# Ch6 데이터 타입

**데이터 타입**은 값의 종류를 말한다.

> **원시 타입** : 숫자, 문자열, 불리언, undefined, null, 심벌
>
> **객체 타입** : 객체, 함수, 배열 등

### 숫자 타입

- 배정밀도 64비트 부동소수점 형식을 따른다.
- 모든 수를 실수로 처리하며, 정수만 표현하기 위한 타입이 따로 존재하지 않는다.
- 정수, 실수, 2진수, 8진수, 16진수 리터럴은 모두 메모리에 배정밀도 64비트 부동소수점 형식의 2진수로 저장된다.
  - 참조하면 모두 10진수로 해석된다.

```
var binary = 0b01000001; // 2진수
var octal = 0o101;       // 8진수
var hex = 0x41;          // 16진수

// 표기법만 다를 뿐 모두 같은 값이다.
console.log(binary); // 65
console.log(octal);  // 65
console.log(hex);    // 65
console.log(binary === octal); // true
console.log(octal === hex);    // true
```

- 정수로 표시된다 해도 사실은 실수이고, 정수로 표시되는 수끼리 나누더라도 실수가 나올 수 있다.

```
// 숫자 타입은 모두 실수로 처리된다.
console.log(1===1.0); //true
```

- Infinity: 양의 무한대
- -Infinity: 음의 무한대
- NaN: 산술 연산 불가 (not-a-number)

✔️ 자바스크립트는 대소문자를 구별하므로 주의

### 문자열 타입

문자열은 0개 이상의 16비트 유니코드 문자 (UTF-16) 의 집합으로 전 세계 대부분의 문자를 표현할 수 있다. (‘’, “”, ``으로 텍스트를 감싸서 표현함)

- 따옴표로 감싸서 키워드나 식별자 같은 토큰과 구분함
- 따옴표로 감싸지 않으면 공백 문자를 포함시킬 수 없음
- 자바스크립트의 문자열은 원시 타입이며, 변경 불가능한 값임

### 템플릿 리터럴

**템플릿 리터럴**은 멀티라인 문자열, 표현식 삽입, 태그드 템플릿 등 편리한 문자열 처리 기능을 제공하며 런타임에 일반 문자열로 변환되어 처리된다.

→ `` 을 사용해 표현한다.

#### 멀티라인 문자열

- 일반 문자열 내에서 줄바꿈 등 공백을 표현하려면 백슬래시(\) 로 시작하는 escape sequence를 사용해야 한다.
- 템플릿 리터럴은 **escape sequence를 사용하지 않고도 줄바꿈이 허용되며 모든 공백도 있는 그대로 적용**된다.

```
var template = `<ul>
  <li><a href="#">Home</a></li>
</ul>`;

console.log(template);
/*
<ul>
  <li><a href="#">Home</a></li>
</ul>
*/
```

#### 표현식 삽입

문자열은 문자열 연산자 + 를 사용해 연결할 수 있다.

템플릿 리터럴 내에서는 표현식 삽입을 통해 문자열을 삽입할 수 있고 표현식 삽입을 하려면 `${ }`으로 표현식을 감싸야 한다.

이때 표현식의 평가 결과가 문자열이 아니더라도 문자열로 타입이 강제로 변환되어 삽입된다.

```
var first = 'Ung-mo';
var last = 'Lee';

// ES6: 표현식 삽입
console.log(`My name is ${first} ${last}.`); // My name is Ung-mo Lee.
```

### 불리언 타입

**불리언 타입**의 값은 논리적 참, 거짓을 나타내는 **true**와 **false** 뿐이다.

### undefined 타입

- **undefined** 타입의 값은 undefined가 유일하다.
- 변수를 참조했을 때 undefined 가 반환된다면 참조한 변수가 선언 이후 값이 할당된 적이 없는, 즉 **초기화되지 않은 변수** 라고 할 수 있다.
- 변수에 값이 없다는 것을 명시하고 싶을 때는 **null** 을 할당한다.

### null 타입

- **null** 타입의 값은 null이 유일하다.
- 변수에 값이 없다는 것을 의도적으로 명시할 때 사용
- 변수에 null 을 할당하는 것은 변수가 이전에 참조하던 값을 더 이상 참조하지 않겠다는 의미
- 이전에 할당되어 있던 값에 대한 참조를 명시적으로 제거하고 자바스크립트 엔진은 가비지 콜렉션를 실행할 것이다.
- 함수가 유효한 값을 반환할 수 없는 경우 명시적으로 null 을 반환하기도 한다.

```
var foo = 'Lee';

// 이전에 할당되어 있던 값에 대한 참조를 제거. foo 변수는 더 이상 'Lee'를 참조하지 않는다.
// 유용해 보이지는 않는다. 변수의 스코프를 좁게 만들어 변수 자체를 재빨리 소멸시키는 편이 낫다.
foo = null;
```

## 심벌 타입

**심벌** 타입은 변경 불가능한 원시 타입의 값이다.

주로 이름이 충돌할 위험이 없는 객체의 유일한 프로퍼티 키를 만들기 위해 사용한다.

## 데이터 타입의 필요성

- 값을 저장할 때 확보해야 하는 메모리 공간의 크기를 결정하기 위해
- 값을 참조할 때 한 번에 읽어 들여야 할 메모리 공간의 크기를 결정하기 위해
- 메모리에서 읽어 들인 2진수를 어떻게 해석해야할지 결정하기 위해

## 동적 타이핑

- 자바스크립트는 변수를 선언할 때 타입을 선언하지 않고 `var`, `let`, `const` 키워드를 사용해 변수를 선언한다.
- 자바스크립트 변수에는 어떤 데이터 타입의 값이라도 자유롭게 할당할 수 있으므로 **값을 할당하는 시점에 변수의 타입이 동적으로 결정된다.**
- **변수는 선언이 아닌 할당에 의해 타입이 결정된다. 재할당에 의해 타입은 언제든지 변할 수 있다. → 동적 타이핑**

✔️ 동적 타입 언어는 유연성은 높지만 신뢰성은 떨어지기 때문에 변수를 사용할 때 주의할 점이 있다.

- 변수의 무분별한 남발은 금물
- 변수의 스코프는 최대한 좁게 만들어 변수의 부작용을 억제
- 전역 변수는 최대한 사용하지 않도록 함
- 변수보다는 상수를 사용해 값의 변경을 억제함
- 변수 이름은 변수의 목적이나 의미를 파악할 수 있도록 네이밍함