3. 연산자

연산자에 대해서 알아봅시다.

연산자는 프로그래밍 언어에서 특정 연산을 하도록 하는 문자입니다.

예를 들어서, 우리가 변수와 상수를 배울 때 다음과 같은 코드를 작성했었지요?

```
let value = 1; // 변수 선언
value = 2; // 대입 연산자
```

여기서 두번째 줄에서 사용된 = 문자가 바로 연산자입니다. 연산자의 종류는 굉장히 많습니다. 그 중에서 = 는, 대입 연산자에 해당합니다. 첫번째 줄은 새로운 변수를 선언하는 것으로서, 대입 연산자에 해당하지 않습니다.

산술 연산자

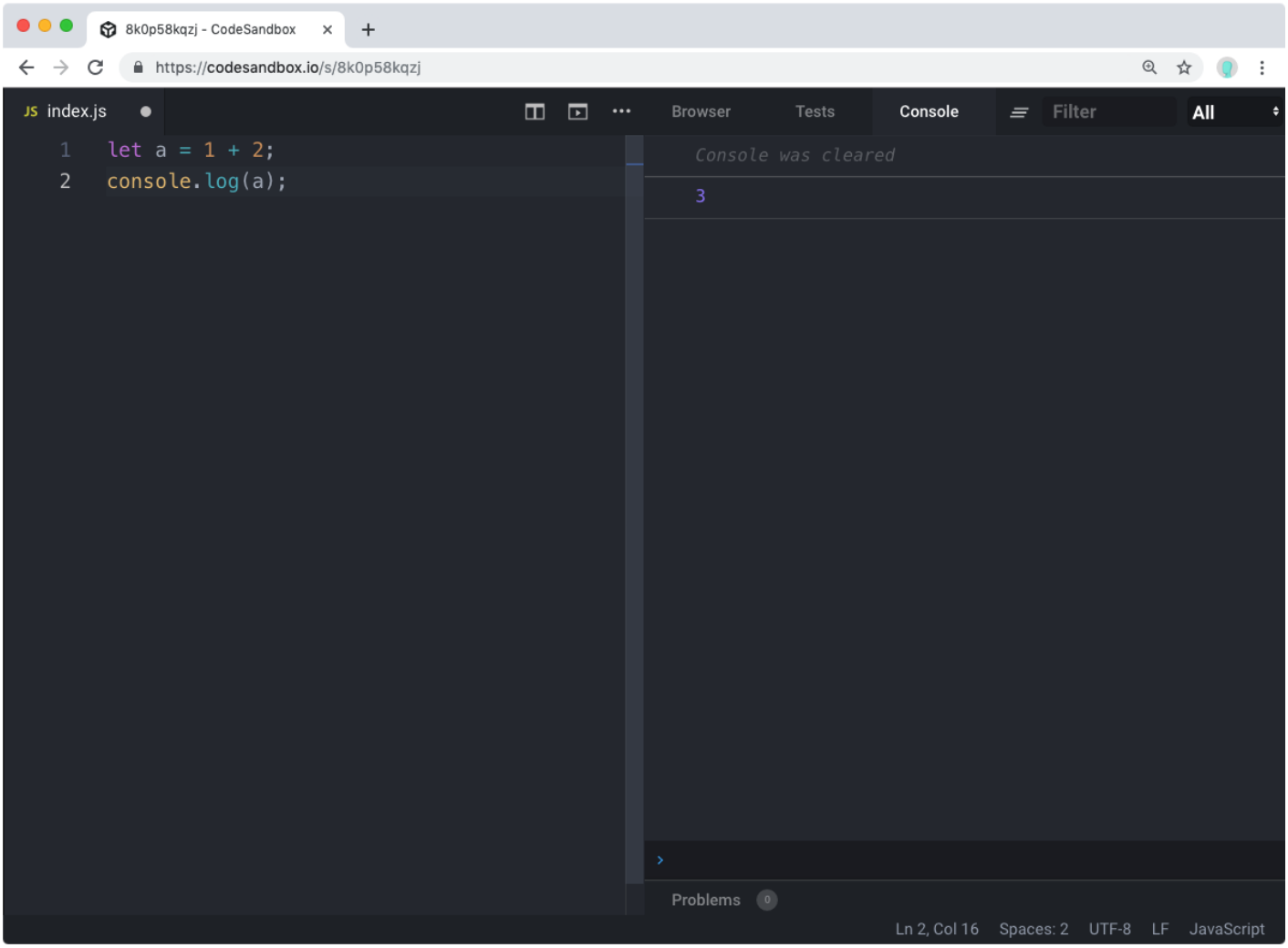
산술 연산자는 사칙연산과 같은 작업을 하는 연산자를 의미합니다.

- +: 덧셈
- -: 뺄셈
- *: 곱셈
- /: 나눗셈

위 4가지가 가장 기본적인 산술 연산자입니다. 이 외에도 몇가지가 더 있는데요, 먼저 위 연산자들을 한번 사용해봅시다.

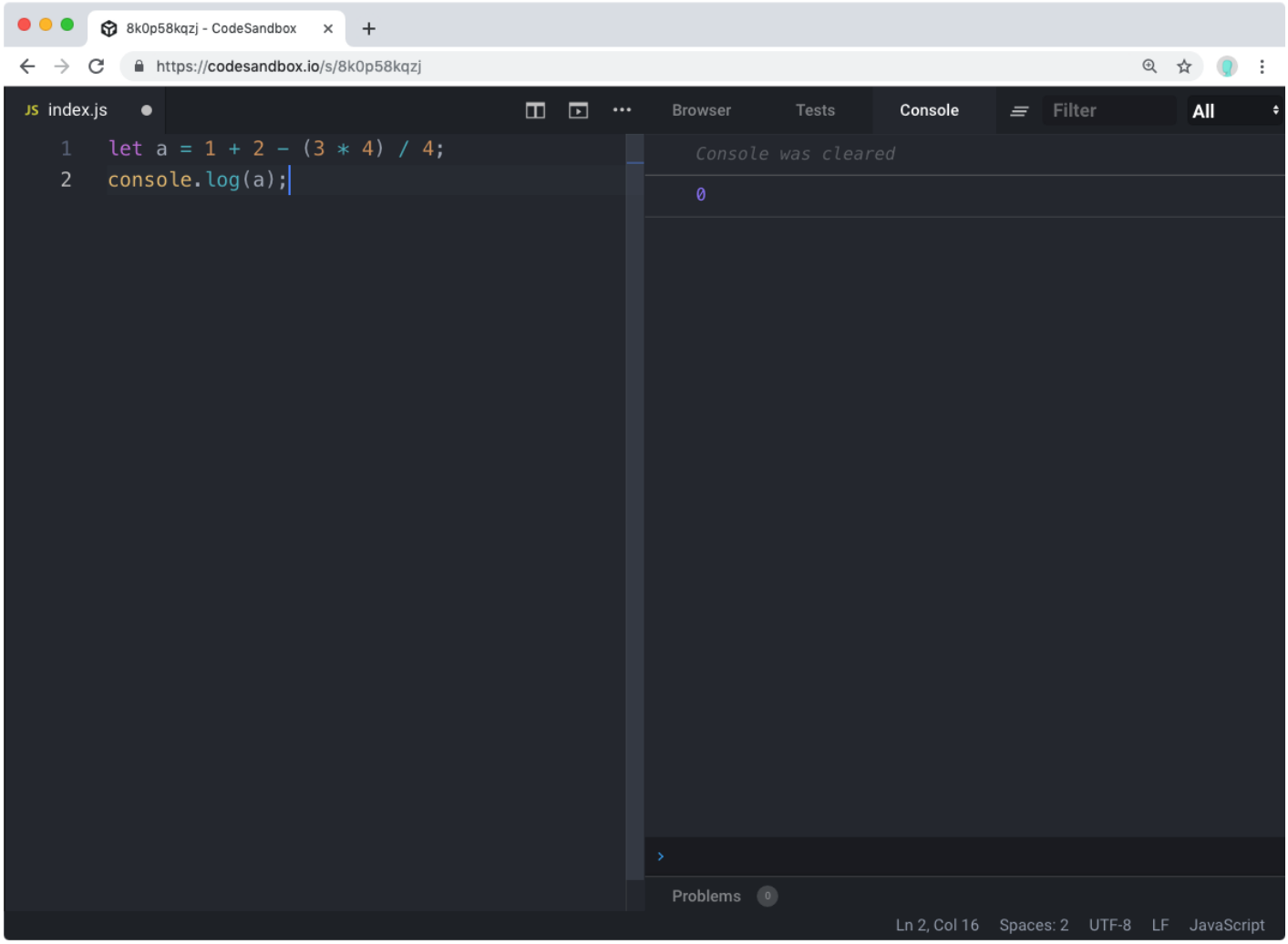
```
let a = 1 + 2;
console.log(a);
```

위 코드는 a 값을 선언 할 때 1 + 2 의 결과물을 담습니다. 따라서 콘솔에선 3 이라는 숫자가 나타나겠죠.



```
let a = 1 + 2 - (3 * 4) / 4;
console.log(a);
```

이렇게 우리가 계산기를 두드릴 때 처럼 할 수도 있습니다.

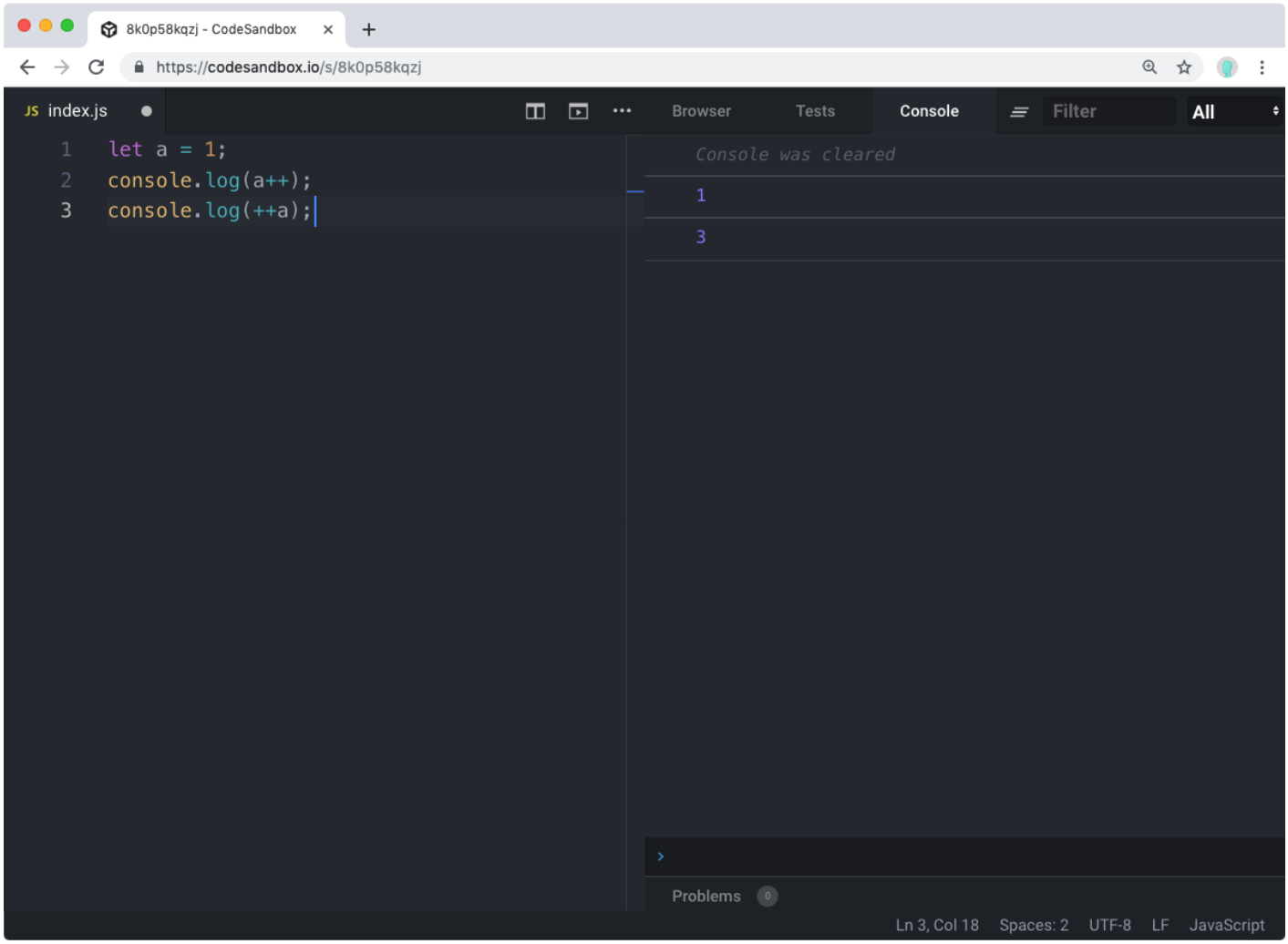


그리고, 이런 것도 할 수 있습니다. 이 또한 산술 연산자의 일부입니다.

```
let a = 1;
a++;
++a;
console.log(a);
```

결과는 3이 나타납니다. ++ 는 특정 변수에 1을 바로 더해줍니다. 그런데, ++ 가 변수 이름 앞에 오는 것과 뒤에 오는것에 차이가 있습니다.

```
let a = 1;
console.log(a++);
console.log(++a);
```



`console.log(a++);` 를 할 때에는 1을 더하기 직전 값을 보여주고 `console.log(++a);` 를 할 때에는 1을 더한 다음의 값을 보여준다는 차이가 있습니다.

덧셈 외에도 뺄셈도 똑같이 할 수 있습니다.

```
let a = 1;
a--;
console.log(a);
```

결과는 0이 나타나게 됩니다.

대입 연산자

대입 연산자는 특정 값에 연산을 한 값을 바로 설정 할 때 사용 할 수 있는 연산자입니다. 예를 들어서 다음과 같은 코드가 있다면

```
let a = 1;
a = a + 3;
```

위 코드를 대입 연산자를 사용하면 다음과 같이 작성할 수 있습니다.

```
let a = 1;
a += 3;
```

덧셈 말고 다른 연산도 가능합니다.

```
let a = 1;
a += 3;
a -= 3;
a *= 3;
a /= 3;
console.log(a);
```

결과는 1이 나타나게 됩니다.

논리 연산자

논리 연산자는, 불리언 타입 (true 혹은 false)를 위한 연산자입니다. 논리 연산자는 다음 장에서 우리가 if 문을 배울 때 매우 유용합니다.

총 3가지가 있습니다.

- `!` : NOT
- `&&` : AND
- `||` : OR

OR의 경우엔 엔터키 위의 \ 문자를 Shift 누른 상태로 누르면 됩니다.

NOT

NOT 연산자는 true 는 false 로, false 는 true 로 바꿔줍니다.

```
const a = !true;
console.log(a);
```

a 값은 false 입니다.

```
const b = !false;
console.log(b);
```

b 값은 true 가 됩니다.

AND

AND 연산자는 양쪽의 값이 둘 다 true 일때만 결과물이 true 입니다.

```
const a = true && true;
console.log(a);
```

다음과 같은 상황엔 모두 false 입니다.

```
let f = false && false;
f = false && true;
f = true && false;
```

OR

OR 연산자는 양쪽의 값 중 하나라도 true 라면 결과물이 true 입니다. 그리고, 두 값이 둘 다 false 일 때에만 false 입니다.

다음 상황엔 t 값은 true 입니다.

```
let t = true || false;
t = false || true;
t = true || true;
```

반면 상황엔 false 입니다.

```
let f = false || false;
```

연산 순서

사칙연산을 할 때 곱셈 나눗셈이 먼저고 그 다음이 덧셈 뺄셈인 것 처럼, 논리 연산자도 순서가 있습니다. 순서는 NOT -> AND -> OR 입니다. 예를 들어 다음과 같은 코드가 있다고 가정해봅시다.

```
const value = !((true && false) || (true && false) || !false);
```

괄호로 감싸져있을 때에는 가장 마지막에 처리를 하니까 맨 앞 NOT 은 나중에 처리하겠습니다.

우선 NOT 을 처리합니다.

```
!((true && false) || (true && false) || true);
```

그 다음엔 AND 를 처리합니다.

```
!(false || false || true);
```

OR 연산자를 좌측에서 우측 방향으로 처리를 하게 되면서 다음과 같이 처리됩니다.

```
!true;
```

결국 결과값은 false 가 됩니다.

비교 연산자

비교 연산자는 두 값을 비교 할 때 사용 할 수 있습니다.

두 값이 일치하는지 확인

```
const a = 1;
const b = 1;
const equals = a === b;
console.log(equals);
```

`===` 는 두 값이 일치하는지 확인해줍니다. 일치한다면, true, 일치하지 않는다면 false 가 나타납니다. 위 코드의 경우엔 true 가 나타나겠죠?

두 값이 일치 하는지 확인 할 때 `=` 문자를 3번 사용하는데요, 2개로도 비교를 할 수는 있습니다.

```
const a = 1;
const b = 1;
const equals = a == b;
console.log(equals);
```

위 코드는 똑같은 결과 true 를 반환하긴 하는데요, `=` 문자가 3개 있을 때와 2개 있을 때의 차이점은 2개 있을때에는 타입 검사까지는 하지 않는다는 것입니다.

예를 들어서 `==` 를 사용하면 숫자 1과 문자 '1' 이 동일한 값으로 간주됩니다.

```
const a = 1;
const b = '1';
const equals = a == b;
console.log(equals);
```

결과: true

그리고, 0 과 false 도 같은 값으로 간주되지요.

```
const a = 0;
const b = false;
const equals = a == b;
console.log(equals);
```

결과: true

그리고 undefined 와 null 도 같은 값으로 간주됩니다.

```
const a = null;
const b = undefined;
const equals = a == b;
console.log(equals);
```

결과: true

앞으로 여러분이 두 값이 일치하는지 비교 할 때에는 `==` 대신 `===` 를 사용 할 것을 권장 드립니다. `==` 를 사용하다보면 실수를 할 확률이 높아집니다.

두 값이 일치하지 않는지 확인

두 값이 일치하지 않는지 확인 할 때에는 `!==` 를 사용하면 됩니다.

```
const value = 'a' !== 'b';
```

결과물은 true 가 됩니다.

`!=` 를 사용하게 되면 타입 검사를 하지 않습니다.

```
console.log(1 != '1');
console.log(1 !== '1');
```

처음엔 false, 두번째에서는 true가 나타납니다. 두 값이 일치하지 않는지 확인 할 때에도, !== 를 사용 할 것을 권장드립니다.

크고 작음

두 값 중에서 무엇이 더 크고 작은지 비교하기 위해서는 다음 연산자를 사용 할 수 있습니다.

```
const a = 10;
const b = 15;
const c = 15;

console.log(a < b); // true
console.log(b > a); // true
console.log(b >= c); // true
console.log(a <= c); // true
console.log(b < c); // false;
```

위 코드에서 `// false` 이런 식으로 코드의 뒷부분에 텍스트가 적혀있는데요, 이는 주석이라고 부릅니다. 코드에 메모를 다는 것이죠.

여러줄로 주석을 작성 할 때에는 다음과 같이 할 수 있습니다.

```
/*
  여러줄로 주석 작성하기
*/
```

문자열 붙이기

두 문자열을 붙일 때에는 `+` 로 할 수 있습니다.

```
const a = '안녕';
const b = '하세요';
console.log(a + b); // 안녕하세요
```