Ch9 타입 변환과 단축 평가

타입 변환이란?

개발자가 의도적으로 값의 타입을 변환하는 것을 **명시적 타입 변환** 또는 **타입 캐스팅**이라 한다.

개발자의 의도와 상관없이 자바스크립트 엔진에 의해 암묵적으로 타입 변환되기도 하는데 이를 **암묵적 타입 변**환 또는 **타입 강제 변환**이라 한다.

암묵적 타입 변환

암묵적 타입 변환이 발생하면 문자열, 숫자, boolean 과 같은 원시 타입 중 하나로 타입을 자동 변환한다.

1. **문자열 타입으로 변환**

   '+'연산자를 사용하면 문자열 연결 연산자로 동작한다.

```
// 숫자 타입
-0 + ‘’        // -> “0”
-1 + ‘’        // -> “-1”
NaN + ‘’       // -> “NaN”
-Infinity + ‘’ // -> “-Infinity”

// 불리언 타입
true + ‘’  // -> “true”

// null 타입
null + ‘’ // -> “null”

// undefined 타입
undefined + ‘’ // -> “undefined”

// 심벌 타입
(Symbol()) + ‘’ // -> TypeError: Cannot convert a Symbol value to a string

// 객체 타입
({}) + ‘’           // -> “[object Object]”
Math + ‘’           // -> “[object Math]”
[] + ‘’             // -> “”
[10, 20] + ‘’       // -> “10,20”
(function(){}) + ‘’ // -> “function(){}”
Array + ‘’          // -> “function Array() { [native code] }”
```

2. **숫자 타입으로 변환**

```
// 문자열 타입
+‘’       // -> 0
+’0’      // -> 0
+’string’ // -> NaN

// 불리언 타입
+false    // -> 0

// null 타입
+null     // -> 0
```

```
// undefined 타입
+undefined // -> NaN

// 심벌 타입
+Symbol() // -> TypeError: Cannot convert a Symbol value to a number

// 객체 타입
+{}             // -> NaN
+[]             // -> 0
+[10, 20]       // -> NaN
+(function(){}) // -> NaN
```

3. **불리언 타입으로 변환**

   조건식의 평가 결과를 boolean 타입으로 암묵적 타입 변환한다.

```
if (‘’)    console.log(‘1’);
if (true)  console.log(‘2’);
if (0)     console.log(‘3’);
if (‘str’) console.log(‘4’);
if (null)  console.log(‘5’);

// 2 4
```

자바스크립트 엔진은 boolean 타입이 아닌 값을 **truthy 값 (참으로 평가되는 값) 또는 falsy 값 (거짓으로 평가되는 값) 으로 구분**한다.

falsy 값에는 `false`, `undefined`, `null`, `0`, `-0`, `NaN`, `''(빈 문자열)` 이 있다

명시적 타입 변환

문자열 타입으로 변환

1. String 생성자 함수를 new 연산자 없이 호출하는 방법
2. Object.prototype.toString 메서드를 사용하는 방법
3. 문자열 연결 연산자를 이용하는 방법

숫자 타입으로 변환

1. Number 생성자 함수를 new 연산자 없이 호출하는 방법
2. parseInt, parseFloat 함수 사용(문자열만 사용가능)
3. + 단항 산술 연산자를 이용하는 방법
4. + 산술 연산자를 이용하는 방법

불리언 타입으로 변환

1. Boolean 생성자 함수를 new 연산자 없이 호출하는 방법
2. ! 부정 논리 연산자를 두 번 사용하는 방법

단축 평가

논리 연산자를 사용한 단축 평가

- 논리합 (||) 또는 논리곱(&&) 연산자 표현식의 평가 결과는 boolean 값이 아닐 수도 있다.

- 논리합, 논리곱 연산자 표현식은 언제나 2개의 피연산자 중 어느 한쪽으로 평가된다.

논리곱 연산자와 논리합 연산자는 논리 연산의 결과를 결정하는 피연산자를 타입 변환하지 않고 그대로 반환하고 이를 **단축 평가**라 한다.

단축 평가는 표현식을 평가하는 도중에 평가 결과가 확정된 경우 나머지 평가 과정을 생략하는 것이다.

단축 평가 표현식	평가 결과
true || anything	true
false || anything	anything
true && anything	anything
false && anything	false

truthy 값일 때 무언가를 해야 한다면 논리곱 (&&) 연산자 표현식이 if 문을 대체할 수 있다.

→ falsy 값은 논리합 (||)

→ 삼항 조건 연산자는 if ... else 문을 대체

```
var done = true;
var message = ‘’;

// 주어진 조건이 true일 때
if (done) message = ‘완료’;

// if 문은 단축 평가로 대체 가능하다.
// done이 true라면 message에 ‘완료’를 할당
message = done && ‘완료’;
console.log(message); // 완료
```

객체를 가리키기를 기대하는 변수가 null 또는 undefined 가 아닌지 확인하고 프로퍼티를 참조할 때

- 객체는 key, value 로 구성된 property 의 집합이다.
- 변수의 값이 객체가 아니라 null 또는 undefined 인 경우 객체의 프로퍼티를 참조하면 TypeError 가 발생하는데 이때 단축 평가를 사용하면 에러가 발생하지 않는다.

```
var elem = null;
var value = elem.value; // TypeError: Cannot read property ‘value’ of null

//단축 평가를 사용하면

var elem = null;
// elem이 null이나 undefined와 같은 Falsy 값이면 elem으로 평가되고
// elem이 Truthy 값이면 elem.value로 평가된다.
var value = elem && elem.value; // -> null
```

함수 매개변수에 기본값을 설정할 때

함수를 호출할 때 인수를 전달하지 않으면 매개변수에 undefined 가 할당된다. 이 때 단축 평가를 사용해 매개변수의 기본값을 설정하면 undefined 로 인해 발생할 수 있는 에러를 막을 수 있다.

옵셔널 체이닝 연산자

ES11 에서 도입된 optional chaining 연산자 **?.** 는 좌항의 피연산자가 **null 또는 undefined 인 경우**, undefined 를 반환하고 그렇지 않으면 우항의 프로퍼티 참조를 이어간다.

→ 이전에는 논리 연산자 &&를 사용했음

```
var str = '';

// 문자열의 길이(length)를 참조한다. 이때 좌항 피연산자가 false로 평가되는 Falsy 값이라도
// null 또는 undefined가 아니면 우항의 프로퍼티 참조를 이어간다.
var length = str?.length;
console.log(length); // 0
```

null 병합 연산자

ES11 에서 도입된 null 병합 연산자 **??** 는 좌항의 피연산자가 null 또는 undefined 인 경우 우항의 피연산자를 반환하고, 그렇지 않으면 좌항의 피연산자를 반환한다.

→ 이전에는 논리 연산자 ||를 사용했음