Ch4 변수

변수란?

변수는 하나의 값을 저장하기 위해 확보한 메모리 공간 자체 또는 그 메모리 공간을 식별하기 위해 붙인 이름을 말한다.

→ 값의 위치를 가리키는 상징적인 이름

✔️ 변수는 하나의 값을 저장하기 위한 메커니즘이므로 여러 개의 값을 저장하기위해서는 여러 개의 변수를 사용해야 한다. 단, 배열이나 객체 같은 자료구조를 사용하면 관련이 있는 여러 개의 값을 그룹화해서 하나의 값처럼 사용할 수 있다.

식별자

변수 이름을 **식별자**라고도 하는데 식별자는 어떤 값을 구별해서 식별할 수 있는 고유한 이름을 말한다.

값은 메모리 공간에 저장되어 있고 식별자는 어떤 값이 저장되어 있는 **메모리 주소를 기억**해야 한다.

변수 선언

변수 선언이란 변수를 생성하는 것인데 이는 값을 저장하기 위한 메모리 공간을 확보하고 변수 이름과 확보된 메모리 공간의 주소를 연결해서 값을 저장할 수 있게 준비하는 것이다.

변수를 선언할 때는 `var`, `let`, `const` 키워드를 사용한다.

✔️ `var` 키워드의 단점

블록 레벨 스코프를 지원하지 않고 함수 레벨 스코프를 지원함

`var` 키워드의 단점을 보완하기 위해 도입한 것이 `let`과 `const`!

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

✔️ undefined

undefined는 자바스크립트에서 제공하는 원시 타입의 값이다.

변수 선언 이후 변수에 값을 할당하지 않으면 undefined라는 값이 암묵적으로 할당되어 초기화 된다.

`var` 를 사용한 변수 선언은 선언 단계와 초기화 단계가 동시에 진행되므로 쓰레기 값이 나올 위험으로부터 안전하다.

만약 선언하지 않은 식별자에 접근하면 **ReferenceError**(참조 에러)가 발생한다.

변수 호이스팅

변수 호이스팅은 변수 선언문이 코드의 선두로 끌어 올려진 것처럼 동작하는 자바스크립트 고유의 특징을 말한다.

값의 할당, 재할당

변수에 값을 **할당**할때는 할당 연산자 = 를 사용한다.

```
console.log(score); // undefined

var score;  // ① 변수 선언
score = 80; // ② 값의 할당
```

```
console.log(score); // 80
```

변수에 값을 할당할 때는 이전 값 undefined 가 저장되어 있던 메모리 공간을 지우고 메모리 공간에 80을 할당하는 것이 아니라 새로운 메모리 공간을 확보하고 그 곳에 할당 값 80을 저장한다.

재할당은 이미 값이 할당되어 있는 변수에 새로운 값을 또다시 할당하는 것을 말한다.

상수는 값을 재할당할 수 없고 한번 정해지면 변하지 않는다.

✔️ const 키워드를 사용해 선언한 변수는 재할당이 금지된다. 따라서 const 키워드를 사용하면 상수를 표현할 수 있다. (반드시 상수만을 위해 사용하는 것은 아님)

※ 가비지 콜렉터

가비지 콜렉터는 애플리케이션이 할당한 메모리 공간을 주기적으로 검사하여 더 이상 식별자가 참조하지 않는 메모리를 해제하는 기능을 말한다.
자바스크립트는 가비지 콜렉터를 내장하고 있는 언어로 메모리 누수를 방지한다.

※ 언매니지드 언어 / 매니지드 언어

C언어와 같은 **언매니지드 언어**는 개발자가 명시적으로 메모리를 할당하고 해제하기 위해 저수준 메모리 제어 기능을 제공한다. **메모리 제어를 개발자가 주도**할 수 있어서 개발자의 역량에 따라 **최적의 성능**을 확보할 수 있지만 반대의 경우 치명적 오류를 생산할 수 있다.

자바스크립트 같은 **매니지드 언어**는 메모리의 할당 및 해제를 위한 메모리 관리 기능을 언어 차원에서 담당하고 **개발자의 직접적인 메모리 제어를 허용하지 않는다**. 사용하지 않는 메모리의 해제는 **가비지 콜렉터**가 수행하고 **일정한 생산성을 확보**할 수 있다는 장점이 있지만 성능면에서 손실을 감수해야 한다.

식별자 네이밍 규칙

- 특수문자를 제외한 문자, 숫자, 언더스코어(_), 달러 기호($) 를 포함할 수 있다.
  - 단, 특수문자를 제외한 문자, 언더스코어(_), 달러 기호($) 로 시작해야 한다. 숫자로 시작하는 것은 허용하지 않는다.
- 예약어는 식별자로 사용할 수 없다.
  - 식별자로 사용 가능하지만 strict mode 에서는 사용 불가
- ES5부터 유니코드 문자를 허용하므로 한글이나 일본어 식별자도 사용할 수 있다.
  - 하지만 바람직하지 않으므로 권장하지 않는다.
- 변수는 쉼표(,) 로 구분해 하나의 문에서 여러 개를 한번에 선언할 수 있다.
  - 가독성이 나빠지므로 권장하지는 않는다.
- 대소문자를 구별한다.

※ 네이밍 컨벤션은 하나 이상의 영어 단어로 구성된 식별자를 만들 때 가독성 좋게 하기 위해 규정한 명명 규칙이다.

```
// 1. 카멜 케이스 (camelCase)
var firstName;

// 2. 스네이크 케이스 (snake_case)
var first_name;

// 3. 파스칼 케이스 (PascalCase)
var FirstName;

// 4. 헝가리언 케이스 (typeHungarianCase)
var strFirstName; // type + identifier
```

```
var $elem = document.getElementById('myId'); // DOM 노드
var observable$ = fromEvent(document, 'click'); // RxJS 옵저버블
```

→ 일반적으로 변수나 함수의 이름에는 카멜 케이스를, 생성자 함수나 클래스의 이름에는 파스칼 케이스를 사용한다.