7. 연산자

연산자(operator)는 하나 이상의 표현식을 대상으로 산술, 할당, 비교, 논리, 타입, 지수 연산 등을 수행해 하나의 값을 만든다.(연산의 대상을 피연산자(operand)라 한다)

```
// 산술 연산자
5 * 4 // -> 20

// 문자열 연결 연산자
'My name is ' + 'Lee' // -> 'My name is Lee'

// 할당 연산자
color = 'red' // -> 'red'

// 비교 연산자
3 > 5 // -> false

// 논리 연산자
true && false // -> false

// 타입 연산자
typeof 'Hi' // -> string
```

산술 연산자

산술 연산자는 피연산자를 대상으로 수학적 계산을 수행해 새로운 숫자 값을 만든다.

산술 연산자는 피연산자의 개수에 따라 **이항**과 **단항** 산술 연산자로 구분할 수 있다.

이항 산술 연산자	의미
+	덧셈
-	뺄셈
*	곱셈
/	나눗셈
%	나머지

```
5 + 2; // -> 7
5 - 2; // -> 3
5 * 2; // -> 10
5 / 2; // -> 2.5
5 % 2; // -> 1
// % = 좌변을 우변값으로 나눴을때 나눠지지 않는 나머지 값
```

연습 문제(연산자1)

```
10 + 10

20 - 10

30 * 2

50 / 5

100000000 % 2

100000001 % 2
```

결과

단항 산술 연산자

단항 산술 연산자는 1개의 피연산자를 산술 연산하여 숫자 값을 만든다

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

- 피연산자 앞에 위치한 전위 증가/감소 연산자는 먼저 피연산자의 값을 증가/감소시킨 후, 다른 연산을 수행한다.

- 피연산자 뒤에 위치한 후위 증가/감소 연산자는 먼저 다른 연산을 수행한 후, 피연산자의 값을 증가/감소 시킨다.

- 단항 연산자

```
// 부호를 반전한다.
-(-10); // -> 10

// 문자열을 숫자로 타입 변환한다.
-'10'; // -> -10

// 불리언 값을 숫자로 타입 변환한다.
-true; // -> -1

// 문자열은 숫자로 타입 변환할 수 없으므로 NaN을 반환한다.
-'Hello'; // -> NaN
```

연습 문제(연산자2)

```
-(+20);  =>

"Hello" + 1004;  =>

-false;  =>

+(-20);  =>

-(-50);  =>

-"Hi";  =>
```

결과

증가/감소 연산자는 피연산자의 값을 변경하는 부수 효과가 있다.

```
var x = 1;

// ++ 연산자는 피연산자의 값을 변경하는 암묵적 할당이 이뤄진다.
x++; // x = x + 1;
console.log(x); // 2

// -- 연산자는 피연산자의 값을 변경하는 암묵적 할당이 이뤄진다.
x--; // x = x - 1;
console.log(x); // 1
```

연습 문제

버튼을 누르면 count값이 증가 되는 구문

```
<body>
    <!--버튼을 클릭하면 increase함수를 호출-->
    <button type="button" onclick="increase()">증가</button>
    <script>
        var count = 0;
        //버튼을 누르면 아래 있는 함수가 호출됨
        function increase(){
            //이곳에 코드 작성

            console.log(count)
        }
    </script>
</body>
```

결과

할당 연산자

할당 연산자는 우항에 있는 피연산자의 평가 결과를 좌항에 있는 변수에 할당한다.

좌항의 변수에 값을 할당하므로 변수 값이 변하는 부수 효과가 있다.

할당연산자	예	동일 표현
=	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

변수 number의 값을 맞춰 보세요

```
var number = 5

number += 5  =>

number -= 5  =>

number *= 6  =>

number /= 10  =>

number %= 2  =>
```

결과

비교 연산자

좌항과 우항의 피연산자를 비교한 다음 그 결과를 불리언 값으로 반환한다.

비교 연산자는 if 문이나 for 문과 같은 제어문의 조건식에서 주로 사용한다.

동등/일치 비교 연산자

동등 비교 연산자와 일치 비교 연산자는 좌항과 우항의 피연산자가 같은 값으로 평가되는지 비교해 불리언 값을 반환한다.

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

```
// 동등 비교
console.log(5 == 5); // true
// 타입은 다르지만 암묵적 타입 변환을 통해 타입을 일치시키며 동등하다.
console.log(5 == '5'); // true

//동등 비교, 결과를 예측하기 어렵고 실수하기 쉽다.
console.log('0' == ''); // false
console.log( 0 == ''); // true
console.log( 0 == '0'); // true
console.log( false == 'false'); // false
console.log( false == '0'); // true
console.log( false == null); // false
console.log( false == undefined); // false
```

결과 값이 true인지 false인지 맞춰 보세요(==)

```
console.log( undefined == false);

console.log('null' == '');

console.log( 1 == '1');

console.log( true == 'true');

console.log( true == '1');

console.log( null == false);
```

결과

일치 비교 연산자

일치 비교(===) 연산자는 좌항과 우항의 피연산자가 타입도 같고 값도 같은 경우에 한하여 true를 반환한다.

```
// 일치 비교
console.log(5 === 5); // true
// 암묵적 타입 변환을 하지 않고 값을 비교한다.(값과 타입이 모드 같은 경우만 true를 반환)
console.log(5 === '5'); // false

// NaN은 자신과 일치하지 않는 유일한 값
console.log(NaN === NaN); // false

// 숫자가 NaN인지 조사하려면 빌트인 함수인 isNaN을 사용
console.log(isNaN(NaN)); // true
console.log(isNaN(10)); // false
console.log(isNaN(1 + undefined)); // true
```

결과 값이 true인지 false인지 맞춰 보세요(===)

```
console.log("abc" === "abc");

console.log("abc" === "ABC");

console.log( undefined === false);

console.log('null' === '');

console.log( 1 === '1');

console.log( true === '1');

console.log( null === false);
```

```
console.log(NaN === NaN);
```

결과

대소 관계 비교 연산자

대소 관계 비교 연산자는 피연산자의 크기를 비교하여 불리언 값으로 반환한다.

대소 관계 비교 연산자	예제	설명
>	x > y	x가 y보다 크다
<	x < y	x가 y보다 작다
>=	x >= y	x가 y보다 크거나 같다
<=	x <= y	x가 y보다 작거나 같다

```
// 대소 관계 비교
5 > 0; // true
5 > 5; // false
5 >= 5; // true
5 <= 5; // true
```

논리 연산자

우항과 좌항의 피연산자(부정 논리 연산자의 경우 우항의 피연산자)를 논리 연산한다

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

```
// 논리합(||) 연산자
true || true  // true
true || false  // true
false || true  // true
false || false  // false
```

```
// 논리곱(&&) 연산자
true && true  // true
true && false  // false
false && true  // false
false && false  // false

// 논리 부정(!) 연산자
!true  // false
!false  // true
```

연습 문제

결과 값이 true인지 false인지 맞춰 보세요(논리 연산자)

```
5 <= 4 || 4 > 3  =>

10 < 5 || 3 > 30  =>

10 == 10 && 20 > 30  =>

20 > 10 && 30 < 60  =>

10 != 10  =>

50 !< 10  =>
```

답

문제 - 로그인 체크

```
<!DOCTYPE html>
<html lang="ko">
<head>
    <meta charset="UTF-8">
```

```
    <meta name="viewport" content="width=device-width, init
ial-scale=1.0">
    <title>로그인 체크</title>
</head>
<body>
    <h1>Login</h1>
    <form id="loginForm">
        <label for="username">Username:</label>
        <input type="text" id="username" name="username" re
quired>
        <br>
        <label for="password">Password:</label>
        <input type="password" id="password" name="passwor
d" required>
        <br>
        <!--버튼을 클릭하면 checkLogin 함수가 호출-->
        <button type="button" onclick="checkLogin()">Login
</button>
    </form>
    <p id="message"></p>

    <script>
        function checkLogin() {
                // validUsername과 validPassword 변수에는 정의
된 올바른 사용자 이름과 비밀번호가 저장 되어 있습니다.
            const validUsername = 'user';
            const validPassword = 'pass';

//1번 username변수에 #username에 입력한 값을, password변수에 #pas
sword에 입력한 값을 할당 합니다.
                        //여기에 코드를 작성해 주세요(1번)

            //2번 && 연산자를 사용하여 username과 password가 val
idUsername과 validPassword와 둘 다 올바른지 확인
            //여기에 코드를 작성해 주세요(2번)

            //아이디가 message인 요소를 선택
            const messageElement = document.getElementById
```

```
('message');
            //3번 username과 password가 올바르면(true) message
Element에 "로그인 성공" 이라는 글자를 넣어 주고, username과 passwo
rd중 하나라도 다르면 "로그인 실패" 라는 글자를 넣어 줍니다.
                        //여기에 코드를 작성해 주세요(3번)
        }
    </script>
</body>
</html>
```

답

---

typeof 연산자

피연산자의 데이터 타입을 문자열로 반환한다.

```
typeof '' // 'string'

typeof 1 // 'number'

typeof NaN // 'number'

typeof true // 'boolean'

typeof undefined // 'undefined'

typeof Symbol() // 'symbol'

typeof null // 'object'

typeof [] // 'object'

typeof {} // 'object'

typeof new Date() // 'object'
```

```
typeof /test/gi // 'object'

typeof function(){} // 'function'
```

지수 연산자

ES7에서 도입된 지수 연산자는 좌항의 피연산자를 밑(base)으로, 우항의 피연산자를 지수로 거듭 제곱하여 숫자 값을 반환한다.

```
2 ** 2;   // -> 4

2 ** 2.5; // -> 5.65685424949238

2 ** 0;   // -> 1

2 ** -2;  // -> 0.25
```

연산자 우선순위

우선순위	연산자
1	()
2	new(매개변수 존재), [ ](프로퍼티 접근, ( )(함수호출), ?.(옵셔널 체이닝 연산자)
3	new(매개변수 미존재)
4	X++, X–
5	!X, +X, -X, ++X, –X, typeof, delete
6	**(이항 연산자중에서 우선순위가 가장 높다)
7	*, /, %
8	+, -
9	<, <=, >, >=, in, instanceof
10	==, !=, ===, !==
11	??(null 병합 연산자)
12	&&
13	||
14	? ... : ...

우선순위	연산자
15	할당 연산자(=, +=, -=, ...)
16	,