javascript 자료형

기본 자료형 (primitive type)

javascript 언어의 기본 자료형은 다음과 같다.

- number
- string
- boolean
- undefined

> 💡 정수(int)형과 실수형을 구분하지 않음에 주목하자.
> undefined 도 자료형임에 주목하자.

복합 자료형 (complex type)

javascript 언어의 복합 자료형은 다음과 같다.

- function
- object

> 💡 javascript 언어에서는 함수(function)가 자료형임에 주목하자.
> javascript 언어에서는 함수가 값이라는 말이다.
> 함수를 변수에 대입할 수 있고, 파라미터로 함수를 전달할 수도 있다.
> 어떤 변수 값의 자료형이 함수(function)일 수 있다.

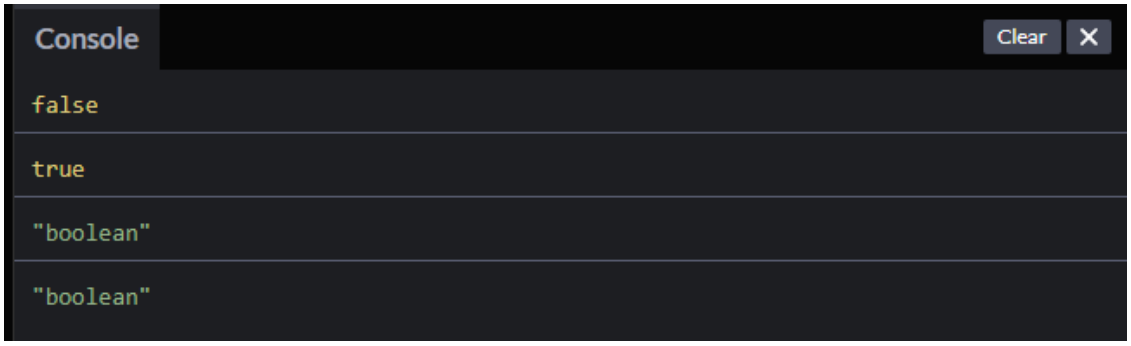
---

boolean 자료형

true, false 값

```
let a = false;
let b = (4 > 3);

console.log(a);
console.log(b);
console.log(typeof a);
console.log(typeof b);
```



false 에 해당하는 값

java 언어에서는. if 문이나 while 문의 조건식에, 반드시 boolean 타입의 조건식만 나올 수 있다.
javascript 언어에서는 그렇지 않다.

javascript 언어에서는 boolean 타입의 아닌 표현식도,
if 문이나 while 문의 조건식에 사용될 수 있다.

조건식에 사용된 값이 boolean 타입이 아닌 경우

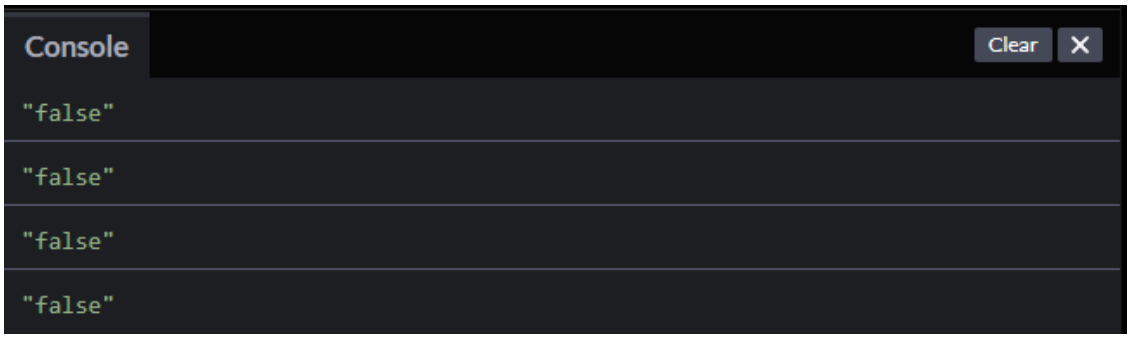
그 값이 NaN, 0, null, undefined, "" 이면, false로 취급된다.
그외 나머지 값은 true로 취급된다.

NaN (Not a Number, 계산 결과가 숫자가 아닐 때 그 값은 NaN 이다)

```
let a = undefined, b = null, c = 0, d = NaN;

console.log(a ? "true" : "false");
```

```
console.log(b ? "true" : "false");
console.log(c ? "true" : "false");
console.log(d ? "true" : "false");
```



조건식 ? 값1 : 값2

조건식이 true 이면, 결과는 값1 이고,
조건식이 false 이면, 결과는 값2 이다.

lazy evaluation

&& || 연산자가 포함된 표현식에서, 실행할 필요가 없는 부분을 실행하지 않는 것을 lazy evaluation 이라고 한다. C, Java, Javascript 등 대부분의 언어는 lazy evalution 한다.

|| 연산자 (or 연산자)
&& 연산자 (and 연산자)

표현식1 || 표현식2

표현식1 값이 true 이면, 결과는 true 이다. 이때 표현식2는 실행되지 않는다.
그래서 위 OR 식의 값은 표현식1의 값이다.

표현식1 값이 false 이면, 결과는 표현식2의 값이다. 이때는 당연히 표현식2가 실행된다.
그래서 위 AND 식의 값은 표현식2의 값이다.

```
let a = 3, b = 0;
```

```
console.log(a || b); // a 값이 true에 해당하는 값이므로, a || b 식의 값은 a의 값 즉 3 이다.
console.log(b || a); // b 값이 false에 해당하는 값이므로, a || b 식의 값은 a의 값 즉 3 이다.
```

표현식1 && 표현식2

표현식1 값이 flase 이면, 결과는 false 이다. 이때 표현식2는 실행되지 않는다.
그래서 위 식의 값은 표현식1의 값이다.

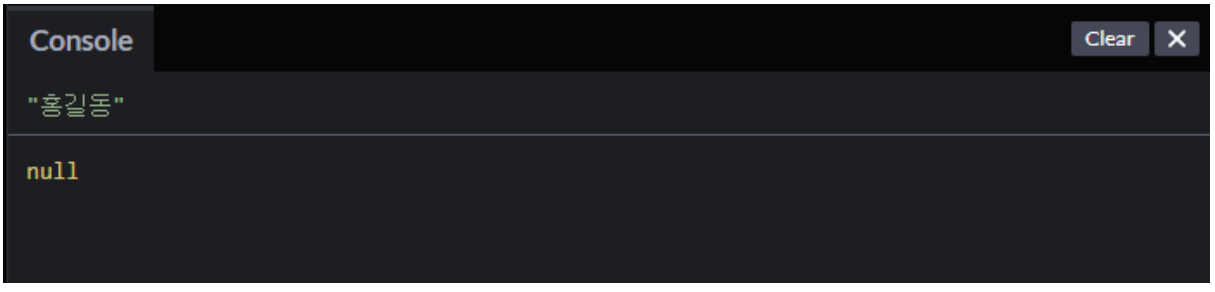
표현식1 값이 true 이면, 결과는 표현식2의 값이다. 이때는 당연히 표현식2가 실행된다.
그래서 위 식의 값은 표현식2의 값이다.

```
let a = 3, b = 0;

console.log(a && b); // a 값이 true에 해당하는 값이므로, a && b 식의 값은 b의 값 즉 0 이다.
console.log(b && a); // b 값이 false에 해당하는 값이므로, a && b 식의 값은 b의 값 즉 0 이다.
```

```
let person1 = { name: "홍길동" }, person2 = null;

console.log(person1 && person1.name);
console.log(person2 && person2.name);
```



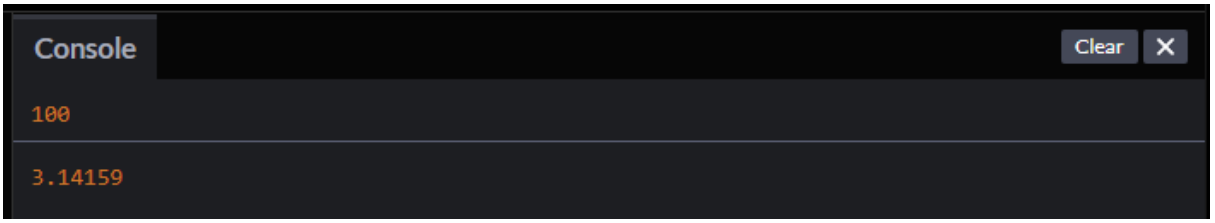
number 자료형

정수 실수 구분이 없다

javascript 언어에서 숫자는, 정수(int)형과 실수형을 구분하지 않고, 그냥 number 타입이다.
숫자 타입 변수의 크기는 64bit 이고, 실수 형태로 저장된다.

```
let a = 100;
console.log(a);

a = 3.14159;
console.log(a);
```



형변환 (string -> number)

string 타입을 number 타입으로 변환하기 위한 함수는 다음과 같다.

- parseInt
- parseFloat
- Number

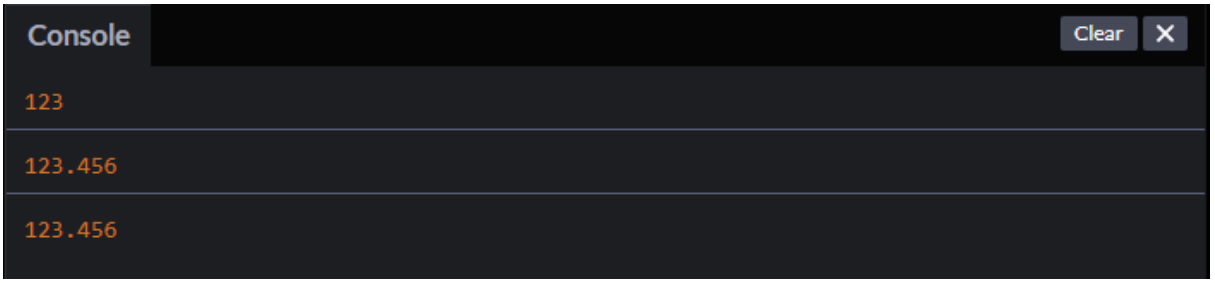
parseInt 함수는 소숫점 아래 값은 버린다.
Number 함수 이름의 첫문자가 대문자임에 주의하자.

```
let s = "123.456";

let a = parseInt(s);
console.log(a);
```

```
let b = parseFloat(s);
console.log(b);

let c = Number(s);
console.log(c);
```



형변환 (number -> string)

number 타입을 string 타입으로 변환하기 위한 함수는 다음과 같다.

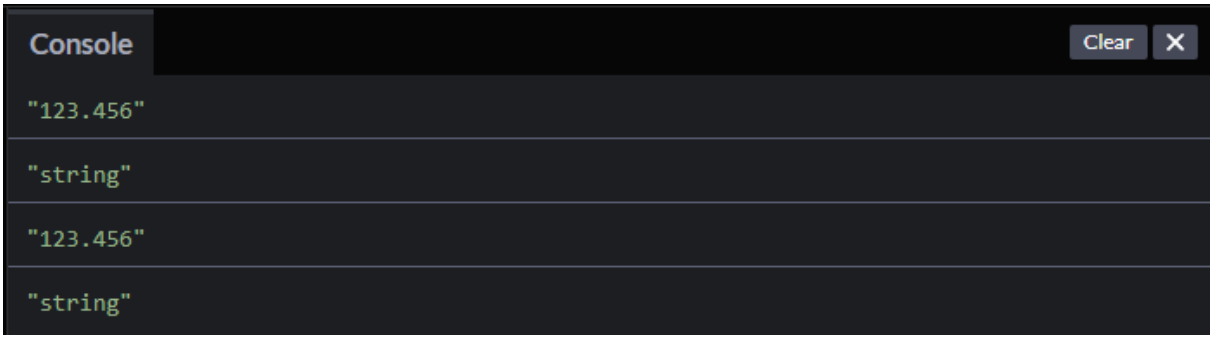
- toString
- String

String 함수 이름의 첫 문자가 대문자임에 주의하자.

```
let a = 123.456;

let s = a.toString();
console.log(s);
console.log(typeof s);

s = String(a);
console.log(s);
console.log(typeof s);
```



Math 클래스

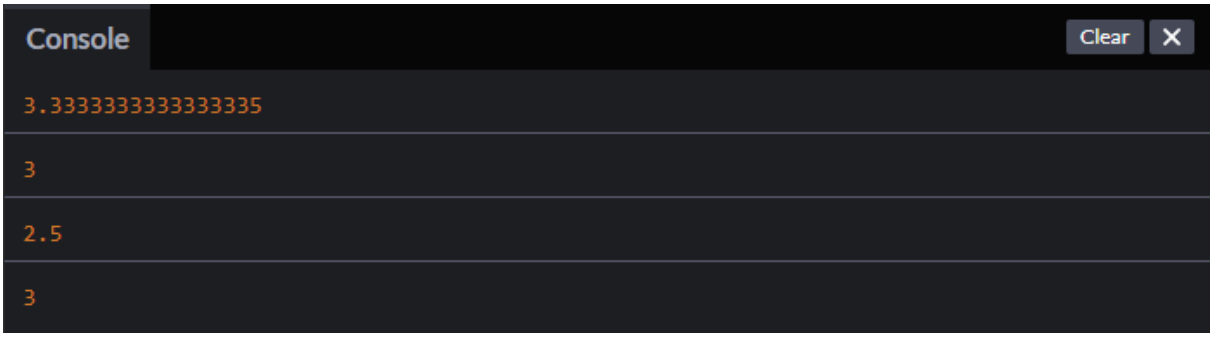
PI - 원주율

```
console.log(Math.PI);
```

round - 반올림 메소드

```
console.log(10 / 3);
console.log(Math.round(10 / 3));

console.log(10 / 4);
console.log(Math.round(10 / 4));
```



pow - 거듭제곱 메소드

```
console.log(Math.pow(2, 3));
console.log(Math.pow(2, 8));
console.log(Math.pow(2, 10));
```

```
8
256
1024
```

sqrt - 제곱근 메소드

```
console.log(Math.sqrt(9));
console.log(Math.sqrt(16));
```

```
3
4
```

abs - 절대값 메소드

```
console.log(Math.abs(-3));
```

```
3
```

floor/ceil - 내림/올림 메소드

```
console.log(Math.floor(4.01));
console.log(Math.floor(4.99));
console.log(Math.ceil(4.01));
console.log(Math.ceil(4.99));
```

```
4
4
5
5
```

내림 : 소수점 아래 값을 버린다.
올림 : 소수점 아래 값이 있으면, 그 값을 버리고 1을 올린다.

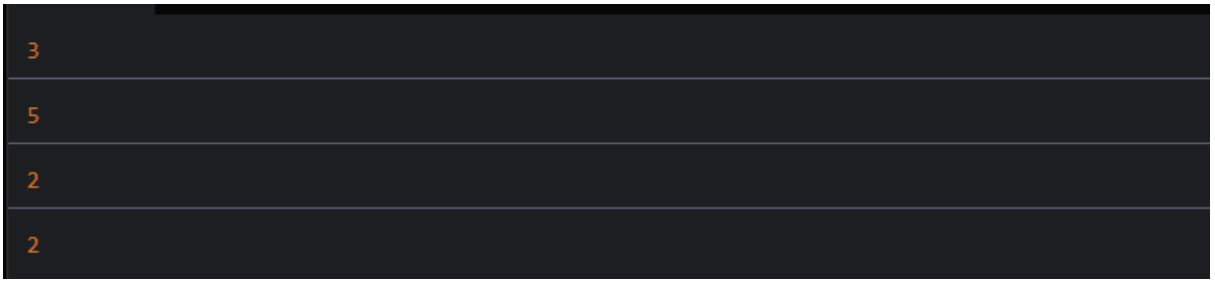
sin, cos, tan - 삼각함수 메소드

```
console.log(Math.sin(Math.PI / 2)); // sin 90도
console.log(Math.cos(Math.PI));     // cos 180도
```

```
1
-1
```

max, min - 최대 최소 메소드

```
console.log(Math.max(2, 3));
console.log(Math.max(2, 3, 5));
console.log(Math.min(2, 3));
console.log(Math.min(2, 3, 5));
```



max 메소드: 파라미터 목록에서 최대값을 리턴한다.
min 메소드: 파라미터 목록에서 최소값을 리턴한다.

random - 난수 메소드

```
for (let i = 0; i < 20; ++i) {
  let a = Math.floor(Math.random() * 6 + 1);
  console.log(a);
}
```

Math.random 메소드는, 0 보다 크거나 같고, 1보다 작은 난수를 리턴한다. (0 <= 리턴값 < 1)

Math.floor(Math.random() * 6 + 1)
이 계산식의 값은 1, 2, 3, 4, 5, 6 중의 하나이다.

```
const MAX = 5, MIN = 2;

for (let i = 0; i < 20; ++i) {
  let a = Math.floor(Math.random() * (MAX - MIN + 1) + MI
N);
  console.log(a);
}
```

Math.random() * (MAX - MIN + 1) + MIN
이 계산식의 값은 MIN 보다 크거나 같고, MAX 보다 작거나 같은 정수들 중의 하나이다.

유효숫자 자릿수

숫자 타입 변수의 크기는 64bit 이고, 실수 형태로 저장되기 때문에, 유효숫자 자릿수 문제가 발생한다.
정수는 15자리까지 정확하다.
15자리가 넘는 정수는 유효숫자 자릿수 부족 때문에 오차가 발생한다.

```
let a;

a = 999999999999999; // 15 자리
console.log(a);

a = 9999999999999999; // 16 자리
console.log(a);

a = 0.9999999999999999; // 16 자리
console.log(a);

a = 0.99999999999999999; // 17 자리
console.log(a);
```

```
999999999999999
10000000000000000
0.9999999999999999
1
```

Infinity

javascript에서 Infinity 키워드는 무한대에 해당하는 값을 의미한다.

```
let a = Infinity;
console.log(a);
console.log(typeof a);

let b = 3 / 0;
```

```
console.log(b);
console.log(typeof b);
```

```
Infinity
"number"
Infinity
"number"
```

3 / 0 계산 결과 값이 Infinity 임에 주목

```
let a = Infinity;
let b = 3 / a;
console.log(b);

let c = 3 + a;
console.log(c);
```

```
0
Infinity
```

3 / 무한대 계산 결과는 0 이다.
3 + 무한대 계산 결과는 무한대이다.

NaN

javascript에서 NaN 키워드는 계산 결과가 숫자가 아님을 의미한다. (Not a Number)

```
let a = 3 * "ABC";
console.log(a);
console.log(typeof a);

let b = Number("abc");
```

```
console.log(b);
console.log(typeof b);
```

```
NaN
"number"
NaN
"number"
```