2. JSX

2-2 JSX

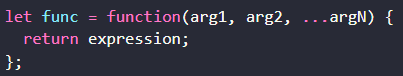
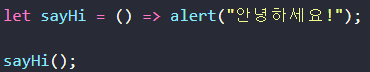
* 컴포넌트
  + 특징: 렌더링하면 함수에서 반환하고 있는 내용을 나타냄
* 문법
  + 컴포넌트에서 여러 요소는 부모 요소 하나로 감싸야 함
    - <div>로 감싸던가, <Fragment>로 감싸면 됨
  + 자바스크립트 표현식은 { } 로 감싸면 됨
    - Const name=’리액트’ … <h1>{name}</h1>
  + If 문을 쓰려면 {name==’lol’ ? (<h1>리액트</h1>) : (<h2>리액트 no </h2>)}
  + Undefined로 렌더링은 OR(||)연산자를 이용하기
  + 인라인 스타일링(background-color) 같은 경우는 카멜 표기법으로 backgroundColor로 바꾸기
  + class 대신 className으로
  + 태그는 꼭 닫기
  + 주석은 {/\*…..\*/}

3. 컴포넌트

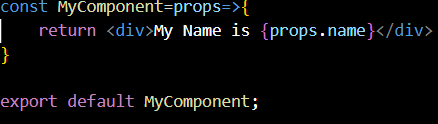
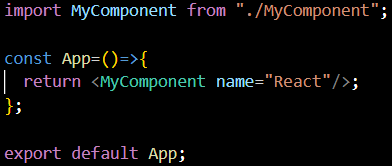
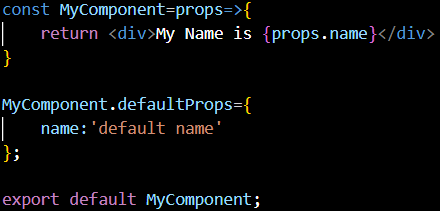
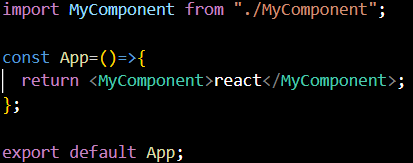
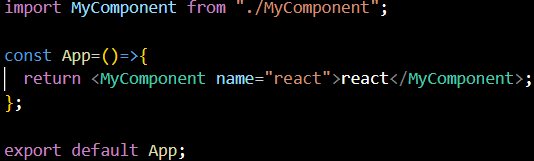
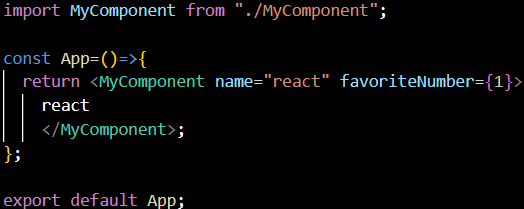
3-1 클래스형 컴포넌트

* 컴포넌트 선언 방식: 함수 컴포넌트, 클래스 컴포넌트
  + 차이점: 클래스형 컴포넌트의 경우 state기능 및 라이프사이클 기능을 사용가능, 임의 메서드 정의할 수 있음. render 함수가 필수.

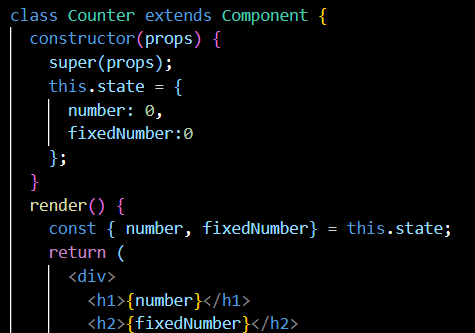
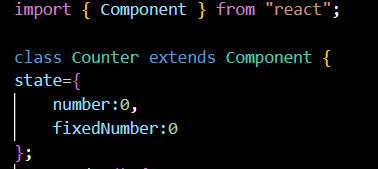
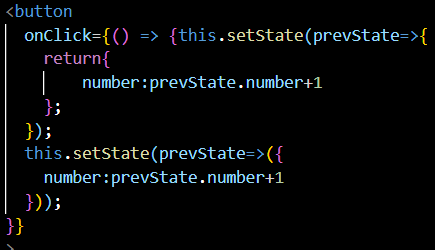
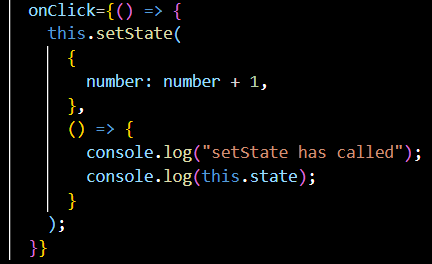
3-2 첫 컴포넌트 생성

* 화살표 함수
  + 목적: 단순하고 간결한 문법으로 함수를 만들기 위함
  + 실행
    - 기본적인 틀
      * 
        + 같은 뜻: 
    - 인수가 하나도 없을 때, 괄호를 비워 놓기. **단, 괄호생략불가!**
      * 
  + 일반 함수는 자신이 종속된 객체(괄호후 선언된)를 this로 가리키며, 화살표 함수는 자신이 종속된 인스턴스를 가리킴
  + 장점
    - 값을 연산해 바로 반환해야 할 때 사용해서 가독성 높임
  + 모듈 내보내기
    - export: export default MyComponent
  + 모듈 불러오기
    - import MyComponent from ‘./MyComponent’

3-3 props

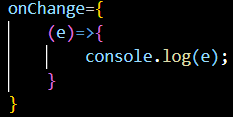
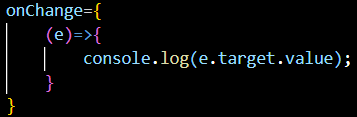
* 목적: 어떠한 값을 컴포넌트에게 전달할 때 사용(state는 컴포넌트에 데이터 저장)
* 실행
  + 컴포넌트에서 name이라는 props를 렌더링 하도록 수정
    - 
    - props 랜더링 할 때 JSX 내부에 {}기호로 감싸 주기
  + props 값 지정하기
    - 
  + props 기본값 설정
    - 
  + children: 컴포넌트 태그 사이의 내용(아래 예로는 ‘react’)을 보여주는 props
    - 
    - 
  + 비구조화 할당(destructuring assignment) 문법: props.name, props.children 과 같이 props라는 키워드를 반복적으로 붙이는 것을 막기 위함
    - 
    - 혹은, 
  + propTypes: props 타입을 지정할 때 사용
    - 
      * name값은 무조건 string형태로 전달해야 됨
    - 
  + isRequired: propTypes을 지정하지 않았을 경우 경고 메세지
    - 
    - 
  + 클래스형 컴포넌트에서 props
    - 
      * render 함수에서 this.props 조회
    - 
      * defaultProps와 propTypes 설정할 때 class 내부에서 지정하는 방법

3-4 state

* 목적: 컴포넌트 내부에서 값을 바뀌게 하기 위함(props는 컴포넌트간 데이터 전달)
* 실행
  + 클래스형 컴포넌트의 state
    - 
      * constructor: 컴포넌트에 state을 설정할 때 사용
        + constructor 선언시 super(props)를 반드시 호출해야 함
      * this.state: 초깃값 선언, 객체 형식이어야 함
      * render 함수에서 state를 조회할 때 this.state를 조회
      * onClick이라는 값을 props로 넣음, 버튼이 클릭될 때 호출시킬 함수 설정
      * this.setState: state 값을 바꿈
      * state 객체 안에 여러 값을 선언 할 수 있음
        + 
        + 혹은, 
    - 객체 대신 함수인자 전달하기(클릭할 때 +3 버튼 만들기)
      * 
        + this.setState로 시작하는 함수 두개는 똑같은 역할
    - this.setState가 끝난 후 특정 작업 실행하기
      * 
    - useState: 유저의 상호작용이나, 함수에 의해 변경될 수 있는 값들을 지정하고 변경하는 데 사용
      * 
        + getter는 message, setter는 setMessage
      * 한 컴포넌트에서 userState 여러 번 사용하기

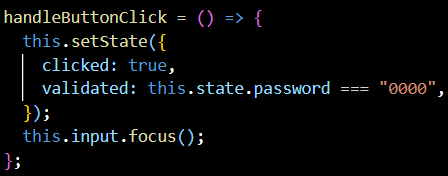
4. 이벤트 핸들링

4-1. 리액트의 이벤트 시스템

* 이벤트
  + 목적: 사용자가 웹 브라우저에서 DOM 요소들과 상호 작용하는 것을 위함
  + 특징: 버튼에 마우스 커서를 올렸을때 어떠한 이벤트를 실행, 클릭했을 때 onClick 이벤트 실행 등
  + 주의점
    - 이벤트 이름은 카멜 표기법
    - 이벤트를 설정할 때 함수 형태의 값을 전달해야 함
    - DOM요소에만 이벤트를 설정할 수 있음(컴포넌트에는 설정 불가)
  + 실행
    - onChange 이벤트: javascript를 통해 변화가 일어났는지 탐지
      * 
        + e는 SyntheticEvent(react에서 제공하는 고유 이벤트 객체)로 웹 브라우저의 네이티브 이벤트를 감싸는 객체
      * e.persist() 함수: 비동기적(한번에 여러개가 진행되는)으로 이벤트 객체를 참조할 때 사용
        + 
    - \*임의 메서드: 가독성을 높이기 위함
      * 
        + 임의 메서드(handleChange&handleClick)와 this의 관계가 끊어지는 것을 막기 위해 constructor에서 바인딩
    - Property Initializer Syntax 문법: 새 메서드를 만들 때마다 constructor를 수정하는 것을 간단하게 하기 위함
      * 
    - input 여러 개 다루기
      * 
        + 객체 안에서 key를 []로 감싸면 그 안에 넣은 레퍼런스가 가리키는 실제 값이 key 값으로 사용됨
    - onKeyPress 이벤트: 키를 눌렀을 때 발생하는 이벤트
      * 

5. ref:DOM에 이름 달기

5-1. ref

* ref
  + 목적: 리액트 프로젝트 내부에서 DOM에 이름을 달기 위해
  + 특징: DOM을 꼭 직접적으로 건들어야 할 때만 사용(특정 input에 포커스 주기나 스크롤 박스 조작하기 등)
  + 실행방법: 콜백 함수 사용, createRef 사용
    - 콜백 함수 사용
      * 
        + ref라는 콜백함수로 props를 전달
        + ref값을 parameter로 전달받고, 함수 내부에서는 parameter로 전달 받은 ref를 컴포넌트의 멤버변수로 선언
        + 이렇게 선언되면, 이 input 요소는 this.input을 통해 접근이 가능
      * 
        + onClick을 할떄 input에 포커스(커서가 계속 input에 있음)를 줌
    - createRef 사용
    - 컴포넌트에 ref 달기
      * 
        + const{scrollHeight, clientHeight}=this.box

const scrollHeight=this.box.scrollHeight;

const clientHeight=this.box.clientHeight;

* + - * 