React

hooks

- React 16.8에 새로 도입된 기능으로 기존 함수 컴포넌트에서 할 수 없었던 작업 수행 가능
- useState
  - 가장 기본적인 hook이며, 함수 컴포넌트에서 가변적 상태 유지 가능
  - Const [name, setName] = useState(‘ ‘);
  - 반환되는 배열에서 첫번째 원소는 상태 값, 두번째 원소는 상태를 설정하는 함수
  - 이 함수에 파라미터를 넣어 호출하면 전달 받은 파라미터로 값이 바뀌고 컴포넌트가 **리렌더링** 됨
  - useState 여러 번 사용 가능

hooks

- useEffect
  - 리액트 컴포넌트가 렌더링 될 때마다 특정 작업을 수행할 수 있도록 함
  - 클래스형 컴포넌트의 componentDidMount() 와 componentDidUpdate()를 합친 형태
  - useEffect에서 설정한 함수를 컴포넌트가 맨처음 화면에 렌더링 될때만 실행하고 업데이트될 때에는 실행하지 않으려면 함수의 두번째 파라미터로 비어 있는 배열([ ])을 넣어준다
  - 특정 값이 업데이트 될 때만 실행하고 싶을 때에는 두번째 파라미터로 전달되는 배열안에 검사하고 싶은 값을 넣어준다
  - 기본적으로 렌더링되고 난 직후에 실행되며 두번째 파라미터(배열)에 무엇을 넣는지에 따라 실행 조건이 달라짐
  - 컴포넌트가 언마운트 되기 전이나 업데이트 되기 직전에 어떤 작업을 수행하고자 하는 경우 useEffect에서 뒷정리(cleanup) 함수를 리턴해야 한다

hooks

- useReducer
  - useState보다 더 다양한 컴포넌트 상황에 따라 다양한 상태를 다른 값으로 업데이트
  - 현재 상태와 업데이트를 위해 필요한 정보를 담은 액션(action) 값을 전달 받아 새로운 상태를 반환
  - useReducer에서 사용하는 액션 객체는 반드시 액션 type를 가지고 있을 필요 없음

hooks

- useMemo
  - 함수 컴포넌트에서 발생하는 연산의 최적화
  - 렌더링 과정에서 특정 값이 바뀌었을 때만 연산을 실행하고 원하는 값이 바뀌지 않았다면 이전에 연산했던 결과를 다시 사용
- useCallback
  - useMemo와 비슷한 기능을 하며 렌더링 기능을 최적화 상황에 적용
  - 첫번째 파라미터에 생성하고자 하는 함수
  - 두번째 파라미터인 배열에는 어떤 값이 바뀌었을 때 함수를 생성해야 하는지 명시
    - 배열이 비었으면 컴포넌트가 렌더링 될 때 만들었던 함수를 계속 사용
    - 배열 안에 있는 값이 바뀌거나 추가될 때 새로 만들어진 함수 사용
- useRef
  - 함수 컴포넌트에서 ref를 쉽게 적용할 수 있게 함
  - 로컬 변수(렌더링과 상관없이 바뀔 수 있는 값)를 사용해야 할 경우

hooks

- 커스텀 hooks
  - 여러 컴포넌트에서 비슷한 기능을 공유할 경우, 이를 custom hook으로 만들어 로직 재사용
- Hooks library
  - https://nikgraph.github.io/react-hooks
  - https://github.com/rehooks/awesome-react-hooks

class 사용

- 기존 자바스크립트(ES5)에서는 객체 생성 위해 prototype 사용
- Class(ES6)는 prototype과 같은 개념

함수형 컴포넌트

- 클래스 형 컴포넌트와 달리 state가 없고 생명주기 함수 사용할 수 없음
- 상위 컴포넌트에서 props와 context를 파라미터로 전달받아 사용
- Render() 함수가 없고 return 만 사용하여 화면을 그림

화살표 함수

- Function 대신 '=>' 문자열 사용
- Return 문자열 생략 가능
- 콜백 함수에서 this를 bind해야 하는 문제도 발생하지 않음

Fragments 사용

- 컴포넌트 단위로 html 요소를 리턴할 때 반드시 <html> 태그로 전체를 감싸주어야 함
- <Fragments> 태그를 사용하면 불필요한 html 태그를 추가하지 않아도 됨
  - <React.Fragment> </React.Fragment>
  - <> </>

컴포넌트 styling 방식

- 일반 CSS
- Sass
  - 자주 사용되는 CSS 전처리기(pre-processor) 중 하나이며 확장된 CSS 문법 사용
- CSS 모듈
  - 스타일 작성시 CSS 클래스가 다른 CSS 클래스의 이름과 충돌하지 않도록 파일마다 고유한 이름을 자동으로 생성하는 옵션
- Styled-components
  - 스타일을 자바스크립트 파일에 내장시키는 방식
  - 스타일이 적용된 컴포넌트를 만드는 방식
  - 반응형 디자인 적용