2022/5/2

1. String

   - substr()은 더 이상 사용하지 않음

   - 템플릿 리터럴은 IE에서는 지원하지 않음

   - replace()는 기본적으로는 맨 처음 나오는 것만 바꾸지만, 정규식을 이용하면 전체를 바꿀 수 있음 (ex. "hello world".replace(/l/g, "k") -> "hekko workd"))

2. Boolean

- == 는 값만 비교(한 쪽을 자동 형변환), ===는 타입도 비교 (ex. 0 == false -> true, 0 === false -> false)

- != 는 ==의 반대, !==는 ===의 반대

- !!을 앞에 붙이면 그것의 정확한 Boolean 값을 알 수 있음

- null, undefined, ""(공백), 0, false의 Boolean 값은 false, 나머지는 true

- [], {} 도 true인 객체로 취급되는 것에 주의

- NaN === NaN -> false인 이유는 규칙 상 한 쪽이 NaN이면 false를 반환하게 되어 있기 때문 -> Number.isNaN() 사용

- === 비교에서 한 쪽이 undefined이면 무조건 true인 이유는 undefined 값이 어떤 값이 될 지 알 수 없으며, 규칙 상 한 쪽이 undefined이면 true를 반환하게 되어 있기 때문

- isNaN(undefined) -> true

- 그냥 isNaN()은 우선 인자의 타입을 Number로 변환한 후 NaN 인지를 검사하기 때문에 에러가 생길 수 있지만, Number.isNaN()은 인자의 타입이 Number인지와 NaN인지를 모두 검사하기 때문에 Number.isNaN()을 사용하는 게 좋음

3. Object

- Object.prototype.method() 형태의 메소드는 객체에서 직접 사용할 수 있음 (ex. let a = "123"; a.toString();)

- Object.method() 형태의 메소드는 객체 원형과 함께 사용해 줘야 함 (ex. let a = "123"; Number.isNaN(a);)

- 객체 접근 시 객체이름["key1"]["key2"]... 식으로 접근하는 것과 객체이름.key1.key2... 식으로 접근하는 것은 동일한 결과

- key 선언 시 따옴표를 쓰지 않으면 띄어쓰기 했을 때 오류가 생기기도 함