Ch.7 연산자

▼ 🌟++증가,- -감소 연산자

++X : 먼저 피연산자의 값을 증가시킨 후 다른 연산을 수행한다. **선증가 후할당**

X++ : 먼저 다른 연산을 수행한 후 피연산자의 값을 증가시킨다. **선할당 후증가**

▼ +,- 단항 연산자

+단항 연산자

- 숫자 타입이 아닌 피연산자에 +단항 연산자를 사용하면 피연산자를 숫자 타입으로 암묵적 타입 변환하여 반환한다. 이때 피연산자를 변경하는 것은 아니고 숫자 타입으로 변환한 값을 생성해서 반환한다. 따라서 부수효과는 없다.
- +연산자는 피연산자 중 하나 이상이 문자열인 경우 문자열 연결 연산자로 동작한다. (암묵적 타입 변환 또는 타입 강제 변환)

-단항 연산자

- 피연산자의 부호를 반전한 값을 반환한다.
- +단항 연산자와 마찬가지로 숫자타입이 아닌 피연산자에 사용하면 피연산자를 숫자타입으로 변환하여 반환한다. 역시 이때 피연산자를 변경하는 것은 아니고 부호를 반전한 값을 생성해 반환한다. 따라서 부수효과는 없다.

  ▼ 원래의 피연산자를 변경하지 않고, 대신 새로운 값을 생성하기 때문에 부수효과(side effect)가 없습니다.

```
let a = 5;
```

```
let b = -a;
console.log(a); // 5
console.log(b); // -5
```

위 코드에서 a는 5로 설정되고, b는 -a로 설정됩니다. b는 a의 부호를 반전한 새로운 값을 갖게 되며, 원래의 a 값은 변경되지 않고 그대로 5로 남아 있습니다. 여기서 -a는 a의 값을 변경하지 않고 새로운 값 -5를 생성하여 b에 할당한 것입니다.

▼ 동등 비교 연산자 (==) vs 일치 비교 연산자(===)

p81(e115)-p

동등 비교 연산자는 좌항과 우항의 피연산자를 비교할때 먼저 암묵적 타입 변환을 통해 타입을 일치 시킨 후 같은 값인지 비교한다.

일치 비교 연산자는 좌항과 우항의 피연산자가 타입도 같고 값도 같은 경우에 한해 true를 반환한다. 즉, 암묵적 타입변환을 하지 않고 값을 비교한다.

- 일치비교 연산자에서 주의할 것은 NaN이다. NaN은 자신과 일치하지 않는 유일한 값이다. 따라서 숫자가 NaN인지 조사하려면 빌트인 함수 Number.isNaN을 사용한다.

```
NaN === NaN; // false
```

- 숫자 0도 주의하자. 자바스크립트에는 양의 0과 음의 0이 있는데 이들을 비교하면 true를 반환한다.

▼ Object.is 메서드

p82(e116)

NaN과 0을 비교할때 예측 가능한 정확한 비교 결과를 반환한다. 그외에는 일치비교 연산자와 동일하게 동작한다. 예제 07-18

```
-0 === +0; // true
Object.is(-0,+0); // false

NaN === NaN; // false
Object.is(NaN, NaN); // true
```

▼ 삼항조건 연산자 vs if..else문

삼항 조건 연산자

- 조건식의 평가 결과가 불리언 값이 아니면 불리언 값으로 암묵적 타입 변환된다.
- 삼항 조건 연산자의 첫번째 피연산자는 조건식이므로 삼항조건 연산자 표현식은 조건문이다.
- 경우의 수가 3가지라면(양수, 음수 0) 아래와 같이 바꿔 쓸 수 있다. p96(e130)

```
var kind = num ? (num > 0 ? '양수' : '음수') : '영';
```

삼항조건 연산자 표현식은 값으로 평가할 수 있는 표현식인 문이다. 따라서 **삼항 조건 연산자 표현식은 값처럼 다른 표현식의 일부가 될 수 있어 매우 유용하다. p85(e119)**

if...else문은 표현식이 아닌 문이다. 따라서 if...else문은 값처럼 사용할 수 없다.

▼ 논리 연산자

논리 부정 연산자!는 언제나 불리언 값을 반환한다. 만약 피연산자가 불리언 값이 아니면 불리언 타입으로 암묵적 타입변환 된다.

▼ 드 모르간의 법칙

p87(e121)

드 모르간의 법칙(De Morgan's laws)은 논리학과 집합론에서 중요한 규칙으로, 논리 연산자의 부정을 다루는 두 가지 기본적인 법칙을 포함합니다. 이 법칙들은 논리식의 부정을 변형할 때 유용하게 사용됩니다.

드 모르간의 법칙은 다음과 같이 두 가지로 요약할 수 있습니다:

1. **논리 연산에 대한 드 모르간의 법칙**:
   - NOT (A AND B) = (NOT A) OR (NOT B) $\neg(A \wedge B) = (\neg A) \vee (\neg B)$
   - NOT (A OR B) = (NOT A) AND (NOT B) $\neg(A \vee B) = (\neg A) \wedge (\neg B)$

2. **집합론에 대한 드 모르간의 법칙**:
   - $(A \cap B)^c = A^c \cup B^c$
   - $(A \cup B)^c = A^c \cap B^c$

예제 07-28

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

▼ typeof 연산자

- typeof 연산자로 null값을 연산해보면 null이 아닌 object를 반환한다. null을 반환하는 경우는 없다. 따라서 값이 null 타입인지 확인할 때는 일치 연산자===를 사용하자.

```
var foo = null;
foo === null;  // true
```

- 선언하지 않은 식별자를 typeof연산자로 연산하면 ReferenceError가 발생하지 않고 undefined를 반환한다.

▼ 지수연산자

- **
- 이항 연산주 중에서 우선순위가 가장 높다.
- 지수연산자가 도입되기 이전에는 Math.pow메서드를 사용했다.
- 음수를 거듭제곱 밑으로 사용해 계산하려면 괄호로 묶어야 한다. (-5)**2; // 25
- 다른산술연산자와 마찬가지로 할당 연산자와 함께 사용할 수 있다. num **= 2;

▼ 그 외의 연산자

p90(e124)

delete연산자: 객체의 프로퍼티를 삭제하는 연산자.

+대부분의 연산자는 다른 코드에 영향을 주지 않지만 일부 연산자는 다른 코드에 영향을 주는 부수효과가 있다. 부수효과가 있는 연산자는 할당 연산자=, 증가/감소 연산자(++—), delete연산자다.