

연습문제 07 폴이

문제 (1)

index.js

기본적인 초기화 작업만 수행함.

```
import React from 'react';
import ReactDOM from 'react-dom/client';
import App from './App';

const root = ReactDOM.createRoot(document.getElementById('root'));
root.render(
  <React.StrictMode>
    <App />
  </React.StrictMode>
);
```

App.js

```
import React from 'react';

const App = () => {
  // 결과를 출력할 <div>에 대한 참조변수
  const console = React.useRef();

  // 상태값
  const [rowNum, setRowNum] = React.useState(0);

  // 사용자의 입력값을 상태값에 적용하기 위한 이벤트 리스너
  const onRowNumChange = e => {
    setRowNum(e.currentTarget.value);
  };

  // rowNum 상태값이 변경된 경우 실행됨 --> 화면 내용 갱신
  React.useEffect(() => {
    let html = '';
    for (let i=0; i<rowNum; i++) {
      let str = '';

      for (let j=0; j<i+1; j++) {
        str += '*';
      }

      html += str + "<br/>";
    }
  });
}
```

```

        console.current.innerHTML = html;
    }, [rowNum]);

    return (
        <div>
            <h1>Exam07</h1>
            <p>useState, useEffect, useRef를 사용한 별찍기 구현</p>

            <hr />

            <div>
                <label htmlFor='rowNum'>rownum: </label>
                <input id='rowNum' type='text' value={rowNum} onChange=
{onRowNumChange} />
            </div>

            <hr />

            <div style={{
                fontSize: '16px'
            }} ref={console}></div>
        </div>
    );
};

export default App;

```

문제 (2)

App.js

```

import React from 'react';

/**
 * 사칙연산을 수행하기 위한 reducer함수
 * @param {int} state - 현재 상태값
 * @param {object} action - 액션 {x: 첫번째숫자, y: 두번째숫자, exec: 연산자 }
 * @returns 새로운 상태값
 */
function getResultValue(state, action) {
    let resultValue = 0;

    switch (action.exec) {
        case '+':
            resultValue = action.x + action.y;
            break;
        case '-':
            resultValue = action.x - action.y;
            break;
        case '*':
            resultValue = action.x * action.y;

```

```

        break;
      case '/':
        resultValue = action.x / action.y;
        break;
      default:
        resultValue = 0;
    }

    return resultValue;
  }

const App = () => {
  // 첫 번째 숫자를 위한 input 태그의 참조변수
  const x = React.useRef();
  // 두 번째 숫자를 위한 input 태그의 참조변수
  const y = React.useRef();
  // 연산자를 위한 select 태그의 참조변수
  const exec = React.useRef();

  // 상태값 갱신을 위한 reducer
  const [resultValue, setResultValue] = React.useReducer(getResultValue,
0);

  // 버튼 클릭 이벤트 --> useCallback을 활용하여 중복 처리 방지함.
  const onClick = React.useCallback(e => {
    // reducer에 전달할 action값을 JSON 형식으로 전달
    setResultValue({
      x: Number(x.current.value),
      y: Number(y.current.value),
      exec: exec.current[exec.current.selectedIndex].value
    });
  }, []);

  // resultValue 상태값이 변경된 후 그에 맞춰 변경되는 색상값 상태변수
  const color = React.useMemo(() => {
    return resultValue % 2 === 0 ? '#f60' : '#06f';
  }, [resultValue]);

  return (
    <div>
      <h1>Exam07</h1>
      <p>useReducer, useMemo, useCallback을 활용한 사칙연산</p>

      <hr />

      { /* 입력영역 구성 */ }
      <div>
        <input ref={x} type='text' />
        <select ref={exec}>
          <option value="+">+</option>
          <option value="-">-</option>
          <option value="*">*</option>
          <option value="/">/</option>
        </select>

```

```

        <input ref={y} type='text' />
        <button type='button' onClick={onButtonClick}>결과확인
    </button>
  </div>

  <hr />

  { /* 연산결과 상태값과 색상 상태값을 표시함 */ }
  <h1 style={{
    fontSize: '20px',
    color: color
  }}>결과값: {resultValue}</h1>
</div>
);
};

export default App;

```