03. state

> - 리엑트에서 state는 컴펴넌트 내부의 값을 변경할 수 있는 값을 의미
> - props는 컴퍼넌트가 사용하는 과정에서 부모 컴포넌트가 설정하는 값이며
> - 컴퍼넌트 자신은 해당 props(부모컴퍼넌트의 설정값)를 읽기 전용으로만 사용 할 수 있다.
> - 따라서, props를 변경하려면 부모 컴퍼넌트에서 변경해야 한다.
> - 리액트에는 2가지 종류의 state가 있다.
>
>> 1. 클래스형 컴퍼넌트가 지니고 있는 state
>> 2. 함수형 컴퍼넌트에서 useState라는 함수에서 사용하는 state

3.1 클래스형 컴퍼넌트의 state

1. src/mysrc 폴더생성
2. State01_Counter.js 파일생성
   - 경고메시지 : Imported JSX component State01_Counter must be in PascalCase or SCREAMING_SNAKE_CASE
   - State01_Counter를 State01Counter로 변경

```
src/App.js
import "./App.css";
import State01_Counter from "./mysrc/State01_Counter";

function App() {

  return (
    <State01_Counter />
  );

}
export default App;

mysrc/State01_Counter.js
import React, { Component } from 'react';

class State01_Counter extends Component {

  constructor(props) {
    super(props);
    // state초기값 설정하기
    this.state = {
      number: 0
    }
  }
```

```
  render() {
    const { number } = this.state;
    return (
      <div>
        <h1>{number}</h1>
        <button onClick={() => {
          this.setState({ number: number + 1})
        }}>+1</button>
      </div>
    );
  }
}

export default State01_Counter;
```

- 클래스 컴퍼넌트에서 state를 설정할 때는 `생성자메서드를 작성하여 설정` 한다.
- constructor를 작성할 때 맨위에 `super(props)를 호출` 해야 한다.
- 생성자함수가 호출되면 현재 클ㄹ스형 컴퍼넌트가 상속하고 있은 `React.Componet 클래스가 지닌 생성자함수를 호출`
- `this.state에 초기값을 설정` , `컴퍼넌트이 state는 객체이어야 한다`
- render함수에서 현재 state를 조회할 때는 this.state를 조회한다.
- button.onClick에 화살표함수를 설정 이것을 `이벤트를 설정` 한다라고 하다

3.1.1 state객체안에 복수의 값

```
mysrc/State01_Counter.js
import React, { Component } from 'react';

class State01_Counter extends Component {

  constructor(props) {
    super(props);
    // state초기값 설정하기
    this.state = {
      number: 0,
      otherNumber: 0
    }
  }

  render() {
    const { number, otherNumber } = this.state;
    return (
      <div>
        <h1>{number}</h1>
        <h2>변경되지 않는 값 : {otherNumber}</h2>
        <button onClick={() => {
          this.setState({ number: number + 1})
        }}>+1</button>
      </div>
    );
  }
}
```

3.1.2 생성자를 선언하지 않고 state 초기값 설정하기

```
mysrc/State01_Counter.js
import React, { Component } from 'react';

class State01_Counter extends Component {

  state = {
    number: 0,
    otherNumber: 0
  }

  render() {
    const { number, otherNumber } = this.state;
    return (
      <div>
        <h1>{number}</h1>
        <h2>변경되지 않는 값 : {otherNumber}</h2>
        <button onClick={() => {
          this.setState({ number: number + 1})
        }}>+1</button>
      </div>
    );
  }
}
```

3.1.3 this.setState에 객체대신 함수인자 전달하기

- this.setState함수를 사용하여 state값을 변경할 때는 `상태가 비동기적으로 업데이트` 된다.
- onClick에 설정한 함수내부에서 this.setState를 두번 호출할 경우는?
  - 2번 호출한다고 해서 2씩 더해지는 것이 아니라 1씩만 더해진다.
  - this.setState를 2번 사용한다고 해서 state값이 변경되지 않기 때문이다.

```
mysrc/State01_Counter.js
import React, { Component } from 'react';

class State01_Counter extends Component {
  state = {
    number: 0,
    otherNumber: 0
  }

  render() {
    const { number, otherNumber } = this.state;
    return (
      <div>
        <h1>Number = {number}</h1>
        <h2>변경되지 않는 값 : {otherNumber}</h2>
        <button onClick={() => {
          this.setState({ number: number + 1 })
          this.setState({ number: this.state.number + 1 })
        }}>setState두번호출</button>
      </div>
    );
  }
```

- 이에 대한 해결책으로는 this.setState를 사용할 때 `객체 대신 함수를 인자로 넣어주는 것`
  - `this.setState(prevState => ({ ... }));`

```
mysrc/State01_Counter.js
import React, { Component } from 'react';

class State01_Counter extends Component {
  state = {
    number: 0,
    otherNumber: 0
  }

  render() {
    const { number, otherNumber } = this.state;
    return (
      <div>
        <h1>Number = {number}</h1>
        <h2>변경되지 않는 값 : {otherNumber}</h2>
        <button onClick={() => {
              this.setState(prevState => {
                return {
                  number: this.state.number + 1
                };
              });
              //함수에서 객체를 반환
              this.setState(
                 prevState => ( { number: prevState.number + 1 } )
              );
            }
        }>setState두번호출</button>
      </div>
    );
  }
}
```

3.1.4 this.setState 작업후 특정 작업 실행하기

- `this.setState({...}, () => { ... } );`

```
mysrc/State01_Counter.js
import React, { Component } from 'react';

class State01_Counter extends Component {
  state = {
    number: 0,
    otherNumber: 0
  }

  render() {
    const { number, otherNumber } = this.state;
    return (
      <div>
        <h1>Number = {number}</h1>
        <h2>변경되지 않는 값 : {otherNumber}</h2>
```

```
          <button onClick={() => {
              this.setState(
                {number: number + 1},
                () => {
                  console.log('setState가 호출!!!');
                  console.log(this.state);
                }
              );
            }
          }>callback함수호출하기</button>
        </div>
      );
    }
  }
```

3.2 함수 컴퍼넌트에서 useState

- 리액트 16.8이전버전에서는 함수 컴퍼넌트에서 state를 사용할 수 없었다.
- 그 이후 버전에서는 state를 사용할 수 있게 되었는데 사용법이 이전과는 다르다.
- 이 과정에서 `Hooks라는 것을 사용`하게 되었다.

3.2.1 배열 비구조화 할당

- 배열 비구조화 할당은 객체 비구조화 할당과 유사
- 즉, 배열안의 요소를 쉽게 추출할 수 있다.

```
const arr = [1, 2];
const [one, two] = arr; // 배열 비구조화 할당
```

3.2.2 useState사용하기

```
src/App.js
import "./App.css";
import State01_Counter from "./mysrc/State01_Counter";
import State02Say from "./mysrc/State02Say";

function App() {

  return (
    <>
      <State01_Counter />
      <hr/>
      <State02Say />
    </>
  );

}

export default App;

mysrc/State02Say.js
import { useState } from 'react';

const State02Say = () => {
```

```
  const [message, setMessage] = useState('');
  const onClickEnter = () => setMessage('안녕하세요!');
  const onClickLeave = () => setMessage('안녕히 가세요!');

  return (
    <div>
      <button onClick={onClickEnter}>입장</button>
      <button onClick={onClickLeave}>퇴장</button>
      <h1>{message}</h1>
    </div>
  );
};

export default State02Say;
```

- usetState함수의 인자에 초기값 설정
- 클래형 컴퍼넌트의 초기값은 객체를 넣어주어야 하지만, usetState에서는 반드시 객체가 아니어도 상관없다. 즉, `값의 형태는 자유`
- 함수를 호출하면 `배열이 반환`, 배열의 `첫 번째 요소는 현재상태, 두 번째 요소는 상태를 변경하는 함수`
- 이 함수를 `setter함수`라고 한다., 그리고 배열 비구조화 할당을 통해 이름을 자유롭게 정의할 수 있다.

3.2.3 한 컴퍼넌트에서 useState 여러번 사용하기

```
mysrc/State02Say.js
import { useState } from 'react';

const State02Say = () => {
  const [message, setMessage] = useState('');
  const [color, setColor] = useState('black');

  const onClickEnter = () => setMessage('안녕하세요!');
  const onClickLeave = () => setMessage('안녕히 가세요!');

  return (
    <div>
      <button onClick={onClickEnter}>입장</button>
      <button onClick={onClickLeave}>퇴장</button>

      <h1 style={{ color }}>{message}</h1>
      <button style={{ color: 'red' }} onClick={() =>
setColor('red')}>빨간색</button>
      <button style={{ color: 'green' }} onClick={() =>
setColor('green')}>초록색</button>
      <button style={{ color: 'blue' }} onClick={() =>
setColor('blue')}>파란색</button>
    </div>
  );
};

export default State02Say;
```

3.3 state를 사용할 때 주의사항

- state값을 변경할 때 setState or useState를 통해 전달 받은 `세터함수를 사용` 해야 한다.
- 그렇다면, 배열이나 객체를 업데이트할 때는
  - `배열이나 객체 사본을 만든 후에 그 사본에 값을 업데이트한 후`
  - `그 사본의 상태를 setState or setter함수를 통해 업데이트`
- 객체에 대한 사본을 만들 때는 `spread 연산자 ...object` 를 사용하여 처리하고 `배열은 배열의 내장함수를 활용` 한다

- 개발자도구 console에서 실행해 보기 ```js // 객체 const object = {name:'홍길동', age:900} const copyObject = { ...object, addr:'서울' } // 사본을 만들고 addr만 덮어(추가)쓰기

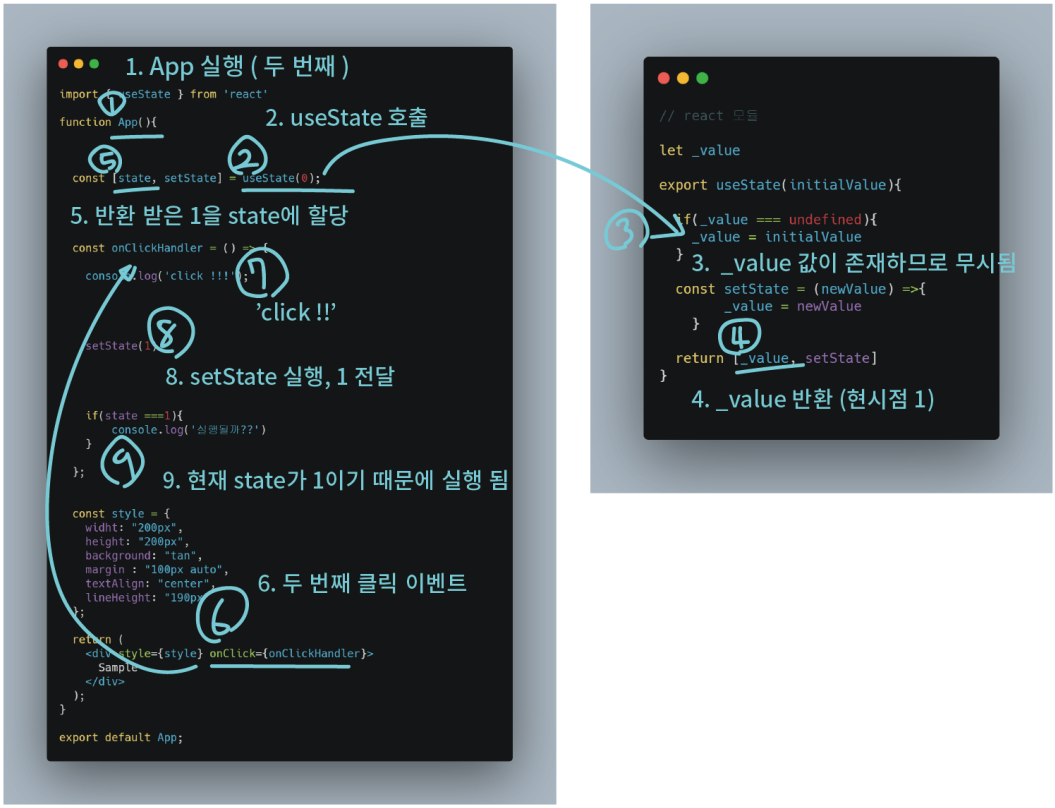
// 배열 const array = [{id: 1, name: '홍길동' }, {id: 2, name: '손흥민' }, {id: 3, name: '이강인' }];

let copyArray = array.concat({id:4, name: '김민재'}); // 추가 copyArray.filter(item => item.id !== 2); // 삭제 copyArray.map(item => (item.id === 1 ? {...item, name:'소향'} : item)); // 수정, 실행만 함, 결과를 저장하려면 변수에 대입해야 함

```

정리

> - props와 state는 둘 다 컴퍼넌트에서 사용하거나 랜더링할 데이터를 담고 있는 점에서 비슷하지만 그 역할은 매우 다르다.
> - `props는 부모 컴퍼넌트가 설정` 하고 `state는 컴퍼넌트 자체적으로 컴퍼넌트 내부에서 값을 수정` 할 수 있다.
> - props를 사용한다고 해서 값이 무조건 고정적이지는 않다.
> - 부모 컴퍼넌트의 state를 자식 컴퍼넌트의 props로 전달하고,
> - 자식 컴퍼넌트에서 특정 이벤트가 발생할 때 부모 컴퍼넌트의 메서드를 호출하면 props도 유동적으로 변경할 수 있다.
> - 컴퍼넌트를 만들 때는 useState를 사용할 것을 권장
> - 리액트 개발팀이 `함수컴퍼넌트와 Hooks(userState..)를 사용하는 것이 컴퍼넌트개발방식` 이라 공지!!

# 3.4 useState 동작 process

- React-useState는 어떻게 동작할까
  - 참고: https://velog.io/@jjunyjjuny/React-useState%EB%8A%94-%EC%96%B4%EB%96%BB%EA%B2%8C-%EB%8F%99%EC%9E%91%ED%95%A0%EA%B9%8C

## 1. 최초 렌더링 - 첫 클릭 이벤트 발생
```

2. setState 실행

3. setState로 인한 리렌더링 - 두 번째 클릭 이벤트 발생


1. App 실행 (두 번째)
2. useState 호출
5. 반환 받은 1을 state에 할당
'click !!'
8. setState 실행, 1 전달
9. 현재 state가 1이기 때문에 실행 됨
6. 두 번째 클릭 이벤트
// react 모듈
let _value
3. _value 값이 존재하므로 무시됨
4. _value 반환 (현시점 1)