

React Session 03

Props, useState

까먹은 거 다 알아 복습해요
나도 까먹음



VS Code 켜 죠 잉

Component

JSX

yarn create react-app 남

렌더링/리렌더링

재사용

map()

Styled
Component

package.json

npm install

yarn? npm?

.gitignore

Virtual DOM

CSR?

01

컴포넌트 분리

일어면 안되;;

App.js

```
<div>
  <div>
    <span>
      <p>aaa</p>
      <p>aaa</p>
      <p>aaa</p>
    </span>
    <span>
      <p>BBB</p>
      <p>aCCC</p>
      <p>SDSa</p>
    </span>
    <span>
      <p>ESa</p>
      <p>aEa</p>
    </span>
  </div>
  <div>
    <span>
      <p>Ea</p>
      <p>aSEa</p>
    </span>
    <span>
      <p>aaa</p>
      <p>aaa</p>
      <p>aaa</p>
    </span>
    <span>
      <p>aaa</p>
      <p>aaa</p>
      <p>aaa</p>
    </span>
  </div>
</div>
```

재사용성 X
복잡도 큼
유지보수 X

Button

Button

page 1

Button

같은 컴포넌트를
여기저기서 쓰고
싶을 때

page 2

Button

즉, Button을
재사용하고 싶을 때

02 props

props 란?

- 프로퍼티, Properties의 줄임말
- 상위 컴포넌트가 하위 컴포넌트에 값을 전달할 때 사용 (단방향 흐름)
- Read only, 자식은 props를 마음대로 수정할 수 없다.
- Props로 문자열을 전달할 때에는 “”(따옴표), 문자열 외의 값 전달 시에는 {}(중괄호)를 사용

props (example)

상황1 : 페이지마다 버튼을 각각 만드는 게 귀찮아서
버튼을 하나 만들고 재사용하고 싶어요.

```
const Button = () => {  
  return(  
    <div>  
      <span>button</span>  
    </div>  
  )  
}
```

Button.js

자식 컴포넌트

```
import React from "react";  
import Button from "../Button";  
  
const TestPage = () => {  
  return (  
    <div>  
      <Button />  
    </div>  
  );  
};  
  
export default TestPage;
```

TestPage.js

부모 컴포넌트

props (example)

상황 1-2 : Button에 써진 글자를 바꾸고 싶어요



```
import React from "react";
import Button from "../Button";

const TestPage = () => {
  return (
    <div>
      <Button>ddd</Button>
    </div>
  );
};

export default TestPage;
```

TestPage.js

이렇게 하면
안되나요?

이럴 때 props를 쓰는 겁니다 !

props (example)

상황 1-2 : Button에 쓰인 글자를 바꾸고 싶어요

```
import React from "react";

const Button = (props) => {
  return (
    <div>
      <span>{props.btnName}</span>
    </div>
  );
};

export default Button;
```

자식 컴포넌트는 props를
받아 **그대로** 사용합니다.

자식 맘대로 수정하면 안돼요?

○○ 버르장머리 없게

props (example)

상황 1-2 : Button에 쓰인 글자를 바꾸고 싶어요

```
import React from "react";
import Button from "../Button";

const TestPage = () => {
  return (
    <div>
      <Button btnName="버튼" />
    </div>
  );
};

export default TestPage;
```

props에 대한 값은 부모가
정해서 내려줍니다.

시키는 대로 하세요
너는 이걸 보여주기만 하세요

props (example)

상황 2 : 버튼 색상을 바꾸고 싶어요

```
✓ import React from "react";
  import styled from "styled-components";

✓ const Btn = styled.div`
  display: flex;
  justify-content: center;
  align-items: center;
  background-color: ${(props) => props.color};
  width: 100px;
  height: 80px;
`;

✓ const Button = (props) => {
  ✓ return (
  ✓   <Btn color={props.colorName}>
    <span>{props.btnName}</span>
  ✓   </Btn>
  );
};

export default Button;
```

Button 기능 추가

직접 해볼거요
Vs Code 줍줍

버튼을 클릭하면 숫자가 1씩 커지는 기능을 더하고 싶어요.

시도 1 : 부모 컴포넌트에서 counter 변수와 counterUp 메소드를 자식 컴포넌트에 props로 넘겨준다.

시도 2 : 부모 컴포넌트에서 counter 변수를 자식 컴포넌트에 props로 넘겨주고, 자식 컴포넌트에서 counterUp을 선언해 up해준다.

시도 3 : 자식 컴포넌트에서 다 해결해버린다. (이 경우 props와는 관련X)

결론 : 자식은 props를 변경할 수 없다. 변경할 수 있는 메소드를 넘겨주어 변경한다고 해도, 바람직하지 않다.

일반 변수는 실제로 값이 바뀌어도 Virtual DOM 이 이를 감지하지 못한다. 따라서 ○○○○ 하지 않기 때문에 화면에는 그려지지 않는다.

리렌더링

Props 하다가 왜 갑자기 리렌더링 ?

우리는 사용자와의 상호작용을 통해
일어나는 데이터의 변화를
화면에 렌더링 해야 한다

리렌더링의 조건

- 부모 컴포넌트의 렌더링
- State의 변경
- Props의 변경

03

useState

useState

이 모든 게 useState를 위한 빌드업이었다.

React Hooks가 제공하는
상태 관리를 위한 함수

React Hooks

또 생소한 용어,, 걱정 마 ~~

useState

함수형 컴포넌트

React v16.8

컴포넌트 상태 관리

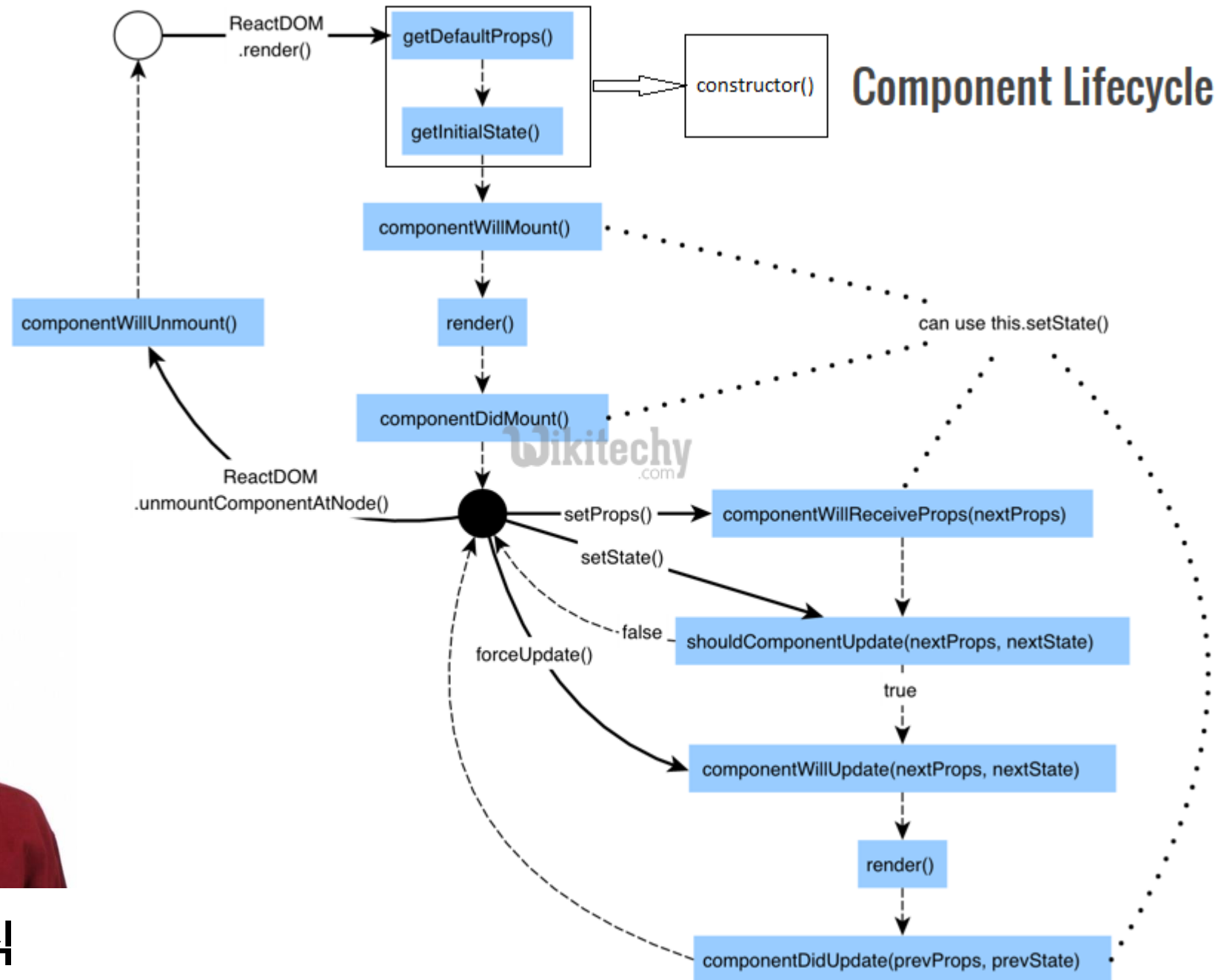
컴포넌트 생애 주기

useContext

useEffect

클래스형 컴포넌트 대체

Class형 컴포넌트



복잡한 로직

useState

```
const [count, setCount] = useState(0);
```

state

state를 변경 ?

Initial Value

구조 분해 할당

[a, b] = [10, 20];

useState

```
import React, { useState } from "react";

const StatePractice = () => {
  const [value, setValue] = useState(0);
  const handleClick = () => {
    setValue(1);
  };
  return <div onClick={handleClick}>{value}</div>;
};

export default StatePractice;
```

**State의 값이 변경될 때마다 (setState를 통해)
화면이 리렌더링 될거야**

useState

```
import React, { useState } from "react";

const StatePractice = () => {
  const [value, setValue] = useState(0);
  const handleClick = () => {
    setValue(value + 1);
    setValue(value + 1);
    setValue(value + 1);
  };
  return <div onClick={handleClick}>{value}</div>;
};

export default StatePractice;
```

3씩 더해질까 ?

```
import React, { useState } from "react";

const StatePractice = () => {
  const [value, setValue] = useState(0);
  const handleClick = () => {
    setValue(value + 1);
    console.log(value);
  };
  return <div onClick={handleClick}>{value}</div>;
};

export default StatePractice;
```

콘솔엔 제대로 찍힐까 ?

useState를 뜯어보자

제대로 이해해보자

useState 작동 원리 _ 단순 버전

```
// react 모듈

let _value

export useState(initialValue){

  if(_value === undefined){
    _value = initialValue
  }

  const setState = (newValue) =>{
    _value = newValue
  }

  return [_value, setState]
}
```

setState는
리렌더링을
발생시킨다.

setState는 state를 변경 시키는 게 아니었다

useState 작동 원리 _ 단순 버전

```
import React, { useState } from "react";

const StatePractice = () => {
  const [value, setValue] = useState(0);
  const handleClick = () => {
    setValue(value + 1);
    console.log(value);
  };
  return <div onClick={handleClick}>{value}</div>;
};

export default StatePractice;
```

1. `_value`에
0 담겨서
return

3. `console.log(0)`

5. 화면이 리렌더링 됨으로써
1이 제대로 그려짐

2. 버튼 클릭,
`handleClick`
함수 실행,
`setValue()`
1을 담아 호출

4. `_value`에
1 담겨서 return,
`setValue`에 의해
화면 리렌더링
(비동기)

```
// react 모듈

let _value

export useState(initialValue){

  if(_value === undefined){
    _value = initialValue
  }

  const setState = (newValue) =>{
    _value = newValue
  }

  return [_value, setState]
}
```


setState는 왜 비동기 ?

Batch Update

React 는 batch update를 16ms 단위로 진행한다.

16ms 동안 변경된 상태 값들을 모아서 단 한번의 리렌더링을 진행한다.

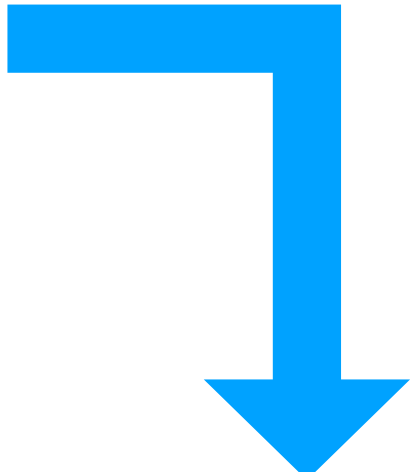
이러한 행동은 웹 페이지 렌더링 횟수를 줄여 좀 더 빠른 속도로 동작하게끔 만든다.

```
import React, { useState } from "react";

const StatePractice = () => {
  const [value, setValue] = useState(0);
  const handleClick = () => {
    setValue(value + 1);
    setValue(value + 1);
    setValue(value + 1);
  };
  return <div onClick={handleClick}>{value}</div>;
};

export default StatePractice;
```

setState의 인자로
함수를 전달해 해결



```
import React, { useState } from "react";

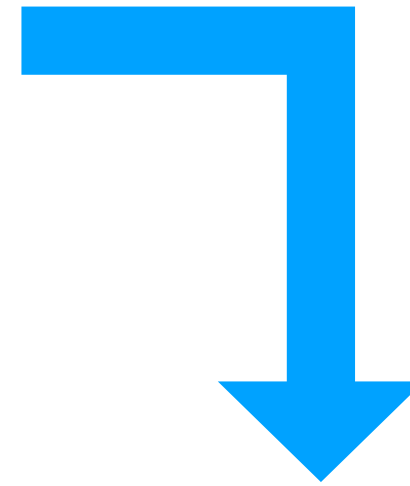
const StatePractice = () => {
  const [value, setValue] = useState(0);
  const handleClick = () => {
    setValue((value) => value + 1);
    setValue((value) => value + 1);
    setValue((value) => value + 1);
  };
  return <div onClick={handleClick}>{value}</div>;
};

export default StatePractice;
```

```
import React, { useState } from "react";

const StatePractice = () => {
  const [value, setValue] = useState(0);
  const handleClick = () => {
    setValue(value + 1);
    console.log(value);
  };
  return <div onClick={handleClick}>{value}</div>;
};

export default StatePractice;
```



useEffect 훅을 통해 해결

```
import React, { useState, useEffect } from "react";

const StatePractice = () => {
  const [value, setValue] = useState(0);

  useEffect(() => {
    console.log(value);
  }, [value]);

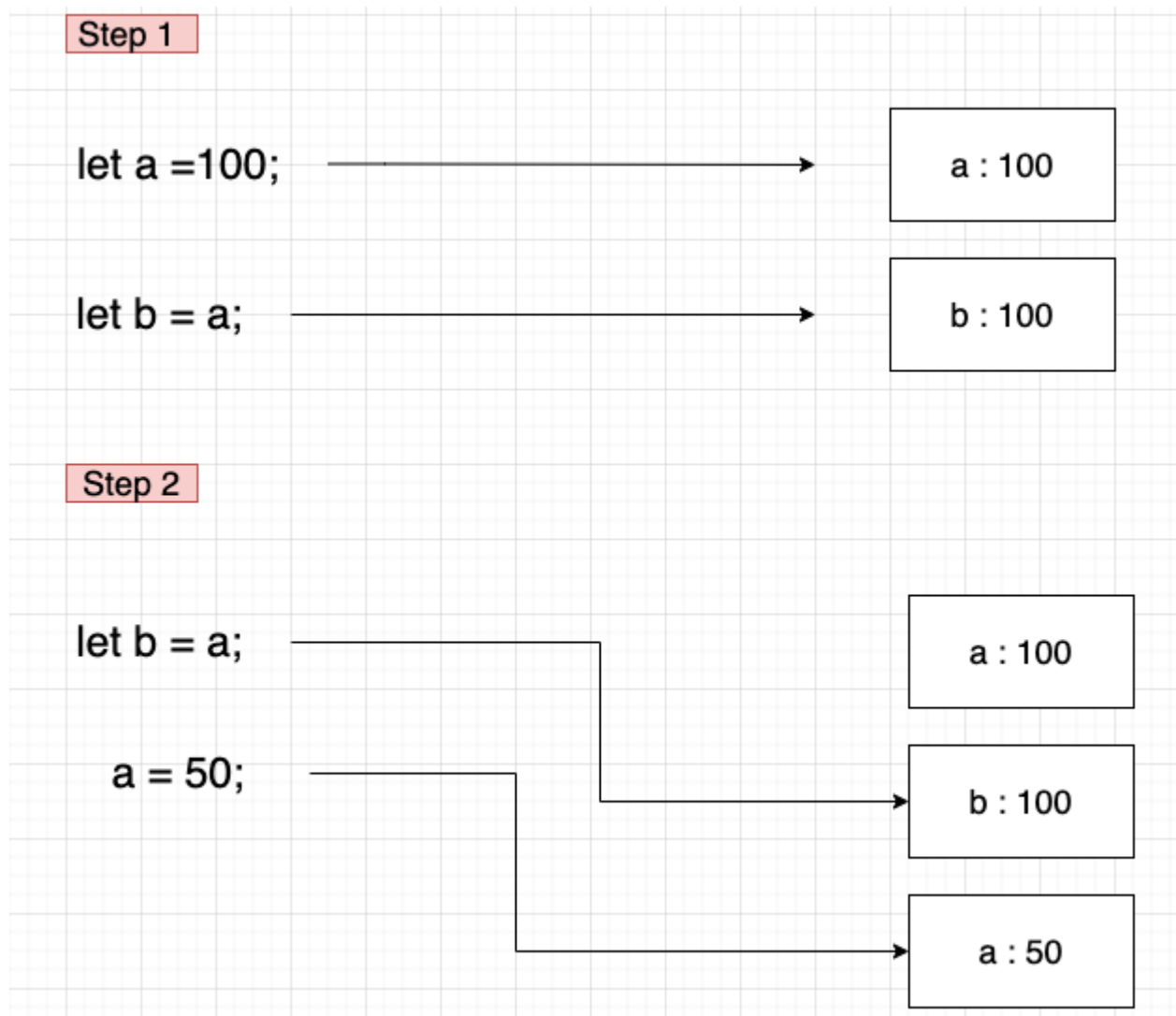
  const handleClick = () => {
    setValue(value + 1);
  };
  return <div onClick={handleClick}>{value}</div>;
};

export default StatePractice;
```

04

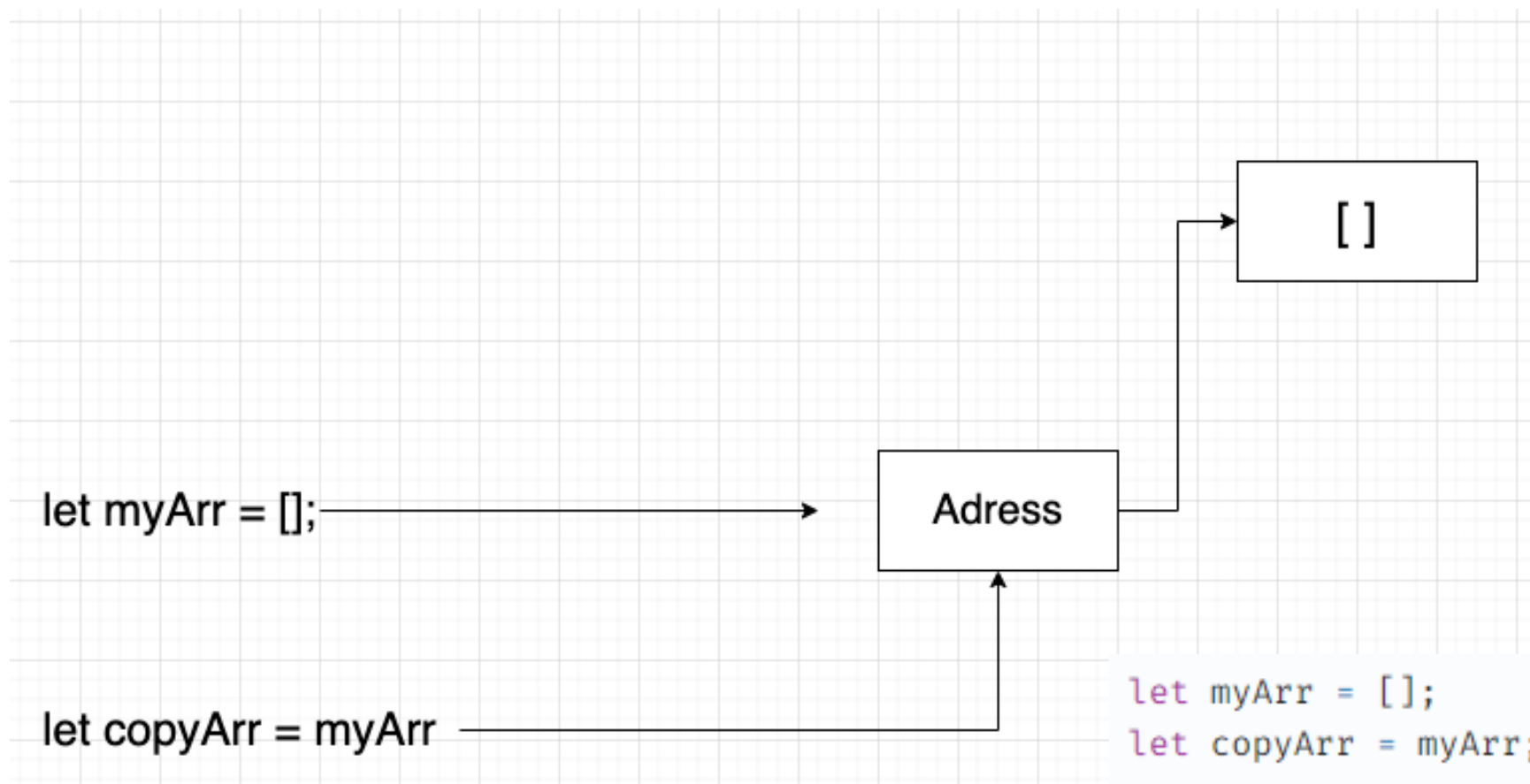
React와 불변성

원시형



```
let a = 100;  
let b = a;  
a = 50;  
  
console.log(b) // 100
```

참조형



```
let myArr = [];  
let copyArr = myArr;  
  
myArr.push("hello");  
  
console.log(copyArr); // ["hello"]
```

리액트 입장에서는 state가 참조하는
주소값이 여전히 똑같이 때문에 state가
바뀌었다고 생각하지 않는다.

이게 왜 문제가 될까?

```
import React, { useState, useEffect } from "react";

const StatePractice = () => {
  const [value, setValue] = useState([1, 2, 3, 4]);

  const handleClick = () => {
    const copy = value;
    copy.push(5);
    setValue(copy);
    console.log(copy);
    console.log(value);
  };

  return (
    <>
      <div onClick={handleClick}>{value}</div>
    </>
  );
};

export default StatePractice;
```

리엑트는 state를 참조값으로 비교해 변화를 감지한다.

참조값에 변화가 없다 ?

State 변화가 없다고 느낀다 ?

화면 리렌더링이 안된다..

리액트에서는 state 값을 변경할 때 불변성을 지켜야 한다.

변경하는 대신 복사 등의 방법을 통해 새로운 값을 만들어야 한다.

Spread Operator (...)

```
let a = [1,2,3,4,5]
let b = a;
a.push(5);
console.log(b) // [1,2,3,4,5]
let c = [...a]
a.push(6)
console.log(c) // [1,2,3,4,5]
```

Spread Operator (...)

```
const StatePractice = () => {  
  const [value, setValue] = useState([1, 2, 3, 4]);  
  
  const handleClick = () => {  
    const copy = [...value];  
    copy.push(5);  
    setValue(copy);  
  };  
  
  return (  
    <>  
      <div onClick={handleClick}>{value}</div>  
    </>  
  );  
};
```

자식 컴포넌트에서 props로 건네 받은
state를 수정하려면 ?

오늘 배운 **props + useState**
써먹어서 실습 ㄱㄱ

실습 상황

같은 버튼을 재사용 할 건데,
Component 1과 2에서 모두 0에서 시작해서
1씩 증가하도록 할래요

실습 상황

같은 버튼을 재사용 할 건데,
Component 1과 2에서 모두 1씩 증가하도록 할 건데
Component1은 0에서 시작,
Component2는 10에서 시작하고 싶어요.

실습 상황

같은 버튼을 재사용 할 건데,
Component 1에서는 3씩 증가하는,
Component 2에서는 1씩 증가하는,
버튼을 만들고 싶어요.

- 부모로부터 내려 받은 state를 수정하기 위해서는, setState를 props로 내려 받아 수정해야 한다.
- 자식은 props를 수정할 수 없다.

05

과 ㅏ ㅑ ㅓ ㅕ 제 ㅖ ㅗ ㅛ

과 제

🔗 (필수) 지난주 당근마켓 따라하기 페이지에 상품마다 “좋아요 버튼”을 달아주세요.

좋아요 버튼은 상품 별로 독립적으로 작용해야겠죠 ?

어떤 것을 props로 내려주어야 할 지

useState를 어떻게 사용할 지 잘 생각해보세요 !

버튼은 재사용을 위해 따로 컴포넌트로 분리해야 합니다.

기존 코드를 유지하지 않고, 아예 새로 뜯어 고쳐

컴포넌트 설계를 해도 좋습니다.



감사합니다

정말 고마워!