



---

# ES6

2020.11.27

Copyright©. Saebyeol Yu. All Rights Reserved.

## CONTENTS

- 001 React가 생긴 이유
- 002 React 프로젝트 생성
- 003 ES6(객체 구조 분해)
- 004 ES6(단축 평가 논리 계산법)

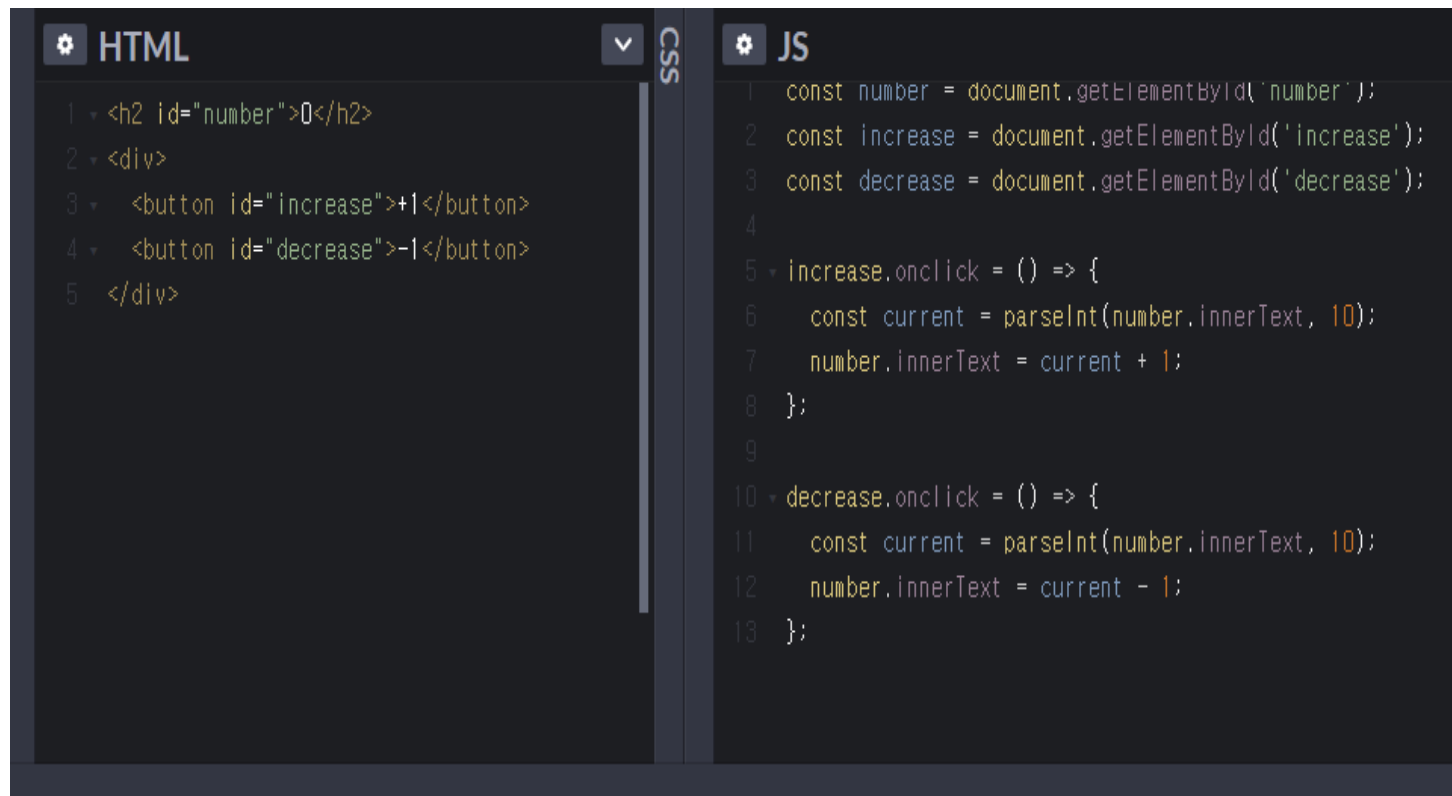
# Part 1.

---

## React가 생긴 이유



# 1 리액트가 생긴 이유



```
HTML
1 <h2 id="number">0</h2>
2 <div>
3   <button id="increase">+1</button>
4   <button id="decrease">-1</button>
5 </div>

CSS

JS
1 const number = document.getElementById('number');
2 const increase = document.getElementById('increase');
3 const decrease = document.getElementById('decrease');
4
5 increase.onclick = () => {
6   const current = parseInt(number.innerText, 10);
7   number.innerText = current + 1;
8 };
9
10 decrease.onclick = () => {
11   const current = parseInt(number.innerText, 10);
12   number.innerText = current - 1;
13 };
```

0

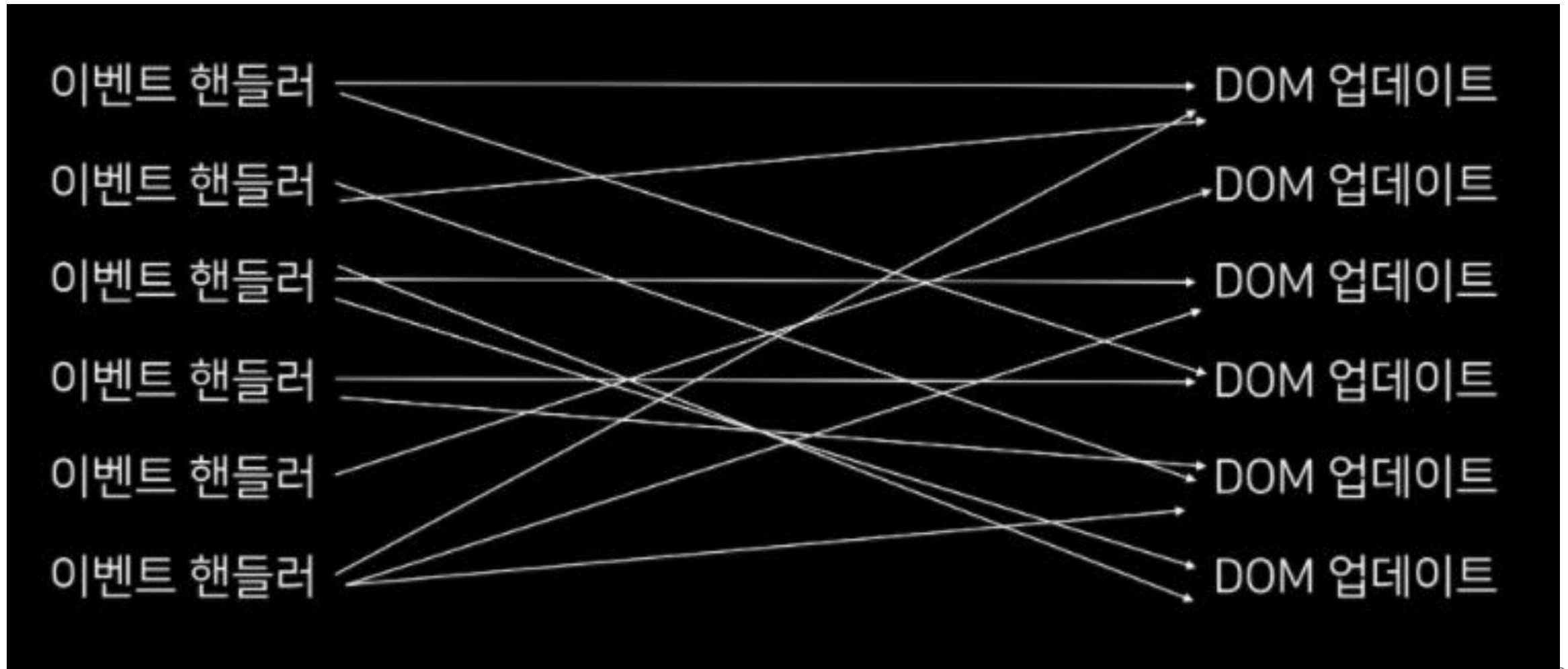
+1 -1



JS를 사용해서 HTML로 구성한 UI를 제어

1. 브라우저의 DOM Selector API를 사용
2. 특정 DOM을 선택
3. 특정 이벤트 설정

# 1 리액트가 생긴 이유



# 1 리액트가 생긴 이유



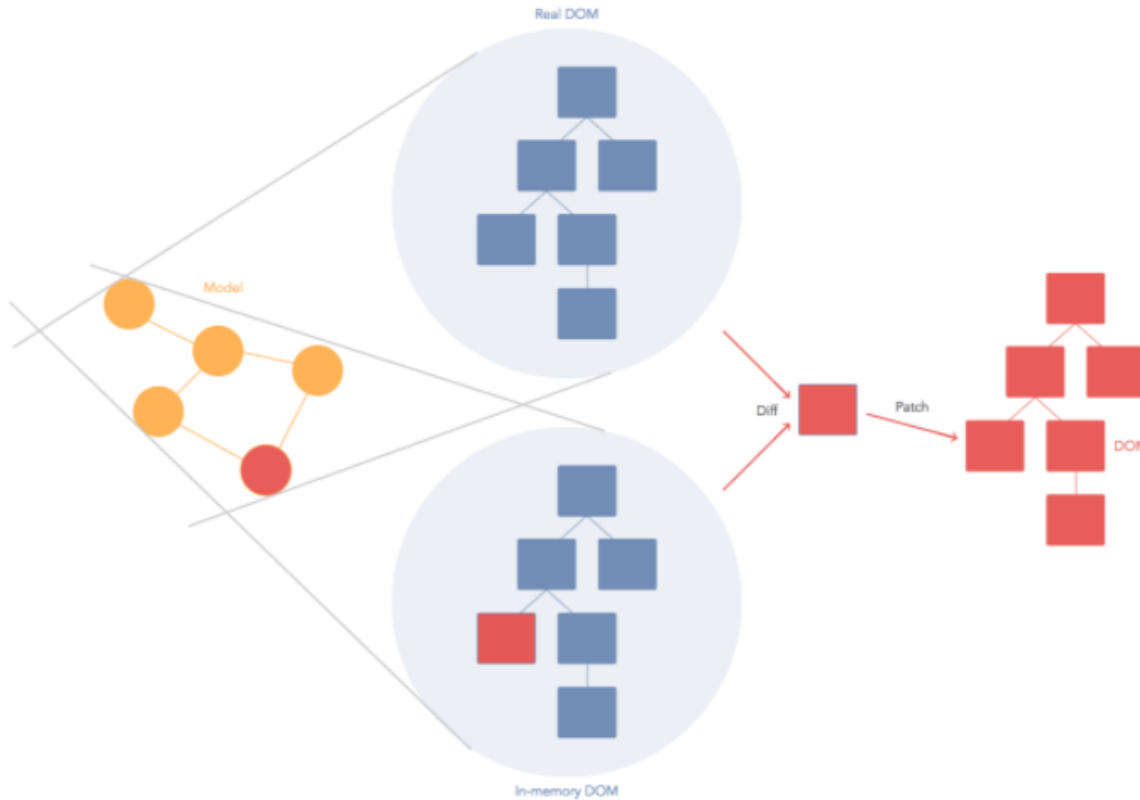
AngularJS는 JS의 특정 값이 바뀌면 특정 DOM의 속성이 바뀌도록 연결  
(업데이트 작업을 간소화)



React는 상태가 바뀌었을때 업데이트 규칙을 정하는 것이 아니라 다 날리고 처음부터 모든걸 새로 구성  
(업데이트를 어떻게 해야하는지 고민이 없기때문에 개발이 쉬워짐 -> 속도가 느려짐)

## Virtual DOM

# 1 리액트가 생긴 이유



1. 메모리에 가상으로 존재하는 DOM  
(브라우저에 실제로 보여지는 DOM이 아님)
2. JS 객체이기 때문에 작동 성능이 실제 브라우저에서 DOM을 보여주는 것보다 훨씬 빠름
3. 업데이트 되면 UI를 Virtual DOM을 통해 렌더링
4. 리액트의 효율적인 비교 알고리즘을 통해서 브라우저에 보여지고 있는 DOM과 비교후 차이가 있는 곳을 감지해서 실제 DOM에 패치

# Part 2.

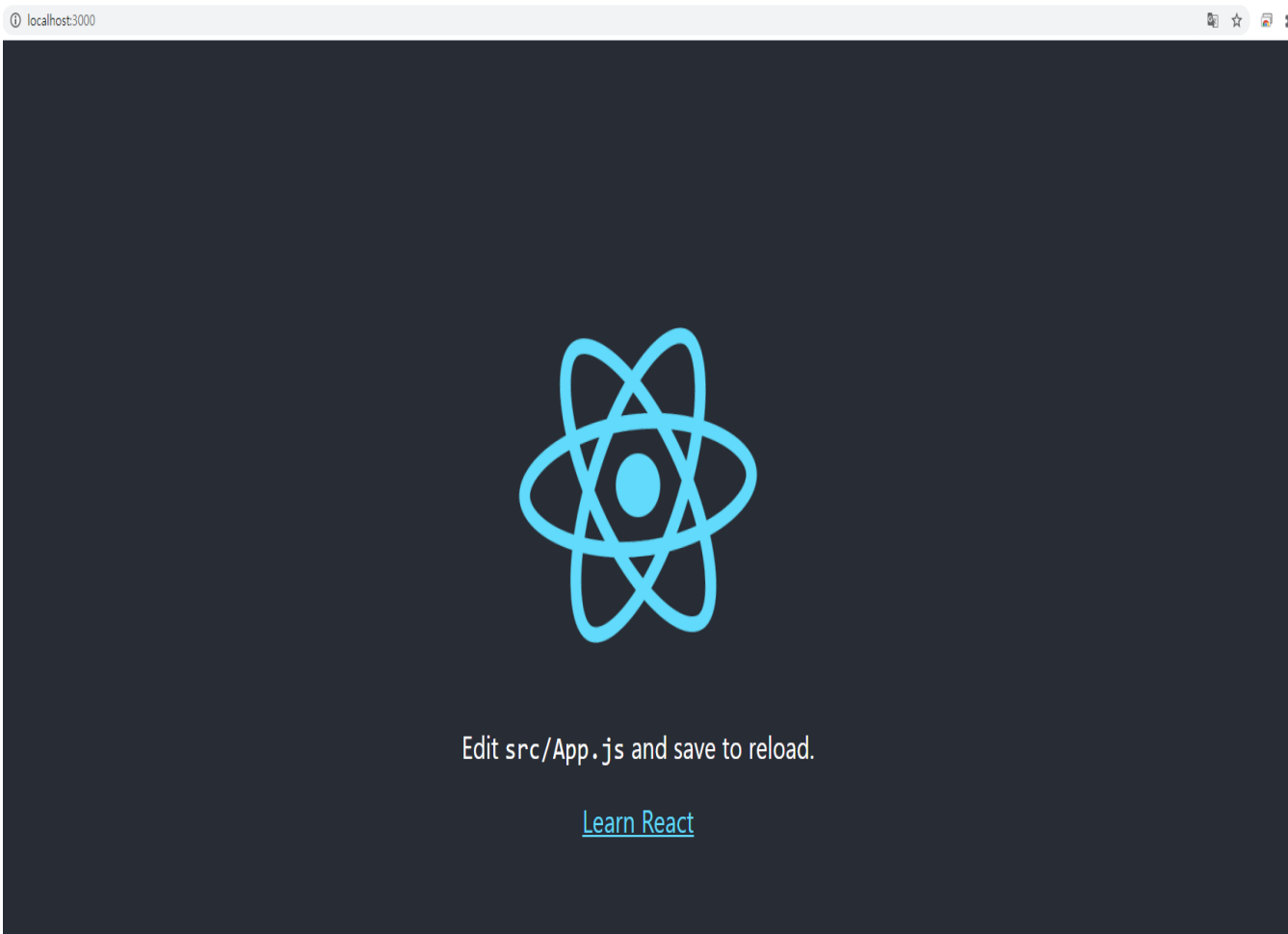
---

## React 프로젝트 생성





## 2 React 프로젝트 생성



### 프로그램 설치

1. Node.js 설치(Webpack과 Babel 사용)
2. npm(노드 패키지 매니저)
3. VSCode

### Webpack

여러가지의 파일을 한개로 결합하기 위해서 사용

### Babel

JSX 문법과 ES6 문법을 변환하기 위해서 사용

### 프로젝트 만들기(실습)

# Part 3.

---

## ES6(객체 구조 분해 - 실습)



# Part 4.

---

## ES6(단축 평가 논리 계산법)



### 3 ES6(단축 평가 논리 계산법)

```
console.log(true && true); //true
console.log(true && false); //false
console.log(true || false); //true
console.log(false || true); //true
```

```
//ES6, IE 0, 크롬 0
console.log(true && 'hello'); //hello
console.log(false && 'hello'); //false
console.log('hello' && 'bye'); //bye
console.log(null && 'hello'); //null
console.log(undefined && 'hello'); //undefined
console.log('' && 'hello'); //''
console.log(0 && 'hello'); //0
console.log(1 && 'hello'); //hello
console.log(1 && 1); //1
```

### 3 ES6(단축 평가 논리 계산법 - AND)

```
const dog = {  
  name: '멍멍이'  
};
```

```
function getName(animal) {  
  if (animal) {  
    return animal.name;  
  }  
  return undefined;  
}
```

```
const name = getName();  
console.log(name);
```

```
const dog = {  
  name: '멍멍이'  
};
```

```
function getName(animal) {  
  return animal && animal.name;  
}
```

```
const name = getName();  
console.log(name); // undefined
```

A && B 연산자를 사용하면  
A가 Truthy한 값이면 B,  
A가 Falsy한 값이면 A

### 3 ES6(단축 평가 논리 계산법 – OR)

```
const namelessDog = {
  name: ''
};

function getName(animal) {
  const name = animal && animal.name;
  if (!name) {
    return '이름이 없는 동물입니다.';
  }
  return name;
}

const name = getName(namelessDog);
console.log(name); // 이름이 없는 동물입니다.
```

A || B 연산자를 사용하면  
A가 Truthy한 값이면 A,  
A가 Falsy한 값이면 B

```
//어떤 값이 Falsy한 값이면 대체 값을 지정시 유용
const namelessDog = {
  name: ''
};

function getName(animal) {
  const name = animal || '이름이 없는 동물입니다.';
  return name;
}

const name = getName(namelessDog);
console.log(name); // 이름이 없는 동물입니다.
```

감사합니다