💡 아래의 코드를 실행하시면 됩니다.

```
var a = '1';

console.log(+a, typeof +a);
console.log(a, typeof a);

a = true;
console.log(+a, typeof +a);
console.log(a, typeof a);

a = false;
console.log(+a, typeof +a);
console.log(a, typeof a);

a = 'Hi';
console.log(+a, typeof +a);
console.log(a, typeof a);
```

```
1 'number'                                  VM1890:2
1 string                                    VM1890:3
1 'number'                                  VM1890:5
true 'boolean'                              VM1890:6
0 'number'                                  VM1890:8
false 'boolean'                             VM1890:9
NaN 'number'                                VM1890:11
Hi string                                   VM1890:12
```

💡 **암묵적 타입 변환** 또는 **타입 강제 변환**에 대해서 알아보시오

암묵적 타입 변환 : 개발자의 의도와는 상관없이 자바스크립트 엔진에 의해 암묵적으로 자동으로 변환되는 것. 타입 강제 변환이라 한다.

암묵적 타입 변환은 변수 값을 재할당해서 변경하는 것이 아닌 자바스크립트 엔진이 표현식을 에러 없이 평가하기 위해 기존 값을 바탕으로 새로운 타입의 값을 만들어 단 한 번 사용하고 버린다.

자신이 작성한 코드에서 암묵적 타입 변환이 발생하는지, 발생한다면 어떤 타입의 어떤 값으로 변환되며, 그리고 타입 변환된 값으로 표현식은 어떻게 평가될 것인지 예측 가능해야 하며, 예측하지 못하면 버그를 생산할 가능성이 높아진다.

💡 아래의 비교가 뭐가 다른지 알아보시오.

```
5 == 5;
5 == '5';
// ================================================
5 === 5;
5 === '5';
```

위 쪽은 값만 동등함을 보지만 아래 쪽은 값과 타입이 일치함을 보기 때문에 다르다.

```
// ================================================
'0' == '';
0 == '';
0 == '0';
// ================================================
false == 'false';
false == '0';
false == null;
false == undefined;
```

```
> '0' == '';
< false
```

```
> 0 == '';
< true
```

```
> 0 == '0';
< true
```

```
> false == 'false';
< false
```

```
> false == '0';
< true
```

```
> false == null;
< false
```

```
> false == undefined;
< false
```

```
NaN === NaN
0 == -0
0 === -0
```

```
> NaN === NaN;
< false
```

```
> 0 == -0;
< true
```

```
> 0 === -0;
< true
```

- 0 === 0 ;
Object.is(-0,0)

NaN === NaN;
Object.is(NaN,NaN);
의 결과가 왜 다를까?

```
> -0 === 0;
< true
> Object.is(-0,0)
< false
```

```
> NaN === NaN
< false
> Object.is(NaN,NaN);
< true
```

부동 소수점 산술의 의미는 IEEE 754에 의해 결정됨.

NaN이 수학 연산에 포함된 경우 결과도 일반적으로 NaN이다. NaN이 관계 비교의 피연산자 중 하나인 경우 결과는 항상 false이다. NaN이 (==,!=,=== 및 !==를 통해) 다른 NaN 값을 포함하여 다른 값과 같지 않은 것으로 비교됩니다.

위에 있는 반환 값을 다 나타내보시오.
예시)

```
typeof 1
```

```
> var num = 7; var str = '끼얏호우'; var bool = true; var unde = undefined; var sym = Symbol('yee');
  var obj = {}; function fun(){console.log(1)}; console.log(typeof num); console.log(typeof str);
  console.log(typeof bool); console.log(typeof unde); console.log(typeof sym); console.log(typeof
  obj); console.log(typeof fun);
  number                                                                    VM969:1
  string                                                                    VM969:1
  boolean                                                                   VM969:1
  undefined                                                                 VM969:1
  symbol                                                                    VM969:1
  object                                                                    VM969:1
  function                                                                  VM969:1
```