자바스크립트의 기본 문법

변수

변수(Variable)는 값(value)을 저장(할당)하고 그 저장된 값을 참조하기 위해 사용한다. 한 번 쓰고 버리는 값이 아닌 유지(캐싱)할 필요가 있는 값은 변수에 담아 사용한다. 또한 변수 이름을 통해 값의 의미를 명확히 할 수 있어 코드의 가독성이 좋아진다.

변수는 위치(주소)를 기억하는 저장소이다. 위치란 메모리 상의 주소(address)를 의미한다. 즉, 변수란 메모리 주소(Memory address)에 접근하기 위해 사람이 이해할 수 있는 언어로 지정한 식별자(identifier)이다.

변수를 선언할 때 `var` 키워드를 사용한다. 할당 연산자 `=`는 변수에 값을 할당하기 위해 사용한다.

아래의 예에서 x는 변수로 선언되었고 변수 x에는 정수값 6이 할당되었다.

```
var x; // 변수의 선언
x = 6; // 정수값의 할당
```

값

```
var str = 'Hello World';
```

위 예제는 str이라는 이름의 **변수**를 선언하고 문자열 **리터럴** 'Hello World'를 **값**으로 할당하였다. 이때 문자열 리터럴 'Hello World'는 문자열 타입의 값이다.

용어	의미
데이터 타입(Data Type)	프로그래밍 언어에서 사용할 수 있는 값의 종류
변수(Variable)	값이 저장된 메모리 공간의 주소를 가리키는 식별자(identifier)
리터럴(literal)	소스코드 안에서 직접 만들어 낸 상수 값 자체를 말하며 값을 구성하는 최소 단위

값 은 프로그램에 의해 조작될 수 있는 대상을 말한다. 값은 다양한 방법으로 생성할 수 있다. 가장 간단한 방법은 리터럴 표기법(literal notation)을 사용하는 것이다.

```
// 숫자 리터럴
10.50
1001

// 문자열 리터럴
'Hello'
"World"

// 불리언 리터럴
true
false

// null 리터럴
null

// undefined 리터럴
undefined

// 객체 리터럴
{ name: 'Lee', gender: 'male' }

// 배열 리터럴
[ 1, 2, 3 ]

// 정규표현식 리터럴
/ab+c/
```

```
// 함수 리터럴
function() {}
```

숫자, 문자열, 불리언과 같은 원시 타입의 리터럴은 다양한 연산자의 피연산자가 되어 하나의 값으로 평가될 수 있다. 이렇게 리터럴은 연산에 의해 하나의 값이 될 수 있다.

```
// 산술 연산
10.50 + 1001
```

자바스크립트의 모든 값은 데이터 타입을 갖는다. 자바스크립트는 7가지 데이터 타입을 제공한다.

원시 타입 (primitive data type)

- `number`
- `string`
- `boolean`
- `null`
- `undefined`
- `symbol` (New in ECMAScript 6)

객체 타입 (Object data type)

- `object`

자바스크립트는 C나 Java외는 다르게 변수를 선언할 때 데이터 타입을 미리 지정하지 않는다. 다시 말해, 변수에 할당된 값의 타입에 의해 동적으로 변수의 타입이 결정된다. 이를 동적 타이핑이라 하며 자바스크립트가 다른 프로그래밍 언어와 구별되는 특징 중 하나이다.

```
// Number
var num1 = 1001;
var num2 = 10.50;
```

```
// String
var string1 = 'Hello';
var string2 = "World";

// Boolean
var bool = true;

// null
var foo = null;

// undefined
var bar;

// Object
var obj = { name: 'Lee', gender: 'male' };

// Array
var array = [ 1, 2, 3 ];

// function
var foo = function() {};
```

연산자

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

```
// 산술 연산자
var area = 5 * 4;

// 문자열 연결 연산자
var str = 'My name is ' + 'Lee';

// 할당 연산자
```

```
var color = 'red';

// 비교 연산자
var foo = 3 > 5;

// 논리 연산자
var bar = (5 > 3) && (2 < 4);

// 타입 연산자
var type = typeof 'Hi';

// 인스턴스 생성 연산자
var today = new Date();
```

피연산자의 타입은 반드시 일치할 필요는 없다. 이때 자바스크립트는 암묵적 타입 강제 변환을 통해 연산을 수행한다.

```
var foo = 1 + '10'; // '110'
var bar = 1 * '10'; // 10
```

키워드

키워드(keyword)는 수행할 동작을 규정한 것이다. 예를 들어 `var` 키워드는 새로운 변수를 생성할 것을 지시한다.

```
// 변수의 선언
var x = 5 + 6;

// 함수의 선언
function foo (arg) {
  // 함수 종료 및 값의 반환
  return ++arg;
}
```

```
var i = 0;
// 반복문
while (1) {
  if (i > 5) {
    // 반복문 탈출
    break;
  }
  console.log(i);
  i++;
}
```

주석

주석(Comment)은 작성된 코드의 의미를 설명하기 위해 사용한다.

코드는 읽기(이해하기) 쉬워야 한다.(가독성이 좋아야 한다) 여러분이 작성한 코드를 다른 누군가가 읽는다면 “아니, 이게 뭐하는 코드야?”라고 생각하는 순간이 있기 마련이다. 여러분이 해야 하는 일은 바로 그런 부분에 주석을 다는 것이다. (읽기 좋은 코드가 좋은 코드이다)

한줄 주석은 `//` 다음에 작성하며 여러 줄 주석은 `/*`과 `*/`의 사이에 작성한다. 주석은 해석기(parser)가 무시하며 실행되지 않는다.

```
// 주석(Comment)은 작성된 코드의 의미를 설명하기 위해 사용한다. 코드는 읽

/*
  주석(Comment)은 작성된 코드의 의미를 설명하기 위해 사용한다.
  코드는 읽기(이해하기) 쉬워야 한다.
*/
```

하지만 과도한 주석은 오히려 가독성을 해칠 수 있다. 주석 없이도 읽을 수 있는 코드가 최선이다.

```
// Bad
// 변수 x는 나이를 나타낸다. x에는 정수 10을 할당한다.
var x = 10;

// Good
var age = 10;
```

문

프로그램(스크립트)은 컴퓨터(Client-side Javascript의 경우, 엄밀히 말하면 웹 브라우저)에 의해 단계별로 수행될 명령들의 집합이다.

각각의 명령을 문(statement)이라 하며 문이 실행되면 무슨 일인가가 일어나게 된다.

문은 리터럴, 연산자(Operator), 표현식(Expression), 키워드(Keyword) 등으로 구성되며 세미콜론(;)으로 끝나야 한다.

```
var x = 5;
var y = 6;
var z = x + y;

console.log(z);
```

문은 코드 블록(code block, {...})으로 그룹화할 수 있다. 그룹화의 목적은 함께 실행되어져야 하는 문을 정의하기 위함이다.

```
// 함수
function myFunction(x, y) {
  return x + y;
```

```
}

// if 문
if(x > 0) {
  // do something
}

// for 문
for (var i = 0; i < 10; i++) {
  // do something
}
```

문들은 일반적으로 위에서 아래로 순서대로 실행된다. 이러한 실행 순서는 조건문(if, switch)이나 반복문(while, for)의 사용으로 제어할 수 있다 이를 흐름제어(Control Flow)라 한다. 또는 함수 호출로 변경될 수 있다.

```
var time = 10;
var greeting;

if (time < 10) {
  greeting = 'Good morning';
} else if (time < 20) {
  greeting = 'Good day';
} else {
  greeting = 'Good evening';
}

console.log(greeting);
```

다른 언어와 달리 자바스크립트에서는 블록 유효범위(Block-level scope)를 생성하지 않는다. 함수 단위의 유효범위(Function-level scope)만이 생성된다.

표현식

표현식(Expression)은 하나의 값으로 평가(Evaluation)된다. 값(리터럴), 변수, 객체의 프로퍼티, 배열의 요소, 함수 호출, 메소드 호출, 피연산자와 연산자의 조합은 모두 표현식이며 하나의 값으로 평가(Evaluation)된다. 표현식은 결국 하나의 값이 되기 때문에 다른 표현식의 일부가 되어 조금 더 복잡한 표현식을 구성할 수도 있다. 아래의 예에서 5 * 10은 50으로 평가(연산)된다.

```
// 표현식
5              // 5
5 * 10         // 50
5 * 10 > 10    // true
(5 * 10 > 10) && (5 * 10 < 100)  // true
```

문과 표현식의 비교

자연어에서 문(Statement)이 마침표로 끝나는 하나의 완전한 문장(Sentence)이라고 한다면 표현식은 문을 구성하는 요소이다. 표현식은 그자체로 하나의 문이 될 수도 있다.

```
// 선언문(Declaration statement)
var x = 5 * 10; // 표현식 x = 5 * 10를 포함하는 문이다.
// 할당문(Assignment statement)
x = 100; // 이 자체가 표현식이지만 완전한 문이기도 하다.
```

표현식과 문은 매우 유사하여 구별이 어려울 수 있다. 표현식은 평가되어 값을 만들지만 그 이상의 행위는 할 수 없다. 문은 var, function과 같은 선언 키워드를 사용하여 변수나 함수를 생성하기도 하고 if, for, while 문과 같은 제어문을 생성하여 프로그램의 흐름을 제어하기도 한다. 표현식을 통해 평가한 값을 통해 실제로 컴퓨터에게 명령을 하여 무언가를 하는 것은 문이다.

함수

함수란 어떤 작업을 수행하기 위해 필요한 문(statement)들의 집합을 정의한 코드 블록이다. 함수는 이름과 매개변수를 갖으며 필요한 때에 호출하여 코드 블록에 담긴 문들을 일괄

적으로 실행할 수 있다.

```
// 함수의 정의(함수 선언문)
function square(number) {
  return number * number;
}
```

함수는 호출에 의해 실행되는데 한번만 호출할 수 있는 것이 아니라 여러번 호출할 수 있다.

```
// 함수의 정의(함수 선언문)
function square(number) {
  return number * number;
}

// 함수의 호출
square(2); // 4
```

동일한 작업을 반복적으로 수행해야 한다면 (동일한 구문을 계속해서 중복해서 작성하는 것이 아니라) 미리 정의된 함수를 재사용하는 것이 효율적이다. 이러한 특성은 코드의 재사용이라는 측면에서 매우 유용하다.

객체

자바스크립트는 객체(object) 기반의 스크립트 언어이며 자바스크립트를 이루고 있는 거의 "모든 것"이 객체이다. 원시 타입(Primitives)을 제외한 나머지 값들(함수, 배열, 정규표현식 등)은 모두 객체이다.

자바스크립트 객체는 키(이름)와 값으로 구성된 프로퍼티(property)의 집합이다. 프로퍼티의 값으로 자바스크립트에서 사용할 수 있는 모든 값을 사용할 수 있다. 자바스크립트의 함수는 일급 객체이므로 값으로 취급할 수 있다. 따라서 프로퍼티 값으로 함수를 사용할 수도 있으며 프로퍼티 값이 함수일 경우, 일반 함수와 구분하기 위해 메소드라 부른다.

```
var person = {
  name: 'Lee',
  gender: 'male',
  sayHello: function () {
    console.log('Hi! My name is ' + this.name);
  }
};

console.log(typeof person); // object
console.log(person); // { name: 'Lee', gender: 'male', sayHel

person.sayHello(); // Hi! My name is Lee
```

이와 같이 객체는 데이터를 의미하는 프로퍼티(property)와 데이터를 참조하고 조작할 수 있는 동작(behavior)을 의미하는 메소드(method)로 구성된 집합이다. 객체는 데이터(프로퍼티)와 그 데이터에 관련되는 동작(메소드)을 모두 포함할 수 있기 때문에 데이터와 동작을 하나의 단위로 구조화할 수 있어 유용하다.

자바스크립트의 객체는 객체지향의 상속을 구현하기 위해 "프로토타입"이라고 불리는 객체의 프로퍼티와 메소드를 상속받을 수 있다. 이 프로토타입은 타 언어와 구별되는 중요한 개념이다.

배열

배열(array)은 1개의 변수에 여러 개의 값을 순차적으로 저장할 때 사용한다. 자바스크립트의 배열은 객체이며 유용한 내장 메소드를 포함하고 있다.

```
var arr = [1, 2, 3, 4, 5];

console.log(arr[1]); // 2
```