https://velog.io/@sdc337dc/0.%ED%81%B4%EB%9E%98%EC%8A%A4%ED%98%95-%EC%BB%B4%ED%8F%AC%EB%84%8C%ED%8A%B8

**선언방식**

함수형

import React from ‘react’;

import ‘./App.css’;

function App() {

const name = ‘react’;

return <div calssName =’react’> {name} </div>

}

클래스형

import React, {Component} from ‘react’;

class App extends Component {

render() {

const name = ‘react’;

return <div className=”react”> {name} </dv>

}

}

**일반적 차이**

클래스형

state, liftCycle 관련 기능 사용이 가능하다

메모리 자원을 함수형보다 조금 더 사용함

임의 메소드를 정희할 수 있음

함수형

state, lifeCycle 관련 기능사용불가능(Hook 을 통해 해결됨)

메모리 자원을 함수형 컴포넌트보다 덜 사용함

컴포넌트 선언이 편하다.

**state 사용 차이**

state : 컴포넌트 내부에서 바뀔 수 있는 값

클래스형

1. constructor안에서 this.state 초기값 설정 가능
2. constuctor 없이 바로 state 초기값 설정 가능
3. 클래스 컴포넌트의 state는 객체 형식
4. this.setState 함수로 state의 값을 변경할 수 있다.

함수형

1. useState를 사용한다.
2. useState함수를 호출하면 배열이 반환되는데 첫번째는 현재 상태, 두번째는 상태함수

**props 사용차이**

props

컴포넌트의 속성을 설정할 때 사용하는 요소

읽기 전용

컴포넌트 자체 props를 수정해서는 안된다.

모든 React 컴포넌트는 자신의 props를 다룰 때 반드시 순수함수처럼 동작해야한다.

수정되는 것은 state

클래스

1. this.props를 통해 값을 불러올 수 있다.
2. 부모 객체의 키 값, 자식 props 활용

함수형

1. props를 불러올 필요없이 바로 호출할 수 있다.

**lifeCycle**

lifeCycle API : 컴포넌트가 DOM 위에 생성되기 전 후 및 데이터가 변경되어 상태를 업데이트 하기

전 후로 실행되는 메소드들이다.

**이벤트 핸들링**

클래스

함수 선언시 에로우 화살로 바로 선언 가능하다.

요소에서 적용하기 위해서는 this를 붙여야한다.

handleChange = e => {

this.setState({

message: e.target.value

});

}

…

onChange={this.handleChange}

함수형

const 키워드 + 함수 형태로 선언해야함

요소에서 적용하기 위해 this를 붙일 필요 없다.