Chap5 표현식과 문

5-1 값

- 식(표현식 expression)이 평가되어 생성된 결과를 말한다.

```
//10+20은 평가되어 숫자 값 30을 생성한다
10+20; //30
```

- 데이터 타입을 가지며, 메모리에 2진수, 즉 비트의 나열로 저장된다.
- 값을 생성하는 방법 : 리터럴 사용

5-2 리터럴

- 사람이 이해할 수 있는 문자 또는 약속된 기호를 사용해 값을 생성하는 표기법 (notation).

```
//숫자 리터럴 3
3
```

  즉 3은 사람이 이해할 수 있는 아라비아 숫자를 사용해 숫자 리터럴 3을 코드에 기술하면 자바스크립트 엔진은 이를 평가해 숫자 값 3을 생성한다.

- 리터럴의 다양한 종류

리터럴	예시	비고
정수 리터럴	`100`	
부동소수점 리터럴	`10.5`	
2진수 리터럴	`0b01000001`	`0b`로 시작
8진수 리터럴	`0o101`	ES6에서 도입. `0o`로 시작
16진수 리터럴	`0x41`	ES6에서 도입. `0x`로 시작
문자열 리터럴	`'Hello'` `"World"`	
불리언 리터럴	`true` `false`	
null 리터럴	`null`	
undefined 리터럴	`undefined`	
객체 리터럴	`{ name: 'Lee', address: 'Seoul' }`	
배열 리터럴	`[ 1, 2, 3 ]`	
함수 리터럴	`function() {}`	
정규 표현식 리터럴	`/[A-Z]+/g`	

5-3 표현식

- **값으로 평가될 수 있는 문이다.** 따라서 문법적으로 값이 위치할 수 있는 자리에는 표현식도 위치할 수 있다.
- 표현식이 평가되면서 새로운 값을 생성하거나 기존 값을 참조한다.
- 리터럴도 표현식이다.
- 예제

```
var score=100; //100은 표현식이다.
```

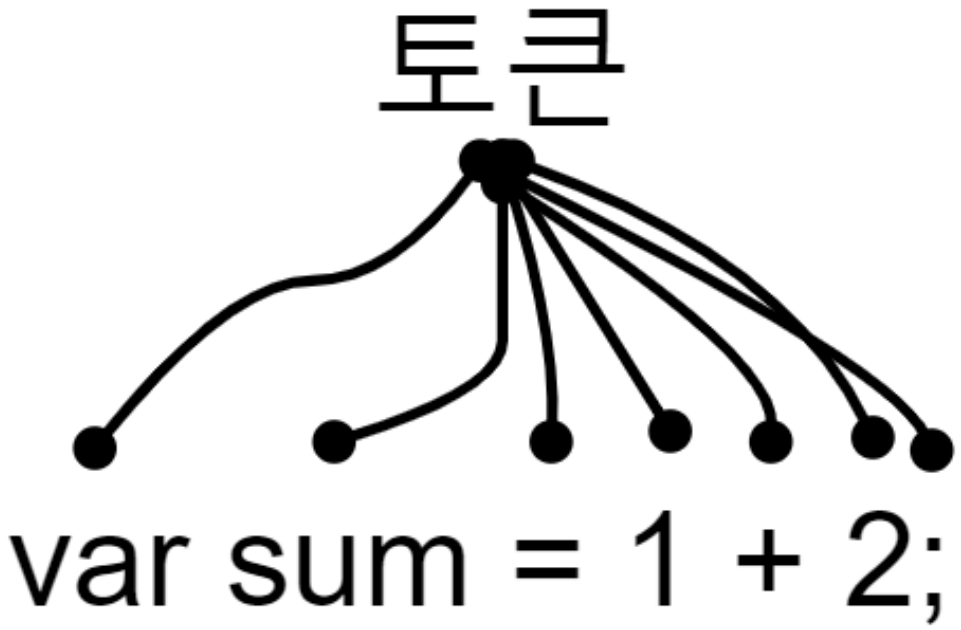
```
var score=50+50; //50+50도 숫자 값100을 생성하므로 표현식이다.
```

```
score; //결과값 100, 기존 값을 참조하므로 표현식이다.
```

5-4 문

- 프로그램을 구성하는 기본 단위이자 최소 실행 단위
- 문은 여러 토큰으로 구성된다.

> 💡 토큰
> 문법적으로 더이상 나눌 수 없는 코드의 기본 요소.
> ex) 키워드, 식별자, 연산자, 리터럴, 세미콜런, 마침표 등의 특수기호



- 문의 종류 : 선언문, 할당문, 조건문, 반복문

```
// 변수 선언문
var x;

// 할당문
x=5;

//함수 선언문
function foo() {}

// 조건문
```

```
if(x?1){console.log(x);}

// 반복문
for (var i=0; i<2; i++){console.log(i);}
```

5-5 세미콜론과 세미콜론 자동 삽입 기능

- 세미콜론의 의미 : 문의 종료. 자바스크립트 엔진이 세미콜론을 기준으로 문의 종료 위치를 파악하고 순차적으로 하나씩 문을 실행한다.
- 0개 이상의 문을 중괄호로 묶는 코드블록 뒤에는 세미콜론을 붙이지 않는다. (ex if문, for 문, 함수 등)
- 세미콜론은 옵션이다. 즉 생략 가능
- 생략 가능하지만 ESLint같은 정적 분석 도구에서도 세미콜론 사용을 기본으로 설정하고 있고 TC39(ECMAScript 기술 위원회)도 세미콜론 사용을 권장하는 분위기이므로 사용을 권장한다.

  ▼ ESLint

  JavaScript와 관련된 코드에서 문제를 식별하고 일관된 코딩 스타일을 유지하기 위한 정적 코드 분석 도구

  ▼ ASI

  자바스크립트 엔진은 코드를 해석할 때 ASI 규칙을 적용하여 누락된 세미콜론을 자동으로 삽입한다.

  - ASI 규칙
    - 줄바꿈이 되는 행의 마지막에
    - 행의 마지막이 } 이고 줄바꿈이 되어 다음 행이 시작될때
    - 파일의 끝에 도달할 때
    - return 이 있는 행의 마지막에
    - break 가 있는 행의 마지막에
    - throw 가 있는 행의 마지막에
    - continue 가 있는 행의 마지막에

5-6 표현식인 문과 표현식이 아닌 문

- 구분 방법 : 변수에 할당해보기

```
//표현식이 아닌 문은 값처럼 사용할 수 없다.
var foo = var x; //SyntaxError:Unexpected token var
```

즉 var x; 와 같은 선언문은 표현식이 될 수 없고 표현식인 문은 값으로 평가할 수 있어야한다.

▼ 완료 값 (completion value)

크롬 개발자 도구에서 표현식이 아닌 문을 실행하면 언제나 undefined를 출력한다. 이를 완료 값이라고 한다. 완료값은 표현식의 평가 결과가 아니다. 따라서 다른 값과 같이 변수에 할당할 수 없고 참조할 수도 없다.

```
> let x = 10; //표현식이 아닌 문. 변수에 할당해보면 값처럼 사용불가
undefined
> x //표현식인 문. 변수에 할당해보면 값처럼 사용 가능
10
```

표현식인 아닌 문은 완료값을 반환하고 표현식인 문은 평가된 값을 반환한다.