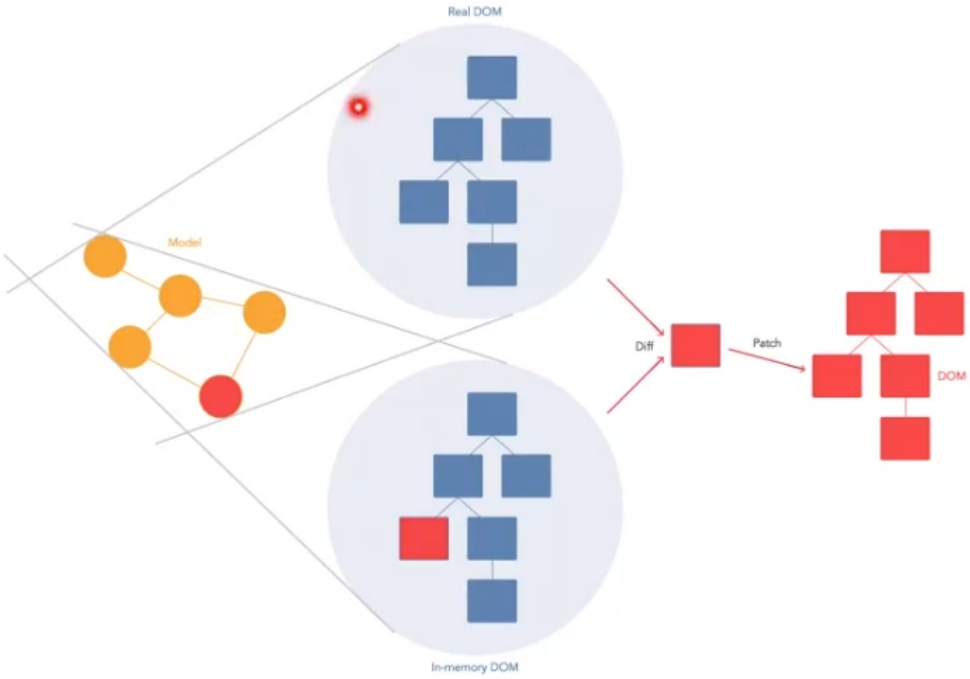
2020.12.03

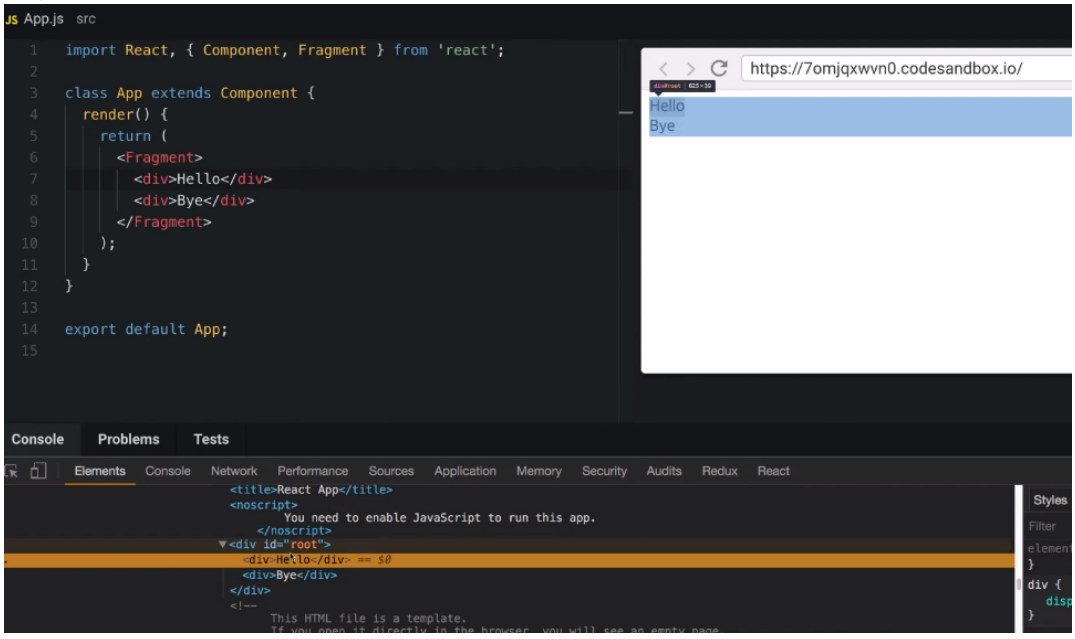
- 프론트엔드 라이브러리 react
- dom 관리와 state를 관리하기 위해 + 생산성과 유지보수성을 높이기 위해! 사용
- angular : 인지도는 성장하는 단계 ㅇㅅㅇ
- react : 컴포넌트라는 것에 집중되어있는 라이브러리 . 한가지 문제를 해결할때 해결할수 있는 방식이 많음.
- vue : 쉽다, 짬뽕

- react : 우리는 지속해서 데이터가 변화하는 대규모 애플리케이션을 구축하기 위해 리액트를 만들었습니다.
- virtual DOM : 가상의 돔
    - 변화가 필요한 곳에만 update
    - 

- web pack : 코드들을 의존하는 순서대로 하나 또는 여러개의 파일로 결과물을 만들어내게 된다.
-

- App.js

```
import React, { Component } from 'react';

class App extends Component {
  render() {
    return (
      <div>
        <h1>안녕하세요 리액트</h1>
      </div>
    );
  }
}

export default App;
```

- render 밑에는 jsx 코드를 return 해주어야 한다.
- jsx 란? html 처럼 생겼지만 javascript 로 변환된다.
  - html이랑 비슷하게 생겼지만, 지켜야 하는 규칙이 있다.
  - 태그는 무조건 닫아줘야 한다.
  - SELF CLOSING TAG
  - 모든 태그가 2개 이상이면 하나로 감싸줘야한다.
  - *fragment*를 사용하면 불필요한 div가 개발자도구(?) 에서 보이지 않는다.
  - 

이름 넣기 (값 전달하기)

- *

```
JS App.js
1  import React, { Component } from 'react';
2
3  class App extends Component {
4    render() {
5      const name = 'misung';
6      return (
7
8        <div>
9         hello {name}
10       </div>
11     );
12   }
13 }
14
15 export default App;
16
```

var const let 차이

- var은 현재 사용하지 않는다.
- const 값이 변하지 않음
- let 값이 유동적으로 변함 = > 블록단위
-

```
function foo() {
  var a = 'hello';
  if (true) {
    var a = 'bye';
    console.log(a); // bye
  }
  console.log(a); // bye
}
```

-
```
function foo() {
  let a = 'hello';
  if (true) {
    let a = 'bye';
    console.log(a); // bye
  }
  console.log(a); // hello
}
```

안에서 조건문 사용하기

-
```
JS App.js ●
1  import React, { Component } from 'react';
2
3  class App extends Component {
4    render() {
5      const name = 'misung';
6      return (
7
8        <div>{
9          1+1 ===3
10         ? '맞다'
11         : '틀리다'
12
13       }
14       </div>
15     );
16   }
17 }
18
19 export default App;
20
```
자료형이랑 값을

모두 비교하는 방식 : ===

- App.js

```
import React, { Component } from 'react';

class App extends Component {
  render() {
    const name = 'misung';
    return (

      <div>{

        name ==='misung' &&<div>미성이다</div>
      }
      </div>
    );
  }
}

export default App;

```

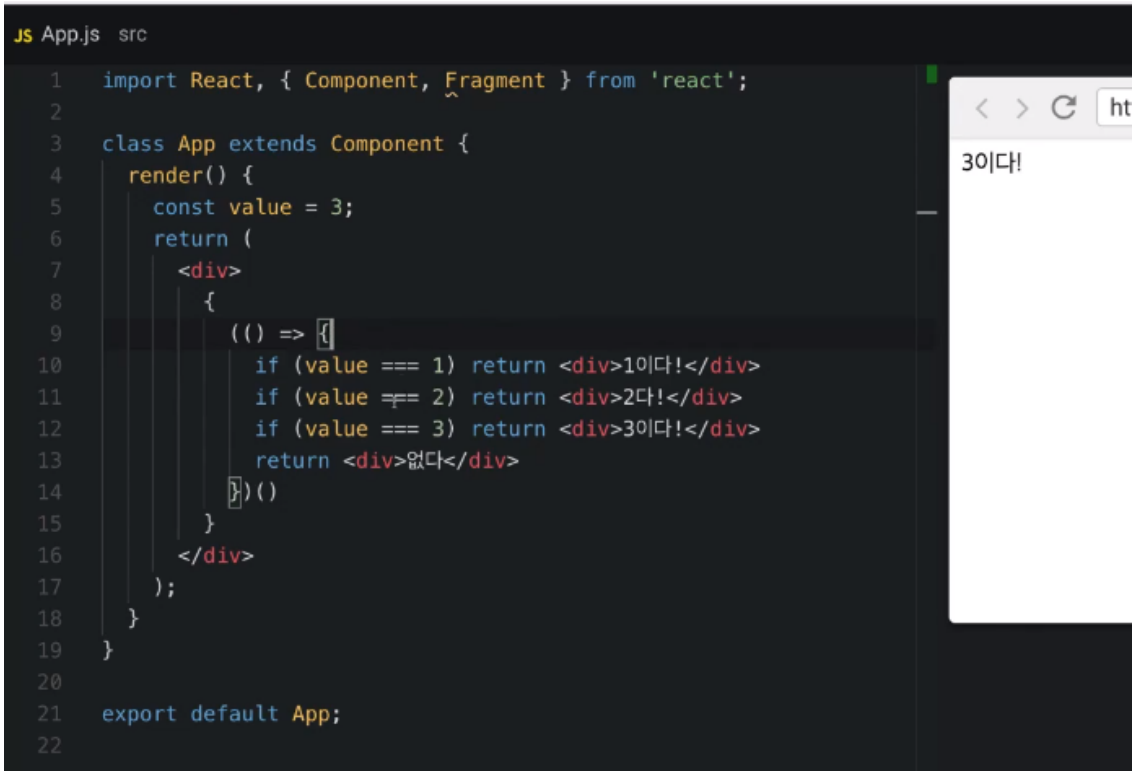
- App.js src

```
import React, { Component, Fragment } from 'react';

class App extends Component {
  render() {
    const value = 3;
    return (
      <div>
        {
          (function() {
            if (value === 1) return <div>1이다!</div>
            if (value === 2) return <div>2다!</div>
            if (value === 3) return <div>3이다!</div>
            return <div>없다</div>
          })()
        }
      </div>
    );
  }
}

export default App;

```

https://7omjqxwvn0.codesandb

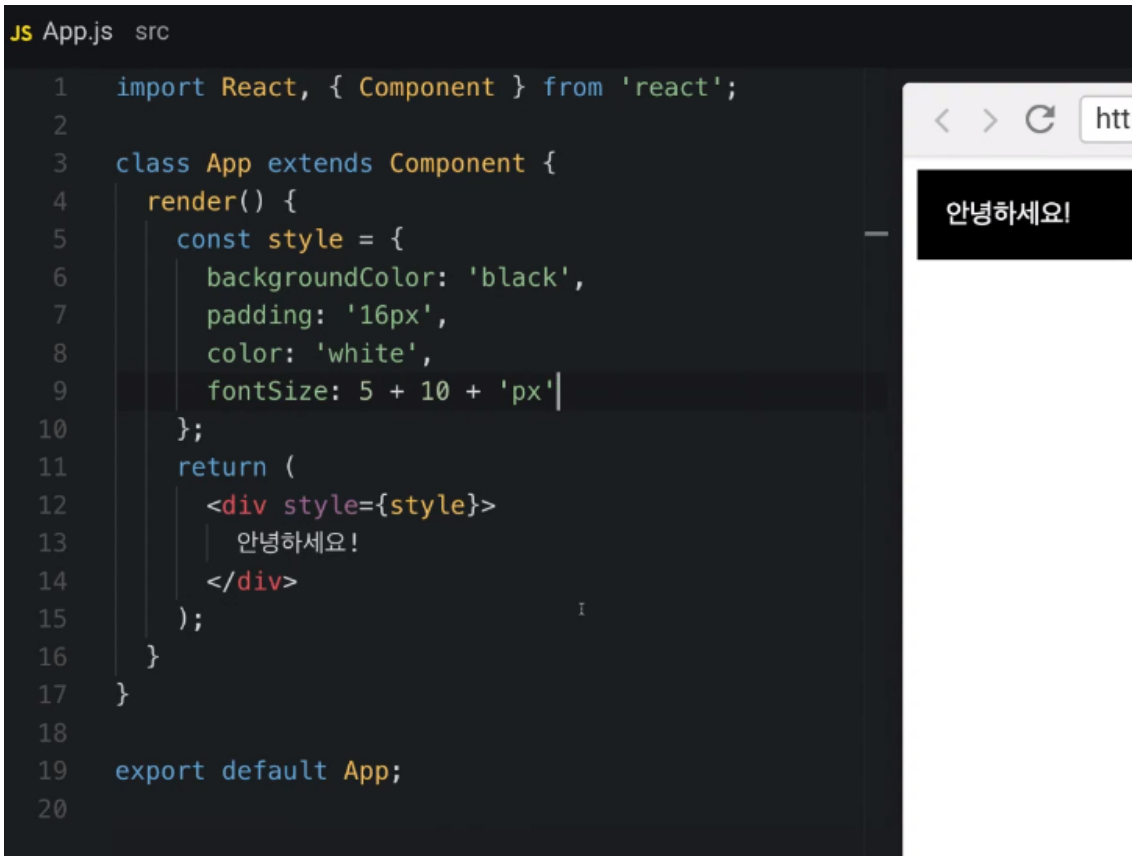
3이다!

- 함수를 감싸주고 , 바로 호출한다. ()

- 

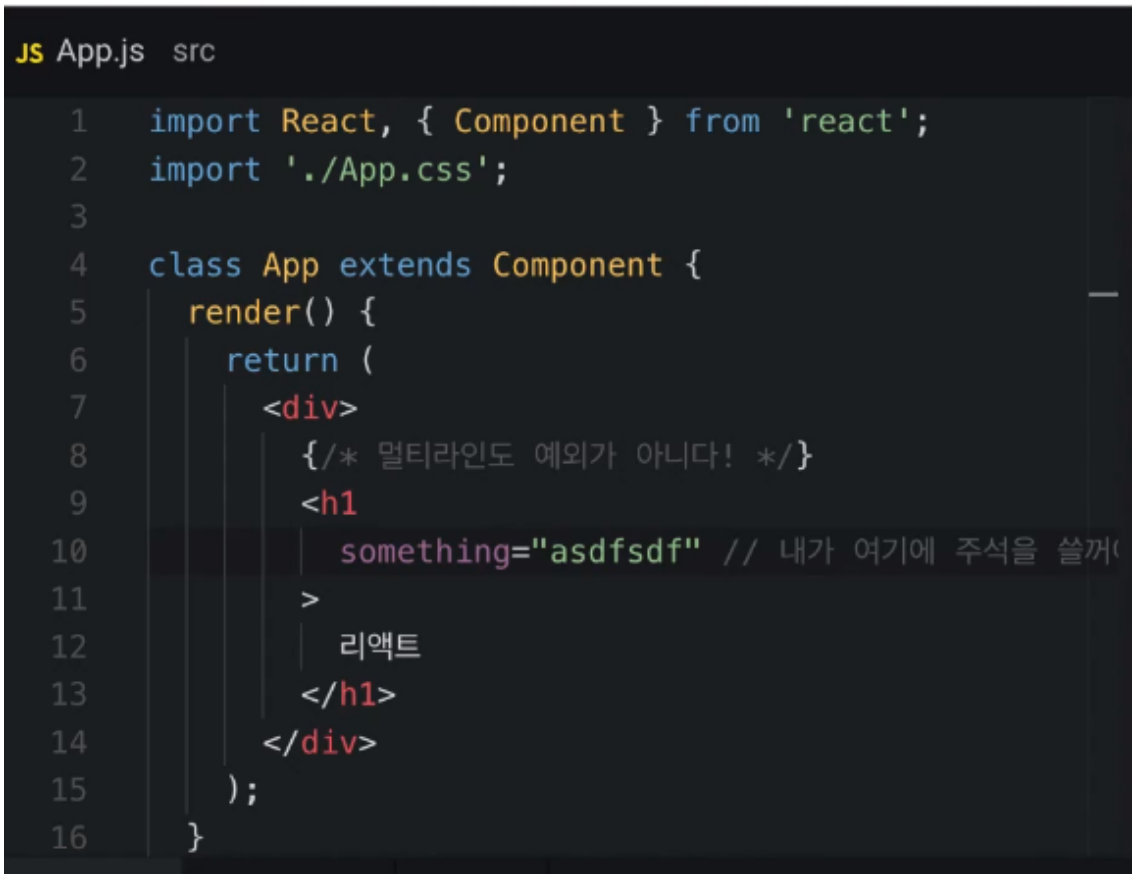
- 이런식으로 화살표 함수로도 표현할수 있다.

스타일 넣기!!

- 

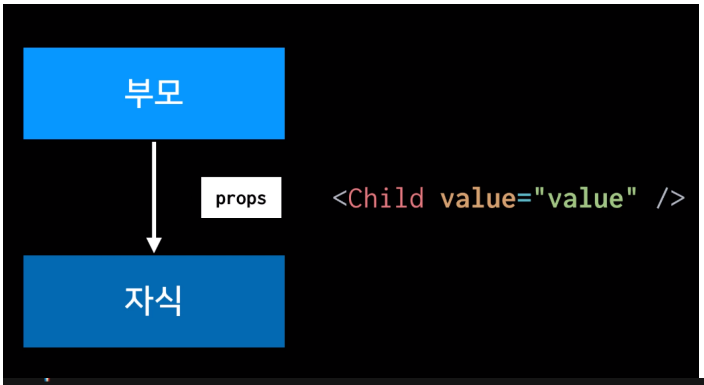
- 

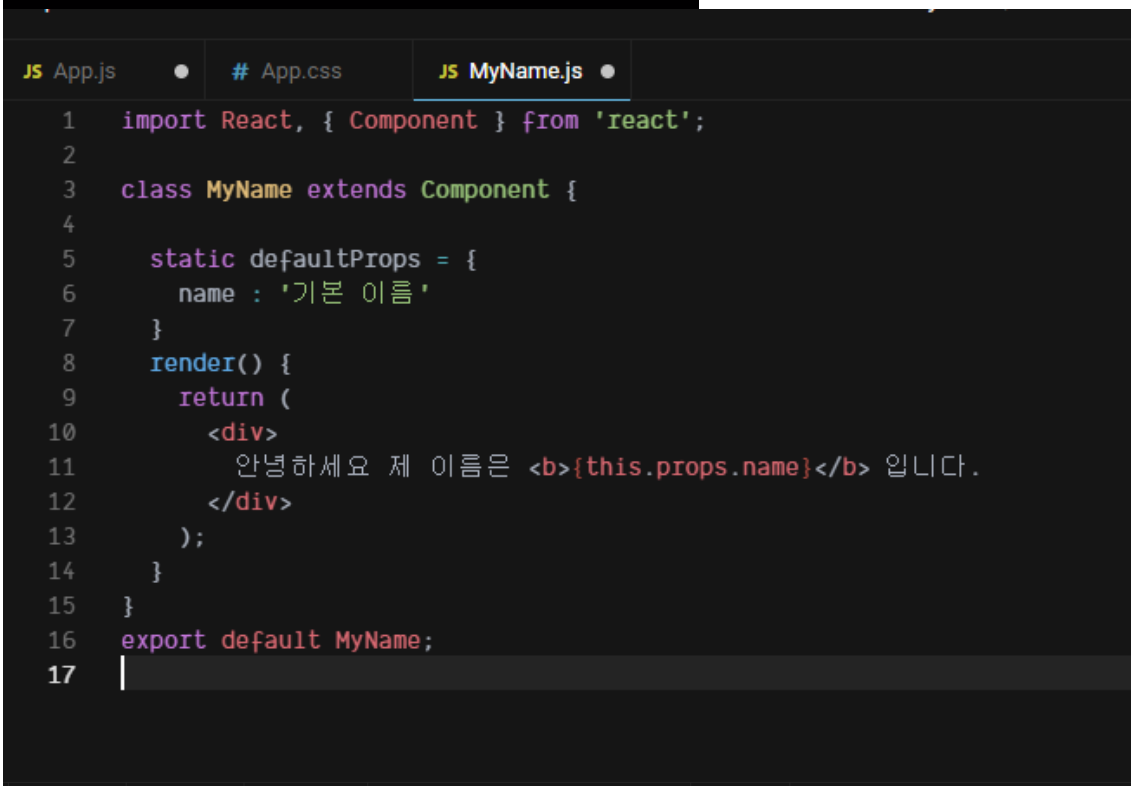
주석 사용하기

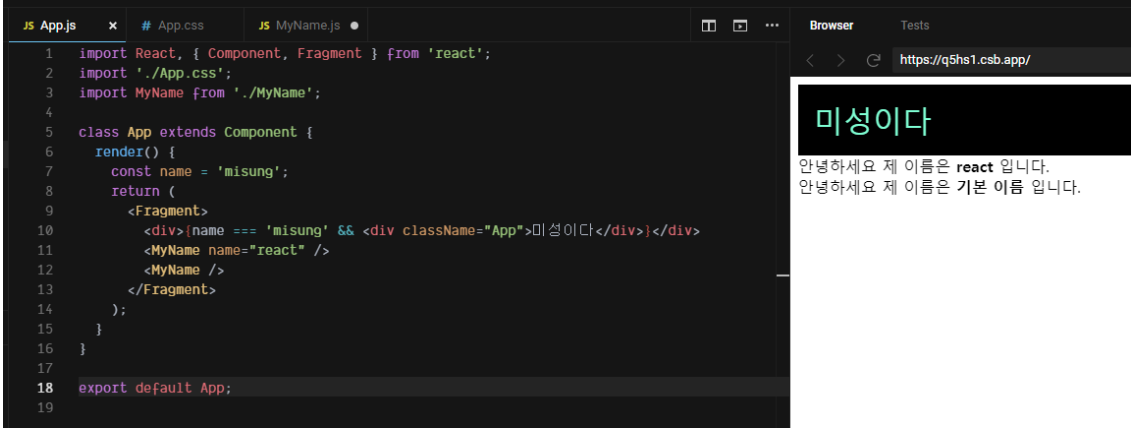
- 

props를 사용하는 방법

- 부모 컴포넌트가 자식 컴포넌트에게 값을 전달할때 사용한다.
- 값을 넣어주는것을 props 라고 한다.

- 

- 

- 

-

함수형

- 초기 마운트 속도가 빠르고 , 메모리 자원을 좀 덜 사용한다.
- 지금은 Hook 을 사용해서 클래스로만 가능했던게 함수형에서도 모두 구현이 가능함.
- 그래서 함수형도 좋은거같음.. 코드도 간결하고? ?.?
-

- MyName.js src

```
import React, { Component } from 'react';

const MyName = ({ name }) => {
  return <div>안녕하세요! 제 이름은 {name} 입니다.</div>;
};

MyName.defaultProps = {
  name: 'velopert'
};

export default MyName;
```

https://4r6lqrlvj9.codesandbox.io

안녕하세요! 제 이름은 velopert 입니다.

state

- 자기가 처음부터 가지고 있는 값, ,,
- setState를 사용해서 값을 변경한다.
-