이벤트가 발생했을 때 해당되는 이벤트를 처리하는 함수가 Event Handler이다.
Event Listener라고도 부른다.

[Event Listener 사용방법]
클래스 컴퍼넌트 형태로 클래스의 함수로 정의하고 생성자에서 바인딩해서 사용, 클래스 필드 문법으로 사용한다.
함수 컴퍼넌트 형태로 함수 안에 함수로 정의 혹은 화살표 함수(Arrow Function)를 사용해서 정의를 합니다.

<구조 분해 할당>
구조 분해 할당(Destructuring Assignment)은 JavaScript에서 배열이나 객체를 해체하여 그 값을 개별 변수에 할당하는 문법입니다. 이를 통해 코드를 더 간결하게 작성하고 변수를 쉽게 선언하고 초기화할 수 있습니다.

<스프레드 연산자>
스프레드 연산자(...)는 JavaScript에서 객체와 배열을 확장하거나 병합하는 데 사용되는 유용한 기능입니다. 이 연산자를 사용하면 기존 객체나 배열의 요소를 새로운 객체나 배열에 쉽게 복사하거나 추가할 수 있음.

[React-Router-DOM]
라우팅은 사용자가 웹 애플리케이션에서 특정 페이지로 이동할 수 있도록 하는 기능이다.
경로 기반의 라우팅을 제공하며 네비게이션과 하이퍼링크의 기능을 제공한다.
HTML5에서 <a>태그로 링크를 한 것을 나타냈지만 리액트에서는 React-Router-DOM을 사용한다.
라우터 구성으로 앱의 URL과 렌더링할 컴포넌트를 매핑한다.

BrowserRouter 컴포넌트를 사용하여 라우터를 구성합니다. Switch 컴포넌트를 사용하여 여러 경로를 매핑합니다. Route 컴포넌트를 사용하여 각 경로에 대한 컴포넌트를 지정하여 라우터를 구성하고 Link 컴포넌트를 사용하여 링크를 생성합니다. to 속성을 사용하여 링크를 클릭하면 이동할 경로를 지정하여 네비게이션 컴퍼넌트를 만듭니다.

<백엔드와 연동(axios, STS4, MySQL 활용)>

1) 패키지 설치후 React에서
import axios from "axios";
axios로 부를 내용들을 함수형 컴퍼넌트나 클래스형 컴퍼넌트로 정의
axios.get, axios.post를 적절히 활용하여서 get방식과 post방식으로 웹을 부를 수 있도록 해야 함.

2) 프록시를 설정하면 개발 서버가 외부 서버에 있는 API를 사용하는 것처럼 요청을 전달할 수 있다.
package.json에 proxy 설정(백엔드 서버의 포트) 추가를 합니다.
"proxy": "http://localhost:8686/"와 같은 형태로 추가

3) STS4에서 application.properties 설정 및 패키지와 클래스 구현
server.port=8081
spring.datasource.url=jdbc:mysql://localhost:3306/mydb
spring.datasource.username=root
spring.datasource.password=root
이와 같은 형태로 하고 Repository, Entity 생성

[기타 내용]
<React 패키지 설치> - axios, react-router-dom, bootstrap, react-bootstrap
npm install axios react-router-dom bootstrap react-bootstrap

https://react-bootstrap.netlify.app/ - React BootStrap 사용법

package.json - npm 설치 버전 확인하는곳