

5. 변수

프로그래밍 언어는 기억하고 싶은 값을 메모리에 저장하고, **저장된 값을 읽어 들여 재사용하기 위해 변수라는 메커니즘을 제공한다.**



변수(variable)는 **하나의 값을 저장하기 위해 확보한 메모리 공간 자체 또는 그 메모리 공간을 식별하기 위해 붙인 이름**을 말한다. (값의 위치를 가리키는 상징적인 이름)

1) 변수 선언 방식

변수명 변수 값
`var result = 10 + 20;`

- 메모리 공간에 저장된 값을 식별할 수 있는 고유한 이름(예제에서는 result)을 **변수이름 (또는 변수명)**이라 한다.

- 변수에 저장된 값(예제에서는 30)을 **변수 값**이라고 한다.
- 변수에 값을 저장하는 것을 **할당assignment(대입,저장)**이라 하고, 변수에 저장된 값을 읽어 들이는 것을 **참조(reference)**라고 한다.
- 변수를 선언할 때는 **var, let, const** 키워드를 사용한다. (**let**과 **const**는 **ES6**에서부터 도입된 키워드)

```
// 변수는 하나의 값을 저장하기 위한 수단이다.
var userId = 1;
var userName = 'Lee';
```

2) 식별자 작성 방식

자바스크립트에서는 식별자를 작성할 때 다음과 같은 작성 방식을 사용할 수 있습니다.

```
var firstName; //카멜 케이스 방식 = 일반적으로 변수, 함수 이름을 지을 때 사용

var first_name; //스네이크 케이스 방식

var FirstName; //파스칼 케이스 = 일반적으로 생성자 함수, 클래스 이름을 지을 때 사용

var $elerm; //헝가리안 케이스
```



자바스크립트에서는 식별자를 작성할 때 코드의 가독성과 통일성을 위해 관행적으로 **Camel Case**방식을 많이 사용합니다.



자바스크립트에서 하이픈(-)은 뉘셈을 위해 예약된 키워드이므로, 식별자를 작성할 때는 사용할 수 없습니다.

키워드(keyword)

- 자바스크립트에서는 몇몇 단어들을 특별한 용도로 사용하기 위해 미리 예약하고 있습니다.
- 이렇게 미리 예약된 단어들을 **키워드(keyword)** 또는 **예약어(reserved word)**라고 합니다.
- 이러한 키워드들은 프로그램 내에서 **식별자로 사용할 수 없습니다.**

```
var firstVar = 10; // var는 변수의 정의를 위해 예약된 키워드입니다.
```

```
function myFirstFunc { // function은 함수의 정의를 위해 예약된 키워드입니다.
    var secondVar = 20; // var는 변수의 정의를 위해 예약된 키워드입니다.
}
```

- | | | |
|--------------|--------------|----------------|
| • break | • abstract | • static |
| • case | • boolean | • super |
| • catch | • byte | • synchronized |
| • continue | • char | • throws |
| • debugger* | • class | • transient |
| • default | • const | • volatile |
| • delete | • debugger | ... |
| • do | • double | |
| • else | • enum | |
| • finally | • export | |
| • for | • extends | |
| • function | • final | |
| • if | • float | |
| • in | • goto | |
| • instanceof | • implements | |
| • new | • import | |

- return
- switch
- this
- throw
- try
- typeof
- var
- void
- while
- with
- int
- interface
- long
- native
- package
- private
- protected
- public
- short

4) 값의 할당

- 변수에 값을 할당(대입,저장)할 때는 할당 연산자 =를 사용한다.
- 할당 연산자는 우변의 값을 좌변의 변수에 할당한다.

```
var score; // 변수 선언
console.log(score); // undefined = 변수가 선언 이후 값이 할당된 적이 없다는 뜻이다.

score = 80; // 값의 할당

//단축표현
var score = 80; // 변수 선언과 값의 할당

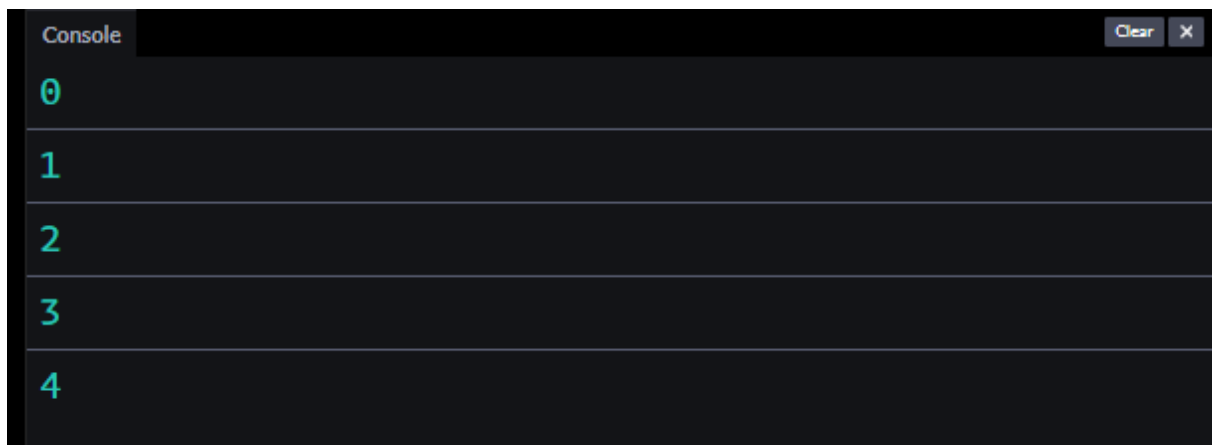
score = 90; // 값의 재할당
console.log(score); // 90 = 선언한 변수는 값을 재할당할 수 있다.
```

자바스크립트 엔진은 변수 선언을 다음과 같은 2단계에 거쳐 수행한다.

- 선언 단계 : 변수 이름을 등록해서 자바스크립트 엔진에 변수의 존재를 알린다.
- 초기화 단계: 값을 저장하기 위한 메모리 공간을 확보하고 암묵적으로 undefined를 할당해 초기화 한다.

변수 선언 생략

```
for (i = 0; i < 5; ++i)
  console.log(i);
```



변수 선언을 생략해도 된다.

위에서 변수 i를 선언하지 않고, 그냥 i = 0;를 실행했다.
이것은 문법 오류가 아니다.

javascript 언어 문법에서는, 변수 선언을 생략해도 된다. 필수가 아니다.
i = 0; 문장을 처음 실행할 때, 변수 i가 자동으로 생성되기 때문이다.

즉 javascript 문법에서는, 변수를 미리 선언하지 않아도, 어떤 변수에 값이 처음 대입될 때,
그 변수가 자동으로 생성된다.

"use strict";

javascript 소스 코드 파일 선두에 "use strict"; 선언이 있으면,
이 소스 파일에서는 변수를 미리 선언하지 않고 사용할 경우에 문법 오류가 발생한다.

```
"use strict";
```

```
for (i = 0; i < 5; ++i)  
  console.log(i);
```

상수 선언 const

```
let s1 = "hello world";  
const s2 = "hello world";
```

```
console.log(s1);  
console.log(s2);
```

const 키워드를 사용하여 상수를 선언한다.

```
s1 = "안녕하세요";
```

let 변수 s1의 값은 변경할 수 있다.

```
s2 = "안녕하세요";
```

const 변수의 값은 변경할 수 없으므로, 에러가 발생한다.