자바스크립트(JavaScript)의 자료형, 연산자

자바스크립트(JavaScript)의 자료형(Date Type)

= 컴퓨터가 처리하는 자료의 형태

- Primitive(기본형)과 Object(객체) 타입이 존재
- 기본적으로 자바스크립트(JavaScript)는 인터프리터가 알아서
  변수의 타입을 파악하고 값을 저장해, 변수의 타입을 따로 쓰지 않고 var 를 씀

EX)

```
var a = 10;
var b = ‘k’;
```

1. 기본형(Primitive Type) : 더이상 나눠질 수 없는 single item

= number, bigInt, stirng, boolean, undefined, null, symbol

1. Number(숫자)

   = 따옴표 없이 표기한 숫자, 정수 및 부동소수점(floating point number)을 나타냄

   - 정수(소수점 X, 표현 방법에 따라 10진수, 8진수, 16진수)
   - 실수(소수점 O, 자바스크립트에서는 정밀한 실수 계산을 못함)
   - 무한대 및 NaN(숫자가 아님) 값을 포함함
   - 관련된 연산은 다양하지만 대표적으로 곱셈`*`, 나눗셈`/`, 덧셈`+`, 뺄셈`-` 이 있음
   - 일반적인 숫자 외에 `Infinity`, `-Infinity`, `NaN`같은 특수 숫자 값(special numeric value)이 포함된다
     - Infinity는 어떤 숫자보다 더 큰 특수 값, 무한대(**∞**)를 의미함
       어느 숫자든 0으로 나누면 무한대를 얻을 수 있음

       EX)

       ```
       alert( 1 / 0 ); // 무한대
       ```

     - `Infinity`를 직접 참조할 수도 있음

       EX)

       ```
       alert( Infinity ); // 무한대
       ```

     - `NaN`은 계산 중에 에러가 발생했다는 것을 나타내주는 값이며, 부정확하거나 정의되지 않은 수학 연산을 사용하면 계산 중에 에러가 발생하는데, 이때 `NaN`이 반환됨

EX)

```
alert( "숫자가 아님" / 2 ); // NaN, 문자열을 숫자로 나누면 오류가 발생
```

- `NaN`은 여간해선 바뀌지 않고, `NaN`에 어떤 추가 연산을 해도 결국 `NaN`이 반환

EX)

```
alert( "숫자가 아님" / 2 + 5 ); // NaN
```

- 연산 과정 어디에선가 `NaN`이 반환 되면, 이는 모든 결과에 영향을 미침
- parseInt('blabla'), Math.sqrt(-1) 등의 함수는 NaN을 반환하게 되며, 42 / 0 처럼 무한대가 나오는 식은 Infinity를 반환함

EX)

```
let n = 123;
n = 12.345;
```

2. BigInt
   = 자주 쓰이지 않음
   - 표준으로 채택된 지 얼마 안된 자료형, 길이에 상관없이 정수를 나타낼 수 있음
   - 끝에 'n'이 붙으면 BigInt형 자료

EX)

```
const bigInt = 1234567890123456789012345678901234567890n;
// 끝에 n이 붙어서 이것은 BigInt형 자료
```

   - 숫자의 표현 범위 : `-2**53` ~ `2**53`
3. String(문자형)
   = 작은 따옴표 ' 나 큰 따옴표 " 로 묶어 나타 냄(기본적인 따옴표로 ' 와 "의 차이를 두지 않음)

EX)

```
let str = "Hello";
let str2 = 'Single quotes are ok too';
let phrase = `can embed another ${str}`;
```

   - 따옴표 3가지 종류
     1. 큰따옴표: `"Hello"`
     2. 작은따옴표: `'Hello'`
     3. 역 따옴표(백틱, backtick): `` `Hello` ``

- char형이 존재하지 않고 숫자도 따옴표로 묶으면 문자형이 됨

- 따옴표 안에 따옴표를 넣어야 할 경우 ‘ “ “ ‘ 또는 “ ‘ ‘ “ 처럼 사용
- 이스케이프(Escape) 문자를 사용 가능
- 문자열끼리 이어 붙이거나 숫자를 붙일 때는 +연산자를 사용

EX) 역 따옴표로 변수나 표현식을 감싼 후 `${…}` 안에 넣어주면,
아래와 같이 원하는 변수나 표현식을 문자열 중간에 손쉽게 넣을 수 있음

```
let name = "John";

// 변수를 문자열 중간에 삽입
alert( `Hello, ${name}!` ); // Hello, John!

// 표현식을 문자열 중간에 삽입
alert( `the result is ${1 + 2}` ); // the result is 3
```

`${…}` 안에는 `name` 같은 변수나 `1 + 2` 같은 수학 관련 표현식을 넣을 수 있고,
이렇게 문자열 중간에 들어간 변수나 표현식은 평가가 끝난 후 문자열의 일부가 됨
큰 따옴표(” ”)나 작은 따옴표(’ ‘)를 사용하면 중간에 표현식을 넣을 수 없다는 점에
주의해야하며, 이 방법은 역 따옴표를 써야만 가능

```
alert( "the result is ${1 + 2}" );
// the result is ${1 + 2} (큰따옴표는 확장 기능을 지원하지 않습니다)
```

- 문자열 인덱싱도 가능

4. Boolean(논리형)
   = 참(true), 거짓(false) 두 값만 가지고 있는 유형
   - 긍정(yes), 부정(no)를 나타내는 값을 저장할 때 사용
     긍정(yes) = 참(true), 부정(no) = 거짓(false)를 의미
   - 조건을 확인할 때 많이 사용
   - 거짓(false) = 0, null, undefined, NaN, ''(빈문자열)
   - 참(true) = any other value

   EX)

   ```
   let nameFieldChecked = true;
   // 네, name field가 확인되었습니다(checked)

   let ageFieldChecked = false;
   // 아니요, age field를 확인하지 않았습니다(not checked)
   ```

   비교 결과를 저장할 때도 사용 됨

   ```
   let isGreater = 4 > 1;

   alert( isGreater ); // true (비교 결과: "yes")
   ```

5. null
   = 값이 유효하지 않을 때의 유형(빈 값을 할당)
   - 어느 자료형에도 속하지 않는 값
   - `null` 값만 포함하는 별도의 자료형을 만

EX)

```
let age = null;
```

자바스크립트의 `null`은 자바스크립트 이외 언어의 `null`과 성격이 다름
다른 언어에선 `null`을 존재하지 않는 객체에 대한 참조나
널 포인터(null pointer)를 나타낼 때 사용함
하지만 자바스크립트에선 `null`을 존재하지 않는(nothing)값,
비어 있는(empty)값, 알 수 없는(unknown)값을 나타내는 데 사용
`let age = null;`은 `나이(age)`를 알 수 없거나 그 값이 비어있음을 보여줌

6. undefined
   = 자료형을 지정하지 않았을 때의 상태

   - `null`값처럼 자신만의 자료형을 형성
   - 변수가 undefined는 ***처음부터 변수에 값이 할당되지 않았다***는 의미
     즉, 값이 할당되지 않는 상태를 나타낼 때 사용함

EX) 변수는 선언했지만, 값을 할당하지 않았다면
해당 변수에 `undefined`가 자동으로 할당됨

```
let age;

alert(age); // 'undefined'가 출력
```

개발자가 변수에 `undefined`를 명시적으로 할당하는 것도 가능

```
let age = 100;

// 값을 undefined로 바꿈
age = undefined;

alert(age); // "undefined"
```

하지만 이렇게 `undefined`를 직접 할당하는 걸 권장하진 않음
변수가 비어있거나, 알 수 없는 상태라는 걸 나타내려면 `null`을 사용할것

`undefined`는 값이 할당되지 않은 변수의 초기값을 위해 예약어로 남겨둘것

7. symbol
   = create unique identifiers for objects

   - map 등 식별자가 필요하거나 동시다발적으로 콘크리트하게 일어날 수 있는
     코드에서 우선순위를 주고싶을때 정말 고유한 식별자가 됨

2. 복합형

1. array(배열)
   = 하나의 변수에 여러 값을 저장

   - 배열의 인덱스(index)는 0부터 시작
   - 배열에 있는 값을 가져오려면 배열에 이름과 대괄호 **[ ]** 안에 인덱스(index)사용
   - .length로 길이를 구할 수 있음

EX)

```
let season = ["봄", "여름", "가을", "겨울"];

season[0] //봄
```

2. object(객체)
   = 함수와 속성이 함께 포함된 유형
   - 여러 자료를 중괄호 **{ }**로 묶은 것

EX)

```
let info = {
  firstName: "Hyun",
  lastName: "Lim",
  age: 25,
  address: "Seoul"
}
```

자바스크립트(JavaScript) 자료형의 특징

- 느슨한 자료형 체크(weak datatype check) → typescript로 보완된 것이 나옴
- 자바스크립트(JavaScript)는 미리 변수의 자료형을 지정하지 않음
- 변수를 지정하고 원하는 값을 할당만 하면 됨

자바스크립트(JavaScript)의 연산자

1. 사칙 연산자 = `+`, `-`, `*`, `/`
2. 나머지 연산자 = `%`
3. 증감 연산자 = 증가 연산자(`++`), 감소 연산자(`--`)
   증가 연산자는 1만큼 증가, 감소 연산자는 1만큼 감소한다
   - 전위(`++n`) 증가 연산자 : ;이 끝나기 전에 이미 증가가 되어 있음
   - 후위(`--n`) 증가 연산자 : ;이 끝난 후 증가가 됨

```
let a = 10;

console.log(a++); // 10
console.log(a); // 11
console.log(++a); // 12
```

4. 할당 연산자(복합대입 연산자) = `+=`, `-=`, `*=`, `/=`, `%=`

EX)

```
a%=b → a=a%b
```

- 변수에 값을 할당하는 연산자
- 사칙 연산자와 조합해서 사용 할 수 있음

5. 연결 연산자
    - 문자열과 문자열을 연결
    - + 기호 사용
6. 형 변환
    - 숫자형과 문자형을 더 하면 숫자를 문자열로 인식함
    - 곱하기나 나누기, 나머지 연산에는 문자형 자료를 모두 숫자로 자동 인식함

    연산자와 피연산자
    CurrenYear - BirthYear + 1
    CurrenYear, BirthYear, 1은 연산 대상이 되기 때문에 ***피연산자***라고 부름
    피연산자를 제외한 더하기, 빼기 같은 것을 ***연산자***라고 부름