# JavaScript 101 - (2) 연산자와 표현식

## JavaScript의 연산자

### 산술 연산자

- 산술연산자는 수학적인 계산을 수행하는데 사용되며 `+`, `-`, `*`, `/`, `%`, `**` 등이 있습니다.

```
var x = 10;
var y = 3;
console.log(x + y); //13
console.log(x - y); //7
console.log(x * y); //30
console.log(x / y); //3.3333333333333335
console.log(x % y); //1 (나머지)
console.log(x ** y); //1000 (거듭제곱)
```

### 대입 연산자

- 대입 연산자는 우측에 있는 값을 좌측에 대입하는 역할을 합니다.

```
var x = 10; //x에 10을 대입합니다.
var y = x; //y에 x값을 대입합니다.
console.log(y); //10
```

### 할당 연산자

- 할당연산자 `+=` 은 좌변의 값에 우변의 값을 더한 값을 좌변에 할당합니다.

```
var x = 10;
x += 5; //x = x + 5 와 같습니다
console.log(x); //15
```

- `-=`, `*=`, `/=` 등도 비슷한 방식으로 작동합니다.

### 비교 연산자

- 비교 연산자는 두 값을 비교하여 불리언 값(`true` 또는 `false`)을 반환합니다. `==`, `!=`, `===`, `!==`, `<`, `>`, `<=`, `>=` 등이 있습니다.

```
var x = 10;
var y = '10';
console.log(x == y); //true (값만 비교)
console.log(x === y); //false (타입까지 비교)
console.log(x != y); //false (값만 비교)
console.log(x !== y); //true (타입까지 비교)
console.log(x < y); //false
console.log(x > y); //false
console.log(x <= y); //true
console.log(x >= y); //true
```

### 논리 연산자

- 논리 연산자는 두 개 이상의 조건을 결합하거나 부정하는데 사용되며 `&&`(AND), `||`(OR), `!`(NOT) 등이 있습니다.

```
var x = true;
var y = false;
console.log(x && y); //false (x와 y가 모두 참일 때만 참)
console.log(x || y); //true (x와 y 중 하나라도 참이면 참)
console.log(!x); //false (x의 반대값)
```

### typeof 연산자

- typeof 연산자는 피연산자의 데이터 타입을 문자열로 반환합니다. 예시:

```
var x = 10;
var y = 'hello';
var z = true;
var w = null;
var u;

console.log(typeof x); //'number'
console.log(typeof y); //'string'
console.log(typeof z); //'boolean'
console.log(typeof w); //'object' (null은 객체로 취급됩니다.)
console.log(typeof u); //'undefined'
```

# JavaScript의 표현식

표현식은 하나 이상의 연산자와 피연산자로 구성되며, 하나의 값으로 평가됩니다. 예를 들어, `2 + 3`은 산술 연산자와 숫자 피연산자로 구성된 표현식이며, `5`라는 값으로 평가됩니다.

JavaScript에서는 다양한 종류의 표현식이 있습니다. 몇 가지 예를 들어보겠습니다.

## 리터럴 표현식

리터럴 표현식은 고정된 값을 나타내는 표현식입니다. 예를 들어, 숫자 리터럴(`10`), 문자열 리터럴(`'hello'`), 불리언 리터럴(`true`), null 리터럴(`null`), undefined 리터럴(`undefined`) 등이 있습니다.

```
//리터럴 표현식
console.log(10); //10
console.log('hello'); //'hello'
console.log(true); //true
console.log(null); //null
console.log(undefined); //undefined
```

## 변수 표현식

변수 표현식은 변수 이름을 나타내는 표현식입니다. 예를 들어, `var x = 10;` 이라고 선언한 후에 `x`라고 쓰면 `x`라는 변수 표현식이 되며, `x`에 할당된 값인 `10`으로 평가됩니다.

```
//변수 표현식
var x = 10;
console.log(x); //10
```

## 함수 호출 표현식

함수 호출 표현식은 함수 이름과 괄호 안에 전달되는 인수들로 구성된 표현식입니다. 예를 들어, `console.log('hello');` 라고 쓰면 `console.log`라는 함수 이름과 `'hello'`라는 인수로 구성된 함수 호출 표현식이 되며, `'hello'`라는 문자열을 콘솔에 출력하고 undefined 값을 반환합니다.

```
//함수 호출 표현식
function add(a,b) {
  return a + b;
}

var result = add(2,3); //add(2,3)은 함수 호출 표현식이며 5로 평가됩니다.
console.log(result); //5

console.log(add(4,5)); //add(4,5)도 함수 호출 표현식이며 9로 평가됩니다.
```

## 객체 접근 표현식

객체 접근 표현식은 객체의 속성이나 메소드에 접근하기 위해 점(`.`) 또는 대괄호(`[]`)를 사용하는 표현식입니다. 예를 들어, `var obj = {name: 'Alice', age: 20};` 이라고 선언한 후에 `obj.name` 또는 `obj['name']` 이라고 쓰면 `obj.name` 또는 `obj['name']` 이라는 객체 접근 표현식이 되며, `obj` 객체의 `name` 속성 값인 `'Alice'`로 평가됩니다.

```
var obj = {name: 'Alice', age: 20};
console.log(obj.name); // Alice
console.log(obj['age']); // 20
```

## 조건부(삼항) 연산자

조건부(삼항) 연산자는 `조건 ? 참일 때 값 : 거짓일 때 값` 형태로 작성되며, 조건을 평가하여 참이면 참일 때 값으로 거짓이면 거짓일 때 값으로 평가되는 표현식입니다. 예를 들어, `var x = 10; var y = x > 0 ? 'positive' : 'negative';` 라고 쓰면 `x > 0 ? 'positive' : 'negative'` 라는 조건부 연산자로 구성된 표현식이 되며, `x > 0` 은 참이므로 `'positive'` 로 평가되어 `y`에 할당됩니다.

```
var x = 10;
var y = x > 0 ? 'positive' : 'negative';
console.log(y); // positive
```