Chap7 연산자

▼ 피연산자

연산의 대상. 값으로 평가되는 표현

▼ 부수효과

피연산자의 값을 변경한다. 예상하지 못한 결과가 일어난다.

7-1 산술 연산자

- 피연산자를 대상으로 수학적 계산을 수행해 새로운 숫자 값을 만든다. 산술이 불가할 경우 NaN을 반환한다.
- 종류

  ▼ 이항 산술 연산자

  - 2개의 피연산자를 산술 연산하여 숫자 값을 만든다.
  - 모든 이항 연산자는 부수효과 없다.

단항 산술 연산자	의미	부수 효과
++	증가	○
--	감소	○
+	어떠한 효과도 없다. 음수를 양수로 반전하지도 않는다.	×
-	양수를 음수로, 음수를 양수로 반전한 값을 반환한다.	×

표 7-2 단항 산술 연산자

▼ 단항 산술 연산자

- 1개의 피연산자를 산술 연산하여 숫자 값을 만든다.

단항 산술 연산자	의미	부수 효과
++	증가	○
--	감소	○
+	어떠한 효과도 없다. 음수를 양수로 반전하지도 않는다.	×
-	양수를 음수로, 음수를 양수로 반전한 값을 반환한다.	×

표 7-2 단항 산술 연산자

▼ 증가 / 감소 연산자

- 부수효과 있음
- 피연산자 앞에 위치한 전위 증가/감소 연산자는 먼저 피연산자의 값을 증가/감소시킨 후, 다른 연산을 수행
- 피연산자 뒤에 위치한 후위 증가/감소 연산자는 다른 연산을 수행한 후, 피연산자의 값을 증가/감소시킨다.

```
var x=5,result;
result=x++;
console.log(result, x); //5,6

result=++x;
console.log(result, x); //7,7

result=x--;
console.log(result, x); //7,6
```

```
result=--x;
console.log(result, x); //5,5
```

▼ +단항 연산자

- 부수 효과 없음
- 숫자 타입이 아닌 피연산자에 사용하면 숫자 타입으로 변환해준다. (타입 변환할 때 쓰임) → 암묵적 타입 변환
- 피연산자 중 하나 이상이 문자열인 경우 문자열 연결 연산자로 동작한다.

▼ -단항 연산자

- 피연산자의 부호를 반전 한 값을 반환한다.
- 숫자 타입이 아닌 피연산자에 사용하면 숫자 타입으로 변환해준다. (타입 변환할 때 쓰임) → 암묵적 타입 변환
- 부수 효과 없음

▼ 문자열 연결 연산자

- +연산자 사용
- 피연산자 중 하나 이상이 문자열인 경우 문자열 연결 연산자로 동작한다.

```
'1'+2; //'12' 암묵적 타입 변환
```

암묵적으로 타입이 문자열로 반환된다.

7-2 할당 연산자

할당 연산자	예	동일 표현	부수 효과
=	x = 5	x = 5	○
+=	x += 5	x = x + 5	○
-=	x -= 5	x = x - 5	○
*=	x *= 5	x = x * 5	○
/=	x /= 5	x = x / 5	○
%=	x %= 5	x = x % 5	○

할당 연산자

▼ 할당문은 표현식인 문일까 표현식이 아닌 문일까?

```
var x;
console.log(x=10); //10, 할당문은 값으로 평가됨
```

위 예제와 같이 할당문은 값으로 평가되기 때문에 표현식인 문이다.

7-3 비교 연산자

- 좌항과 우항의 피연산자를 비교한 다음 그 결과를 불리언 값으로 반환한다.

▼ **동등/일치 비교 연산자**

- 좌항과 우항의 피연산자가 같은 값으로 평가되는지 비교해 불리언 값으로 반환한다.

비교 연산자	의미	사례	설명	부수 효과
==	동등 비교	x == y	x와 y의 값이 같음	X
===	일치 비교	x === y	x와 y의 값과 타입이 같음	X
!=	부동등 비교	x != y	x와 y의 값이 다름	X
!==	불일치 비교	x !== y	x와 y의 값과 타입이 다름	X

- **동등 vs 일치**
  - 동등 비교 : 암묵적 타입을 일치시킨 후 같은 값인 지 비교. 즉 데이터의 타입을 신경쓰지 않고 값만 같으면 true를 반환한다.

-일치 비교 : 타입도 같고 값도 같은 경우에 한하여 true를 반환. 즉 타입과 값 둘다 검사

▼ NaN 일치 비교

- NaN은 자신과 일치하지 않는 유일한 값이다. 따라서 숫자가 NaN인지 조사하려면 Number.isNaN 사용

▼ 양의 0 음의 0 비교

- 자바스크립트에서는 양의 0과 음의 0이 있는데 이들을 일치 비교를 하든 동등 비교를 하든 true를 반환한다.

```
0===-0; //true
0==-0; //true
```

▼ Object.is 메서드

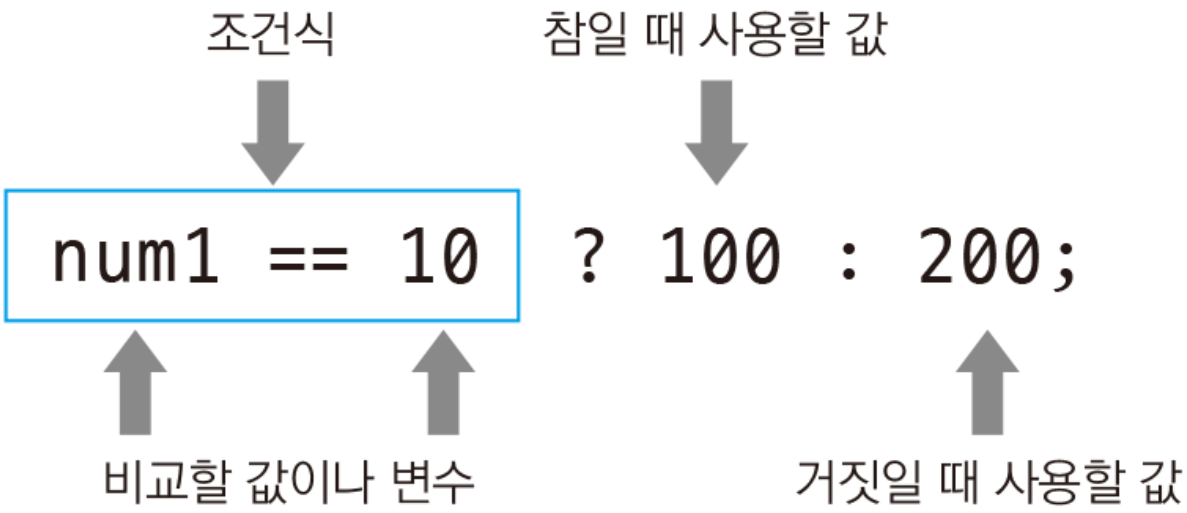
- ES6에서 도입되어 일치 연산자와 동일하게 동작하는데 예측 가능한 정확한 비교 결과를 반환한다.
- NaN일치 비교 가능, 음수0 양수0비교 가능

▼ 대소 관계 비교 연산

- 피연산자의 크기를 비교하여 불리언 값으로 나타냄

할당 연산자	예	동일 표현	부수 효과
=	x = 5	x = 5	○
+=	x += 5	x = x + 5	○
-=	x -= 5	x = x - 5	○
*=	x *= 5	x = x * 5	○
/=	x /= 5	x = x / 5	○
%=	x %= 5	x = x % 5	○

7-4 삼항 조건 연산자



- 첫번째 피연산자가 true이면 두번째 피연산자가 반환되고 false이면 세번째 피연산자가 반환된다
- 즉 ?앞에 쓰이는 첫번째 피연산자는 불리언값으로 평가될 표현식이다.

```
var x=2;

var result=x%2?'짝수':'홀수'
console.log(result); //짝수
```

- 또한 첫번째 피연산자(표현식)는 조건문이다. 따라서 if~else문을 사용해도 삼항 조건 연산자와 유사하게 처리할 수 있다.

```
var x=2,result;

if(x%2) result='짝수';
else result='홀수';
console.log(result); //짝수
```

▼ if~else문과 차이점

- 삼항 조건 연산자 표현식은 값처럼 사용가능한 표현식인 문인 반면 if~else는 값처럼 사용 불가능한 표현식이 아닌 문이다.
- 수행해야하는 조건문이 많다면 if~else문을 그렇지않다면 삼항조건연산자 사용하는 것이 가독성 면에서 좋다

7-5 논리 연산자

논리 연산자	의미	부수 효과
||	논리합(OR)	X
&&	논리곱(AND)	X
!	부정(NOT)	X

표 7-6 논리 연산자

- 우항과 좌항의 피연산자를 논리 연산한다

사진

- 피연산자가 불리언 값이 아니면 불리언 값으로 암묵적 타입 변환을 해서 논리 연산한다

```
!0; //true반환. 0을 false로 암묵적 타입 변환함
!'Hello'; //false반환. truthy값인 'Hello'가 fasle로 암묵적 타입 변
```

- 논리합, 논리곱 연산자의 표현식이 평가 값이 불리언이 아닐 수 있다. (단축 평가)

```
'Cat'&&'Dog';\\Dog
```

논리곱(&&) 연산자에서는 두 번째 피연산자가 논리곱 연산자 표현식의 평가 결과를 결정한다.

이때 논리곱 연산자는 논리 연산의 결과를 결정하는 두 번째 피연산자 문자열 'Dog'를 그대로 반환한다.

```
'Cat'||'Dog';\\Cat
```

논리합(|| |) 연산자는 두 개의 피연산자 중 하나만 true로 평가되어도 true를 반환한다.

논리합 연산자도 좌항에서 우항으로 평가가 진행된다.

첫 번째 피연산자 'Cat'은 Truthy 값이므로 true로 평가된다.

이 시점에 두 번째 피연산자까지 평가해 보지 않아도 위 표현식을 평가할 수 있다.

이때 논리합 연산자는 논리 연산의 결과를 결정한 첫 번째 피연산자 문자열 'Cat'를 그대로 반환한다.

이처럼 논리곱 연산자와 논리합 연산자는 논리 연산의 결과를 결정하는 피연산자를 타입 변환하지 않고 그대로 반환한다.

이를 단축 평가(short-circuit evaluation)라 한다.

▼ 드모르간 법칙

§ 드모르간의 법칙

전체집합 U의 두 부분집합 A, B에 대하여

(1) $(A \cup B)^c = A^c \cap B^c$

(2) $(A \cap B)^c = A^c \cup B^c$

드모르간 법

```
!(x || y) === (!x && !y)
!(x && y) === (!x || !y)
```

이런식으로 드모르간 법칙을 활용하면 복잡한 논리 연산자로 구성된 표현식을 좀 더 가독성있게 표현할 수 있다.

7-6 쉼표 연산자

- 왼쪽 피연산자부터 차례대로 평가하고 마지막 피연산자의 평가 값을 반환한다.

```
var x,y,z;

x=1,y=2,z=3;//3
```

7-7 그룹 연산자

- 소괄호로 감싸 가장 먼저 평가되도록 한다.

7-8 typeof 연산자

- 피연산자의 데이터 타입을 반환하는 연산자

7-9 지수 연산자

- 좌항 피연산자를 밑으로, 우항 피연산자를 지수로 거듭제곱하여 숫자 값을 반환한다
- 음수를 거듭제곱하려면 괄호로 묶어야한다.
- 이항 연산자 중 우선순위가 가장 높다.
- 산술 연산자와 같이 할당 연산자와 같이 쓸 수 있다

```
2**2; //4
(-3)**2 //9
var x=5;
x**=2; //25
2*5**2; //50
```

▼ Math.pow 메서드

- ES7도입된 지수 연산자 이전에 사용한 방법
- 지수 연산자보다는 가독성이 떨어진다

```
Math.pow(2,3); //8
```

7-10 그 외 연산자

다른 장에서 설명 예정

7-11 연산자의 부수효과

- 부수효과 : 다른 코드에 영향을 주는 효과
- 부수효과를 일으키는 연산자 종류

- 할당 연산자
- 증가/감소 연산자
- delete 연산자

7-12 연산자 우선순위

우선순위	연산자
1	( )
2	new(매개변수 존재), ., [ ](프로퍼티 접근), ( )(함수 호출), ?.(옵셔널 체이닝 연산자[9])
3	new(매개변수 미존재)
4	x++, x--
5	!x, +x, -x, ++x, --x, typeof, delete
6	**(이항 연산자 중에서 우선순위가 가장 높다)
7	*, /, %
8	+, -
9	<, <=, >, >=, in, instanceof
10	==, !=, ===, !==

7-13 연산자 결합 순서

- 결합 순서 : 연산자의 어느 쪽(좌항과 우항 중)부터 평가할 것인지 나타내는 순서

결합 순서	연산자
좌항 → 우항	+, -, /, %, <, <=, >, >=, &&, ||, ., [], ( ), ??, ?., in, instanceof
우항 → 좌항	++, --, 할당 연산자(=, +=, -=, ...), !x, +x, -x, ++x, --x, typeof, delete, ? ... : ...