Type Conversion (형 변환)

☑ Type Conversion

function와 operator에 전달되는 값은 대부분 적절한 type으로 자동 변환된다. 이런 과정을 "type conversion" 이라고 한다.

alert가 전달받은 값의 자료형과 관계없이 이를 string로 자동 변환하여 보여주는 것이나, Mathematical 관련 operator가 전달받은 값을 숫자로 변환하는 경우가 type conversion의 대표적인 예시이다.

이외에, 전달받은 값을 의도를 갖고 원하는 type으로 변환(explicitly convert)해 주는 경우도 type conversion이라고 할 수 있다.

여기에서는 primitive conversion의 형변환에 대해서만 다룬다.

❖ String Conversion

string으로의 type conversion은 string의 값이 필요할 때 일어난다.

alert method는 parameter로 string을 받기 때문에, alert(value)에서 value는 string이어야 한다. 만약, 다른 형의 값을 전달받으면 이 값은 string으로 자동 변환된다.

String(value) function를 호출해 전달받은 값을 string로 변환 할 수도 있다.

```
let value = true;
alert(typeof value);      // boolean

value = String(value);    // 변수 value엔 string "true"가 저장된다.
alert(typeof value);      // string
```

false는 string "false"로, null은 string "null"로 변환되는 것과 같이, string으로의 변환은 대부분 예측 가능한 방식으로 일어난다.

❖ Numeric Conversion

numeric으로의 변환은 수학과 관련된 function와 expression에서 자동으로 일어난다.
numeric이 아닌 값에 나누기 /를 적용한 경우와 같다.

```
alert( "6" / "2" ); // 3, string이 numeric으로 자동변환된 후 연산이 수행된다.
```

Number(value) function를 사용하면 주어진 값(value)을 numeric으로 명시해서 변환할 수 있다.

```
let str = "123";
alert(typeof str);       // string

let num = Number(str);   // string "123"이 숫자 123으로 변환된다.

alert(typeof num);       // number
```

numeric 값를 사용해 무언가를 하려고 하는데 그 값을 문자 기반 폼(form)을 통해 입력받는 경우엔, 이런 명시적 type conversion이 필수이다.

한편, 숫자 이외의 글자가 들어가 있는 string을 numeric으로 변환하려고 하면, 그 결과는 NaN이 된다.

예시를 살펴보자.

```
let age = Number("임의의 string 123");

alert(age); // NaN, type conversion이 실패한다.
```

아래는 numeric으로 변환 시 적용되는 규칙이다.

전달받은 값	type conversion 후
undefined	NaN
null	0
true / false	1 / 0
string	시작과 끝의 공백이 제거된다. 나머지 string이 비어 있으면 결과는 0 이다. 그렇지 않으면 number는 string에서 "read" 이다. error는 NaN을 제공한다.

[예시]

```
alert( Number("   123   ") );     // 123
alert( Number("123z") );          // NaN ("z"를 숫자로 변환하는 데 실패함)
alert( Number(true) );            // 1
alert( Number(false) );           // 0
```

null과 undefined은 numeric 으로 변환 시 결과가 다르다는 점에 유의한다. null은 0이 되고, undefined는 NaN이 된다. 대부분의 수학 연산은 type conversion을 수반한다.

❖ Boolean Conversion

boolean으로의 변환은 아주 간단하다.
이 type conversion은 logical operation을 수행할 때 발생한다. Boolean(value)를 호출하면 명시적으로 불리언으로의 type conversion을 수행할 수 있다.

boolean으로 변환 시 적용되는 규칙은 다음과 같다.

☞ 숫자 0, empty string, null, undefined, NaN과 같이 직관적으로도 "비어있다고" 느껴지는 값들은 false가 된다.
☞ 그 외의 값은 true로 변환된다.

[예시]

```
alert( Boolean(1) );              // 숫자 1(true)
alert( Boolean(0) );              // 숫자 0(false)

alert( Boolean("hello") );        // string(true)
alert( Boolean("") );             // 빈 string(false)
```

⊙ 주의: string "0"은 true 이다.

PHP 등의 일부 언어에선 string "0"을 false로 취급한다. 그러나 JavaScript에선 비어 있지 않은 string은 언제나 true 이다.

```
alert( Boolean("0") ); // true
alert( Boolean(" ") ); // 공백이 있는 string도 비어있지 않은 string이기 때문에 true로 변환된다.
```

요약하면 string, number, boolean으로의 type conversion은 자주 일어나는 type conversion 이다.

☞ String Conversion은 무언가를 출력할 때 주로 일어난다. String(value)을 사용하면 string으로 명시적 변환이 가능하다. primitive value를 string으로 변환할 땐, 대부분 그 결과를 예상할 수 있을 정도로 명시적인 방식으로 일어난다.

☞ Numeric Conversion은 수학 관련 연산시 주로 일어난다. Number(value)로도 type conversion을 할 수 있다.

Numeric으로의 변환은 다음 규칙을 따른다.

전달받은 값	type conversion 후
undefined	NaN
null	0
true / false	1 / 0
string	전달받은 string을 "그대로" 읽되, 처음과 끝의 공백을 무시합니다. string이 비어있다면 0이 되고, 오류 발생 시 NaN이 됩니다.

☞ Boolean Conversion은 논리 연산 시 발생한다. Boolean(value)으로도 변환할 수 있다.

boolean으로의 type conversion은 다음 규칙을 따른다.

전달받은 값	type conversion 후
0, null, undefined, NaN, ""	false
그 외의 값	true

type conversion 시 적용되는 규칙 대부분은 이해하고 기억하기 쉬운 편에 속한다. 다만 아래는 예외적인 경우이기 때문에 실수를 방지하기 위해 따로 기억해 두도록 한다.

☞ number으로 변환 시 undefined는 0이 아니라 NaN이 된다.
☞ string "0"과 " "같은 공백은 boolean으로 변환 시 true가 된다.