11.3

**Hooks 추가 공부**

Hooks – info.js

useState – 내가 state로 쓸 변수를 만들어 놓고, state값으로 쓸 변수를 업데이트 시켜서 (setName을 사용해서) 이벤트 함수가 입력 된 값이 들어가지도록 하고, (name = e.target.value) 해당 name을 출력해야하는 곳에 넣어준다.

  const [name, setName] = useState('');

  const [nickname, setNickname] = useState('');

  const onChangeName = (e) => {

    setName(e.target.value);

  };

  const onChangeNickName = (e) => {

    setNickname(e.target.value);

  };

  /\* 1. 두개의 이벤트 함수를 만들어줌 \*/

  return (

    <div>

      <div>

        <input value={name} onChange={onChangeName} />

        <input value={nickname} onChange={onChangeNickName} />

        {/\* value를 onchange로 넣어둠, 변화를 받는 함수 \*/}

        {/\* 2. 이벤트 함수를 input onchange에 넣어둠 \*/}

      </div>

      <div>

        {/\* 이름 입력하면 이름이, 닉네임 입력하면 닉네임이 나와야한다. 어떻게 해야할까? \*/}

        <div>

          <b> 이름 : </b> {name} {/\* 이렇게 하면 바로 변수 지정이 된다. \*/}

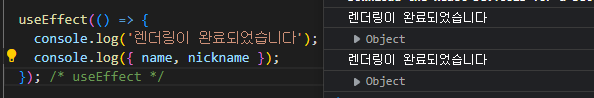
        </div>

        <div>

          <b> 닉네임 : </b> {nickname}

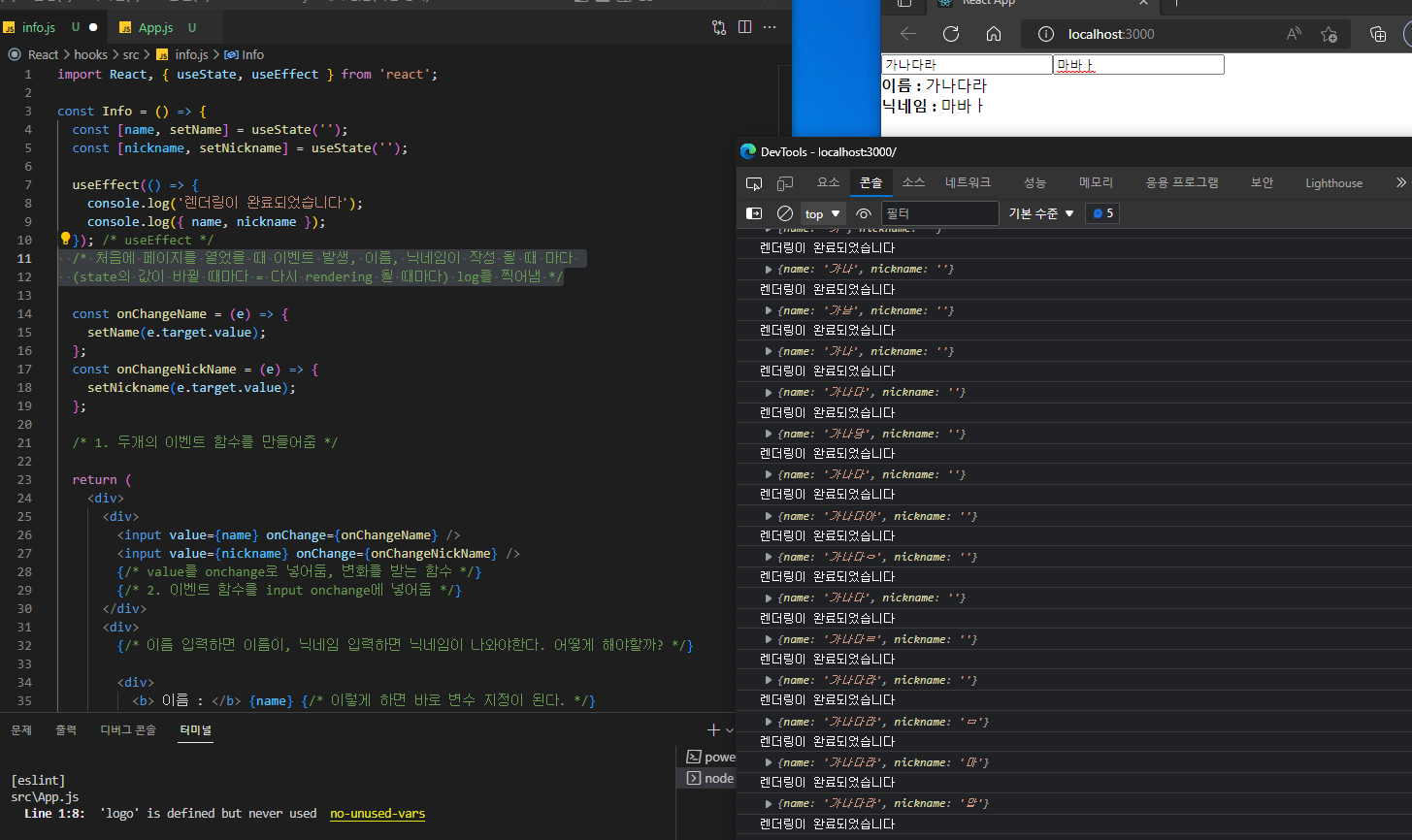
        </div>

userEffect – 컴포넌트가 작동될 때 마다 새로운 행동들이 생기게 함



  /\* 처음에 페이지를 열었을 때 이벤트 발생, 이름, 닉네임이 작성 될 때 마다

  (state의 값이 바뀔 때마다 = 다시 rendering 될 때마다) log를 찍어냄 \*/



  /\* useEffect 2. 업데이트 될 때만 effect가 뜨고, rendering 될 때는 안뜨게 함. \*/

  useEffect(() => {

    console.log('렌더링이 완료되었습니다');

    console.log({ name, nickname });

  }, []); /\* 내용이 빈 배열을 하나 넣어줌 \*/

  /\* useEffect 3. 특정 값이 변경 될 때만 뜨게 하는 방법 \*/

  useEffect(() => {

    console.log('렌더링이 완료되었습니다');

    console.log({ name, nickname });

  }, [name]); /\* 비어있던 배열 안에 인식할 특정 값을 넣어줌 \*/

Index.js의 react.strictmode가 두번 렌더링하게 되어, useEffect에서 렌더링이 두번씩 뜰 경우도 있다.

  // useEffect 4. useEffect안에서 돌리는 함수 (뒷정리 함수)

  useEffect(() => {

    console.log('effect');

    console.log({ name });

    return () => {

      console.log('cleanup');

      console.log(name);

    }; /\* useEffect를 사용할 때 함수를 사용하면 렌더링이 되기 전에 return이 먼저 실행된다!

    즉, 렌더링 되기전에 특정 작업을 주고 싶은 경우에 쓰면 좋다 \*/

  }, [name]);

useReducer

import React, { useReducer } from 'react';

function reducer(state, action) {

  switch (action.type) {

    case 'INCREMENT':

      return { value: state.value + 1 };

      break;

    /\* increment라는 값이 들어왔을 때 증가 된 값을 내보냄 \*/

    case 'DECREMENT':

      return { value: state.value - 1 };

      break;

    /\* decrement라는 값이 들어왔을 때 감소 된 값을 내보냄 \*/

    default:

      return state;

      break;

    /\* 아무것도 없으면 그냥 state를 내보냄 \*/

  }

}

const Counter = () => {

  const [state, dispatch] = useReducer(reducer, { value: 0 });

  /\* value라는 애는 알아서 state로 들어감 \*/

  return (

    <div>

      <p>

        현재 카운터 값은 <b>{state.value}</b>이실예정이신 예정 ㅎ

      </p>

      <button

        onClick={() => {

          dispatch({ type: 'INCREMENT' });

        }}

      >

        {' '}

        +1{' '}

      </button>

      <button

        onclick={() => {

          dispatch({ type: 'DECREMENT' });

        }}

      >

        {' '}

        -1{' '}

      </button>

      {/\* dispatch - 함수처럼 사용할 수 있는 애, 보내다 \*/}

    </div>

  );

};

export default Counter;

디스패치가 작동되면 쭉 값이 reducer로 넘어간다. Action.name에 값이 들어감.

이 중에 name에 대한 값을 가져옴

값을 받아왔으니 Action.value에 값이 들어감. (function reducer 내부) name에 대해 업데이트 된 값을 가져온다.

https://redux.js.org/tutorials/essentials/part-1-overview-concepts

useMemo

import React, { useMemo, useState, useCallback } from 'react';

const getAverage = (numbers) => {

  console.log('계산중 ...');

  if (numbers.length === 0) return 0;

  const sum = numbers.reduce((a, b) => a + b);

  /\* 계산해주는 함수 reduce \*/

  return sum / numbers.length;

};

const Average = () => {

  const [list, setList] = useState([]);

  const [number, setNumber] = useState('');

  /\* 숫자가 바뀔 때마다 일일이 함수가 렌더링 됨. 그걸 막기 위해 useCallBack이용 \*/

  const onChange = useCallback((e) => {

    setNumber(e.target.value);

  }, []);

  const onInsert = useCallback(

    (e) => {

      const nextList = list.concat(parseInt(number));

      setList(nextList);

      setNumber('');

    },

    [number, list]

  ); /\* number, list바뀌었을 때 함수 생성 \*/

  /\* 숫자가 바뀔 때마다 일일이 함수가 렌더링 됨. 그걸 막기 위해 useCallBack이용 \*/

  /\* useMemo 이용하여 일일이 로그 안뜨게 하기 \*/

  const avg = useMemo(() => {

    getAverage(list);

  }, [list]);

  /\* useMemo를 이용해서, 값이 변경될 때만 실행함.

  즉 값을 넣으려 일일이 숫자를 적을 때마다 로그가 찍히지는 않음 \*/

  return (

    <div>

      <input value={number} onChange={onChange}></input>

      <button onClick={onInsert}> 등록 </button>

      <ul>

        {list.map((value, index) => (

          <li key={index}> {value} </li>

        ))}

      </ul>

      <div>

        <b> 평균값 : </b> {avg}

      </div>

    </div>

  );

};

export default Average;

useRef – ref가 포함된 DOM을 직접적으로 컨트롤 할 수 있어짐.

import React, { useMemo, useState, useCallback,useRef } from 'react';

  const onInsert = useCallback(

    (e) => {

      const nextList = list.concat(parseInt(number));

      setList(nextList);

      setNumber('');

      inputEl.current.focus(); /\* 포커스 함수 지정 \*/

    },

    [number, list]

 return (

    <div>

      <input value={number} onChange={onChange} ref={inputEl}></input>

      <button onClick={onInsert}> 등록 </button>

Custom을 작성하여 나만의 hook를 만들 수 있다.

Component가 아닌 hook을 만들거기 때문에 제목을 소문자로 해준다

Useinputs - reducer에서 값이 바뀔 떄마다 state, onchange 가 바뀌게 되고, 이게 state로 들어가서

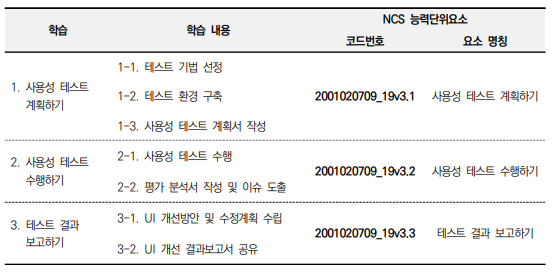
Info.js에 const{} = name에 실행되게 된다

**NCS 서술형 시험 문제**

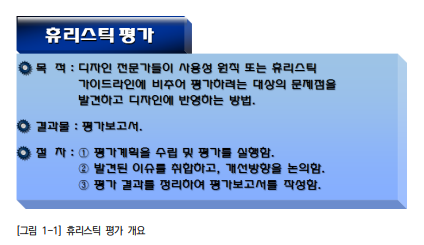
사용성 테스트 계획하기

사용성 테스트 수행하기

테스트 결과 보고하기



휴리스틱 평가. 시험에 나온다



17p 사용성 테스트 기법 선정하기 거기 읽어보기 (ncs\ui테스트\17페이지)

23p 테스트 환경 구축

시험에 나오는 중

**󰊳 프로토타입 (Prototype) 원래의 형태 또는 전형적인 예, 기초 또는 표준을 말하는 것으로 정보시스템의 미완성 버전 또는 중요한 기능들이 포함되어 있는 시스템의 초기모델을 의미한다. 프로토타입 은 사용자의 모든 요구사항이 정확하게 반영될 때까지 계속해서 개선되고 보완되는 것 이 일반적이다**

