**[리액트란]**

-> 프론트엔드 JS 라이브러리다!

-> 사용자 UI의 구성요소를 빌드하기 위한 도구

(빌드란? html구조와 언어에 데이터를 바인딩하여 배포하는 작업을 이름)

**[ 작동원리 ]**

-> 가상돔(Virtual DOM)을 사용하여 최소의 html리소스를 사용함으로

빠르고 쉽게 UI화면을 구성한다!

-> 가상돔은 실제 DOM을 변경하기 전에 메모리상에서 구현하는 오브젝트 DOM이다!

-> 변경사항을 단위 별로 한번에 업데이트하기위한 수단으로써의 도구다!

**[ 리액트 구현의 2가지 스타일 ]**

# 스타일1 : 리액트는 JS 라이브러리 이므로 필요한 부분에만 적용한다!

-> 라이브러리는 CDN방식으로 구현 (별도의 설치가 필요 없다!)

# 스타일2 : 리액트는 **SPA(Single Page Application)**이므로 Node.js 등을

사용하여 한 페이지로만 구현하는 웹을 만들 수 있다!

-> Node.js, 리액트 SPA 개발환경 설정이 필요함!

---------------------------------------------

참고 사이트 : <https://ko.legacy.reactjs.org/>

html 태그와 JS문법을 따옴표없이 사용하는 JSX문법을 사용한다!

JSX(Javascript XML) 문법을 쓰는 파일은 .js대신 .jsx확장자사용!

-> **return 키워드 사용은 따옴표없이 바로 소괄호를 묶어 사용 (소괄호 생략 가능!)**

**[[ 주의사항 ]]**

1. 함수를 만들 경우 **반드시 대문자로 시작**해야 호출됨!(정해진 규칙)

2. 홀로 태그를 사용할 때는 마지막에 스스로 닫기를 꼭 해줌!(/>)

3. 함수 내부의 리턴 값으로 만든 요소는 반드시 최상위요소가 하나여야 함!

-> div와 같은 태그로 최상위를 만들어서 내보내기를 한다

그러나 이런 불필요한 태그 구조를 원치 않는 경우

<React.Fragment></React.Fragment>를 사용하면 내보낼 때

하나로 묶는 역할만 하고 실제로 태그는 출력 안됨

(#참고 : CLI환경(개발환경)에서는 빈 태그를 대신 사용 가능함 <></> )

* **jsx 파일에서 만든 함수를 생성자 함수**라고 함
* 생성자 함수를 만들어서 사용
* 대문자로 시작하는 이름을 사용한다
* 목적 : 코드를 만들어서 리턴한다

가상돔을 세팅하는 리액트 객체를 호출한다! -> ReactDOM

render() -> 요소를 변경하는 메서드

**[호출형태]**

ReactDOM.render(요소를 리턴하는 함수 명으로 된 홀로 태그, 대상 요소)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

예컨데 함수명이 MyFirstReact이므로

요소를 리턴하는 함수명으로 된 홀로태그는? <MyFirstReact />

그리고 대상 요소는 JS문법으로 요소를 선택함!

document.querySelector("#root")

ReactDOM.render(<MyFirstReact />, document.querySelector("#mydiv"));

**[ 리액트 컴포넌트 ]**

- 컴포넌트는 HTML요소를 반환하는 함수다!

**[ 특징 ]**

1. 컴포넌트는 독립적이고 재사용이 가능한 코드 집합

2. JS함수와 비슷하지만 HTML코드 반환이 필수라는 점이 다름!

(특정 모듈로 구성된 HTML코드를 리턴하는 객체)

3. 컴포넌트는 다음 2가지로 생성 가능함

1) 클래스형 컴포넌트

2) 함수형 컴포넌트

-> 클래스형 컴포넌트는 리액트 초 중기에 주로 사용되었으나...

React 16.8버전에서 Hooks와 함께는 더 이상 사용되지 않는다!

Hooks는 함수형 컴포넌트에서만 사용가능하다!

**[ 첫번째 컴포넌트 만들기! ]**

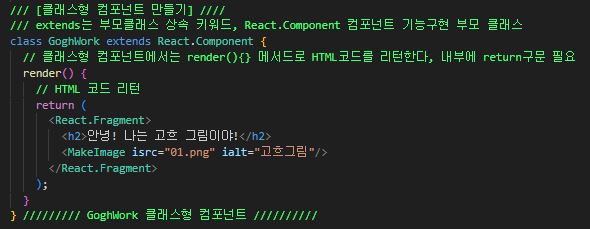
- 리액트 컴포넌트 이름은 반드시 **첫 글자를 대문자**로 만든다! (안 지키면 적용 안됨!!!)

[ 클래스 컴포넌트 ]: 클래스 컴포넌트에서는 extends React.Component 상속문이 포함 돼야함!

-> 컴포넌트에서도 메서드가 필요함, render() 메서드는 HTML을 반환함

(함수형 컴포넌트의 return 키워드를 사용할 수 있는 역할을 함!)

1. **클래스형 컴포넌트**



1. **함수형 컴포넌트**



[ 이미지 생성 공통 컴포넌트] : 변수를 설정하지않아도 비구조화 할당(구조분해할당)을 사용하여 전달 객체를 개별 변수 값으로 처리할 수 있다.

-> **비구조화 할당 방식(=구조분해할당방식)** =>> 쓰는 이유는 객체가 없기 때문

하나의 변수에는 전달된 값이 보낼 때 설정된 객체로 전달된다 {속성:값, 속성:값,...}

순서를 바꿔서 값을 보내도, 이름을 아무거나 써도 노 상관

1. function MakeImage(props){

return (<img src={"./images/"+props.isrc} alt={props.ialt} />) ;}

1. function MakeImage({ialt, isrc}){

return (<img src={"./images/"+isrc} alt={ialt} />) ;}

1. function MakeImage({ialt:ㅋㅋ, isrc:ㅎㅎ}){

return (<img src={"./images/"+ㅎㅎ} alt={ㅋㅋ} />); }

[ Props 사용하기 ]: props는 properties 에서 나온 말. 속성들...

즉, 변수에 값을 할당하여 전달하는 방법. 함수의 전달 값과 같고 속성으로 컴포넌트에 보낸다!

-> props라는 변수명은 마음대로 지을 수 있다

-> 컴포넌트에 {}를 사용하면 개별적인 속성 변수를 전달받을 수 있다(구조 분해 할당 방식)

# 컴포넌트 내부에서 다른 컴포넌트를 호출 할 수 있다!

**[ 컴포넌트의 파일 분리 ]**

리액트는 코드를 재사용하는 것이므로 컴포넌트를 별도의 파일로 분할 하는 것이 일반적이다!

{분할 방법}

1. jsx의 새 파일을 생성한다.

2. 대문자로 시작하는 컴포넌트를 구현한다.

3. 분할 구현된 jsx파일을 import하여 호출한다.

-> 일반적으로 js파일 상단에 import 키워드로 불러오면 되는데 지금 사용하는 CDN방식의 바벨모듈에서는 주의 사항이 있으니 참고 바란다!(아래 참고)

[ 바벨을 사용할 때 모듈로 파일 호출 시 주의사항! ]

설치 형이 아닌 CDN방식의 바벨은 호출 셋 업의 시차로 바로 모듈을 호출하면 에러가 발생한다!

따라서 모듈을 사용할 파일을 아래와 같은 형식으로 메인 html 상단에 호출해 줘야만 한다!!!

-> ★★★상단에 모듈화 한 JS를 먼저 불러준다!

<script src="모듈화 한 js"

data-plugins="transform-es2015-modules-umd"

type="text/babel"></script>

-> 아래쪽에 모듈을 호출하는 JS를 불러준다!

<script src="모듈을 호출하는 JS"

data-plugins="transform-es2015-modules-umd"

type="text/babel"></script>

->>> 위의 호출 속성 중 기본적으로

type="text/babel" 은 당연히 해야하고

->>> 여기에 더하여 하나의 속성을 추가한다!

data-plugins="transform-es2015-modules-umd"

이 속성과 값이 바벨에서 모듈을 사용하게 하는 es2015. 즉, ES6버전에서의 모듈 문법을 사용하게끔 해준다!