

08. 반복문

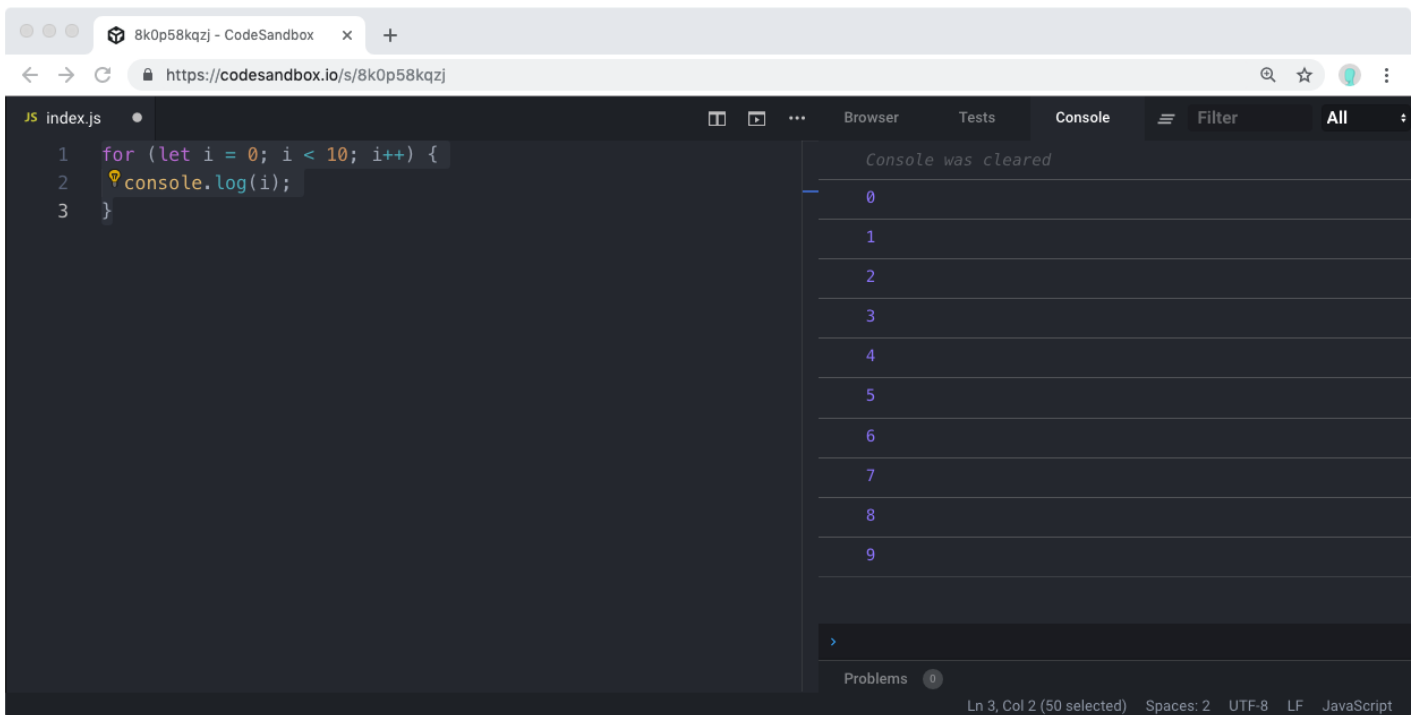
반복문은 특정 작업을 반복적으로 할 때 사용할 수 있는 구문입니다.

for

for 문은 가장 기본적인 반복문입니다. 특정 값에 변화를 주어가면서 우리가 정한 조건이 만족된다면 계속 반복합니다.

한번 다음 코드를 따라 적어보세요.

```
for (let i = 0; i < 10; i++) {  
  console.log(i);  
}
```



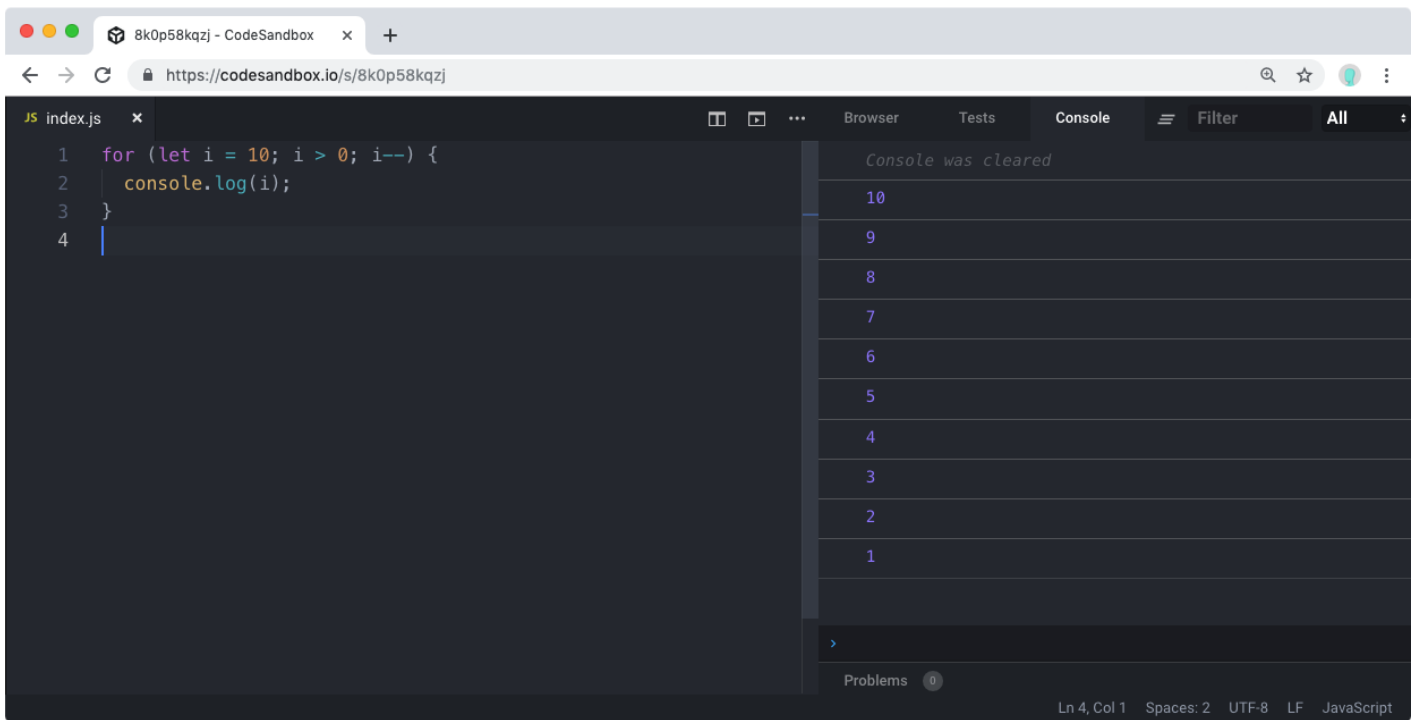
결과가 0부터 9까지 잘 나타났나요?

for 문을 다음과 같이 사용합니다.

```
for (초기 구문; 조건 구문; 변화 구문;) {  
  코드  
}
```

for 문을 사용 할 때 보통 `i++` 를 해서 1씩 증감하는 형태로 사용합니다. 그런데, 1씩 빼는 형태도 가능합니다. 다음 코드를 한번 실행해 보세요.

```
for (let i = 10; i > 0; i--) {  
  console.log(i);  
}
```



```
JS index.js x
1 for (let i = 10; i > 0; i--) {
2   console.log(i);
3 }
4
```

Console

Console was cleared

10

9

8

7

6

5

4

3

2

1

Problems 0

Ln 4, Col 1 Spaces: 2 UTF-8 LF JavaScript

10부터 1까지 결과가 잘 나타났나요?

for 문은 이렇게 숫자에 변화를 주어가면서 반복적으로 작업을 실행합니다.

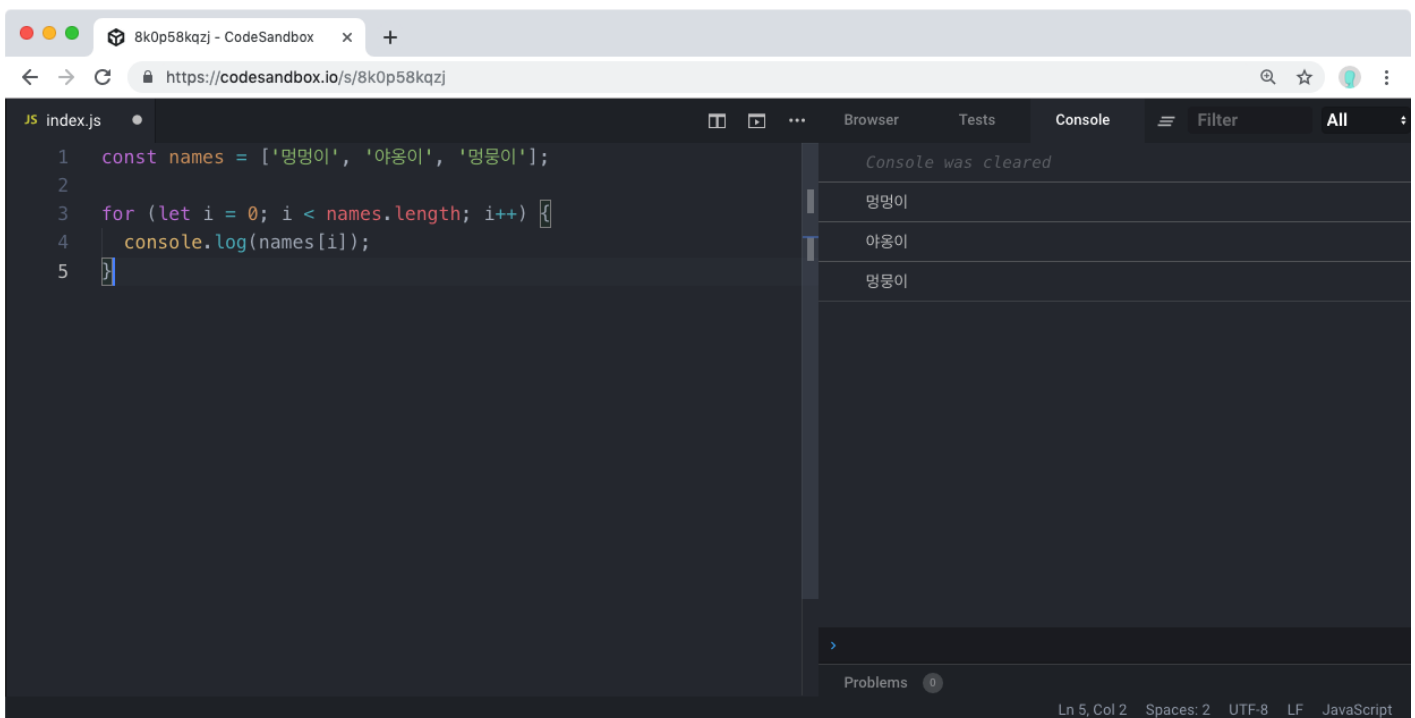
배열과 for

이번에는 우리가 이전에 배운 배열과 for 문을 함께 활용해보겠습니다. 다음 코드를 작성해보세요.

```
const names = ['멍멍이', '야옹이', '멍뽕이'];

for (let i = 0; i < names.length; i++) {
  console.log(names[i]);
}
```

이렇게 하면 names 배열 안에있는 원소들을 하나하나 나열 할 수 있습니다.



```
JS index.js
1 const names = ['멍멍이', '야옹이', '멍뽕이'];
2
3 for (let i = 0; i < names.length; i++) {
4   console.log(names[i]);
5 }
```

Console

Console was cleared

멍멍이

야옹이

멍뽕이

Problems 0

Ln 5, Col 2 Spaces: 2 UTF-8 LF JavaScript

while

while문은 특정 조건이 참이라면 계속해서 반복하는 반복문입니다. for 문은 특정 숫자를 가지고 숫자의 값을 비교하고, 증감해주면서 반복을 한다면, while문은 조건을 확인만 하면서 반복을 합니다. 때문에, 조건문 내부에서 변화를 직접 주어야 합니다.

우리가 가장 처음 작성했던 0 부터 9 까지 콘솔에 출력력을하는 반복문을 while 문으로 구현해보겠습니다.

```
let i = 0;
while (i < 10) {
  console.log(i);
  i++;
}
```

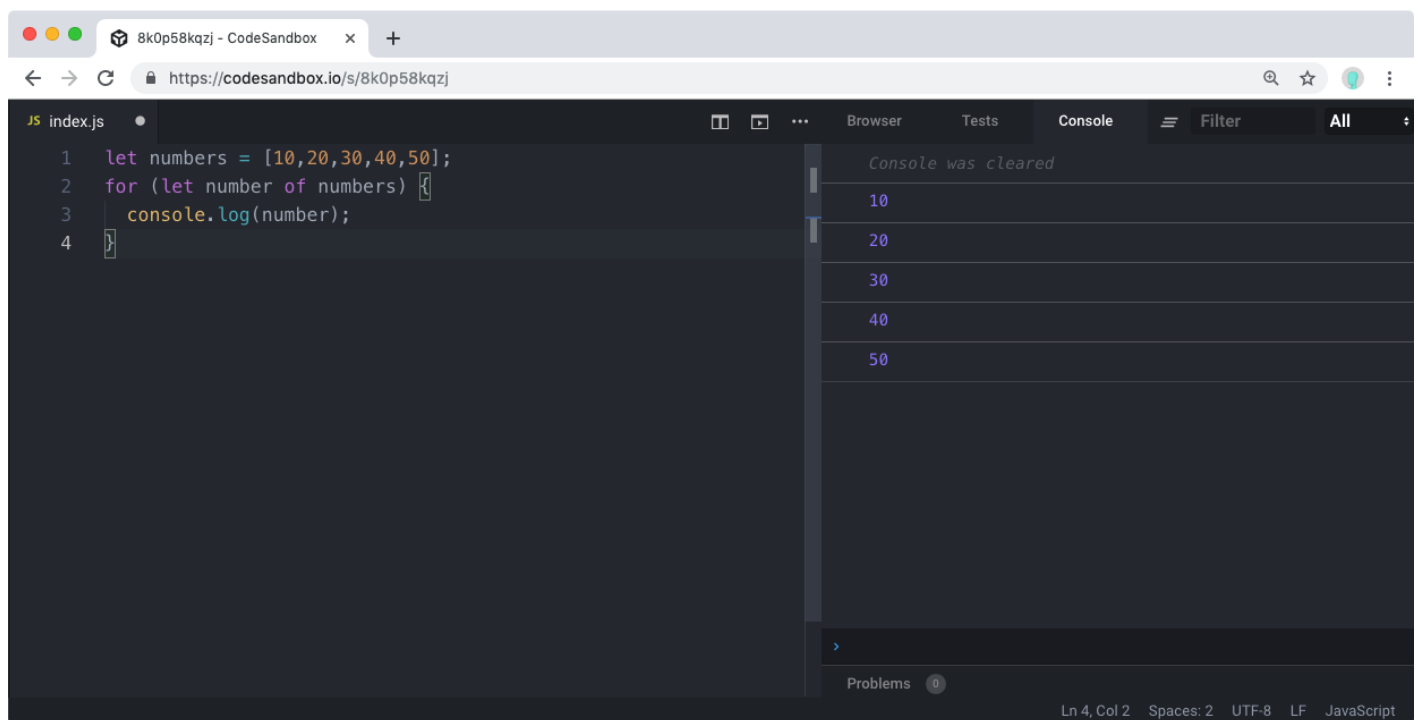
while 문을 사용 할 때에는 조건문이 언젠간 false 가 되도록 신경쓰셔야 합니다. 만약에 언젠간 false 로 전환이 되지 않는다면 반복문이 끝나지 않고 영원히 반복됩니다.

for...of

for...of 문은 배열에 관한 반복문을 돌리기 위해서 만들어진 반복문입니다.

사실 이 구문은 배워봐도 사용 할 일이 별로 없습니다. 보통 배열을 반복할때에는 배열의 내장함수를 많이 사용합니다. 그래도 알아는 둡시다.

```
let numbers = [10, 20, 30, 40, 50];
for (let number of numbers) {
  console.log(number);
}
```



객체를 위한 반복문 for...in

객체를 위한 반복문을 알아보기 전에, 객체의 정보를 배열 형태로 받아올 수 있는 함수 몇가지를 알아보겠습니다.

```
const doggy = {
  name: '멍멍이',
  sound: '멍멍',
  age: 2
};

console.log(Object.entries(doggy));
console.log(Object.keys(doggy));
console.log(Object.values(doggy));
```

The screenshot shows a CodeSandbox environment with a file named `index.js`. The code defines a `doggy` object and logs its entries, keys, and values. The console output shows the result of `Object.entries(doggy)`, which is an array of three arrays, each containing a key and its value.

```
1 const doggy = {
2   name: '멍멍이',
3   sound: '멍멍',
4   age: 2
5 };
6
7 console.log(Object.entries(doggy));
8 console.log(Object.keys(doggy));
9 console.log(Object.values(doggy));
10
```

Console output:

```
▼ [Array(2), Array(2), Array(2)]
  ▼ 0: Array(2)
    0: "name"
    1: "멍멍이"
  ▼ 1: Array(2)
    0: "sound"
    1: "멍멍"
  ▼ 2: Array(2)
    0: "age"
    1: 2
  ▶ ["name", "sound", "age"]
  ▶ ["멍멍이", "멍멍", 2]
```

각 함수의 역할은 다음과 같습니다.

- `Object.entries`: `[[키, 값], [키, 값]]` 형태의 배열로 변환
- `Object.keys`: `[키, 키, 키]` 형태의 배열로 변환
- `Object.values`: `[값, 값, 값]` 형태의 배열로 변환

객체가 지니고 있는 값에 대하여 반복을 하고 싶다면 위 함수들을 사용하셔도 되고, `for...in` 구문을 사용하셔도 됩니다.

```
const doggy = {
  name: '멍멍이',
  sound: '멍멍',
  age: 2
};

for (let key in doggy) {
  console.log(`${key}: ${doggy[key]}`);
}
```

The screenshot shows a CodeSandbox environment with a file named `index.js`. The code defines a `doggy` object and uses a `for...in` loop to log each key-value pair. The console output shows the result of the loop, which is a series of strings in the format `key: value`.

```
1 const doggy = {
2   name: '멍멍이',
3   sound: '멍멍',
4   age: 2
5 };
6
7 for (let key in doggy) {
8   console.log(`${key}: ${doggy[key]}`);
9 }
10
```

Console output:

```
name: 멍멍이
sound: 멍멍
age: 2
```

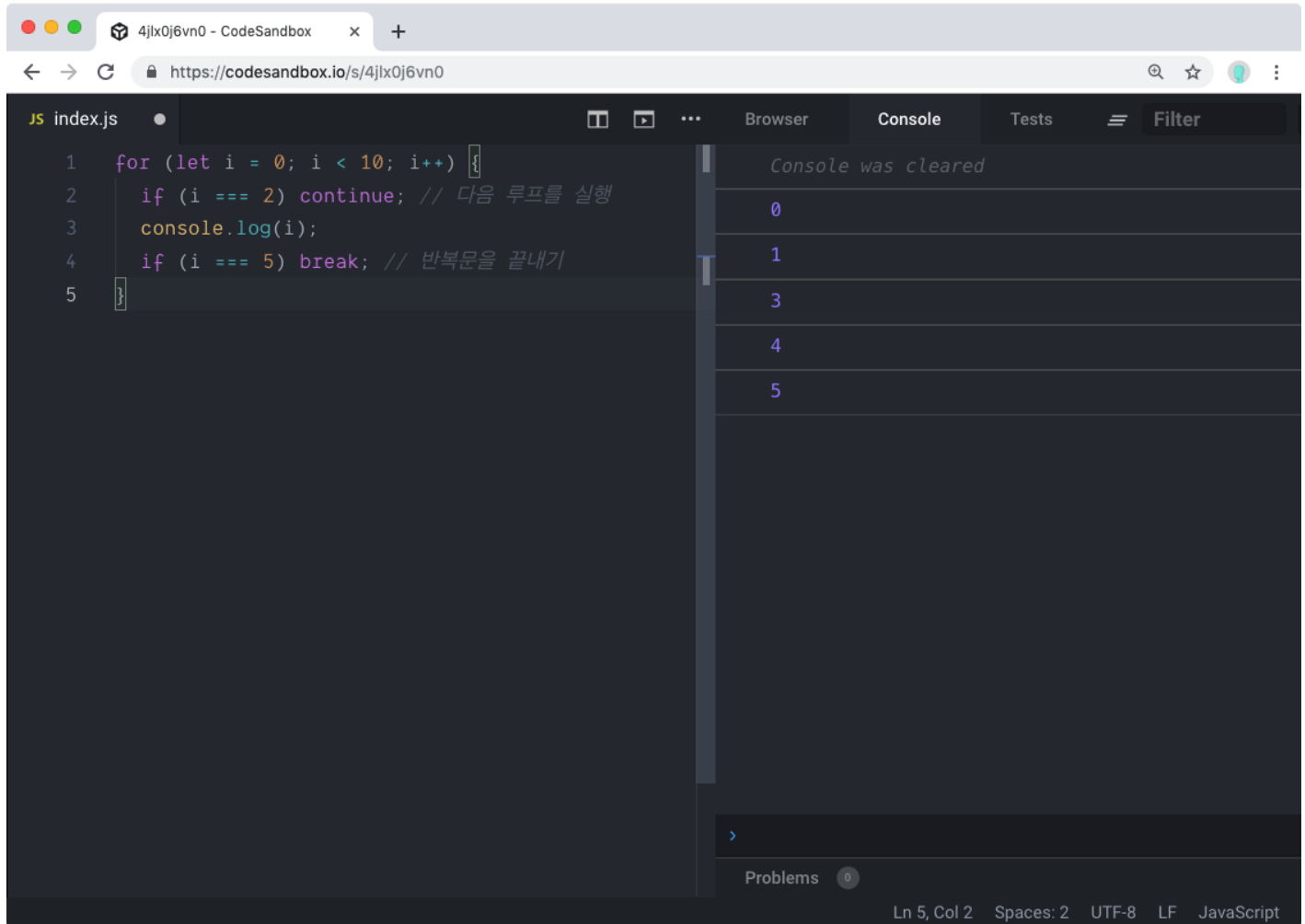
break 와 continue

반복문 안에서는 `break` 와 `continue` 를 통하여 반복문에서 벗어나거나, 그 다음 루프를 돌게끔 할 수 있습니다.

```
for (let i = 0; i < 10; i++) {  
  if (i === 2) continue; // 다음 루프를 실행  
  console.log(i);  
  if (i === 5) break; // 반복문을 끝내기  
}
```

`i` 가 2 일때 `continue` 를 하여 원래 `console.log` 를 해야 하지만 그 코드를 수행하지 않고 바로 3으로 넘어갑니다.

`i` 가 5 일때 `break` 를하여 반복문을 종료시킵니다.



연습

`numbers` 라는 배열을 파라미터로 받아서 총합을 구하는 함수를 만들어봅시다.

```
function sumOf(numbers) {  
  let sum = 0;  
  for (let i = 0; i < numbers.length; i++) {  
    sum += numbers[i];  
  }  
  return sum;  
}  
  
const result = sumOf([1, 2, 3, 4, 5]);  
console.log(result);
```

결과는 15 입니다.

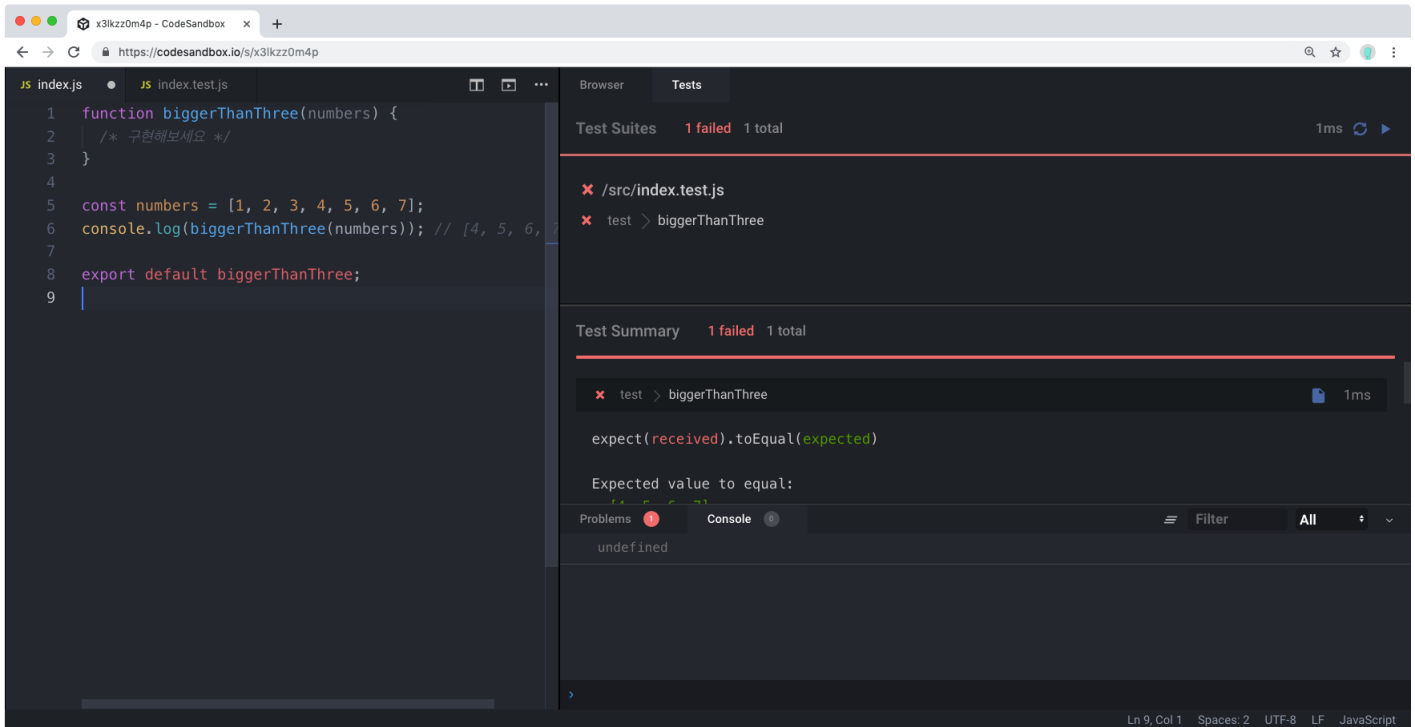
퀴즈

숫자로 이루어진 배열이 주어졌을 때, 해당 숫자 배열안에 들어있는 숫자 중 3보다 큰 숫자로만 이루어진 배열을 새로 만들어서 반환해보세요.

```
function biggerThanThree(numbers) {  
  /* 구현해보세요 */  
}  
  
const numbers = [1, 2, 3, 4, 5, 6, 7];  
console.log(biggerThanThree(numbers)); // [4, 5, 6, 7]
```

다음 링크를 열어서 문제를 풀어보세요.

 [Edit on CodeSandbox](#)



페이지에 들어가면 우측에 Test 탭을 누르면 테스트가 실패했다는 것을 보여줍니다. 해당 함수를 구현하고 나면 다음과 같이 통과가 됩니다.

