4장

- 변수
  - 데이터를 관리하기 위한 핵심 개념
  - JS는 개발자가 직접적인 메모리 제어를 허용하지 않는다
    - 컴퓨터는 CPU를 이용해 계산하고 메모리를 이용해 기억, 이 기억을 재사용하려면 저장된 메모리 공간에 접근 해야하는데 직접 접근하는 것은 치명적 오류 발생 가능
  - 메모리를 저장하고, 저장된 값을 읽어 들여 재사용하기 위해 변수를 사용한다.
  - 값의 위치를 가리키는 상징적인 이름
  - 하나의 값을 저장할 수 있지만 배열이나 객체 같은 자료구조를 사용하면 여러 개의 값을 그룹화해서 하나의 값처럼 사용 가능하다.
  - 메모리 공간에 저장된 값을 식별할 수 있는 고유한 이름을 **변수 이름**이라고 한다.
  - 변수에 저장된 값을 **변수 값**이라고 한다.
  - 변수에 값을 저장하는 것을 **할당**
  - 변수에 저장된 값을 읽어 들이는 것을 **참조**
  - 명확한 네이밍은 코드를 이해하기 쉽게 만들며, 협업과 품질 향상에 도움이 된다.
- 식별자
  - 변수 이름을 식별자라고도 부른다.
  - 어떤 값을 구벌해서 식별할 수 있는 고유한 이름
  - 값이 아니라 메모리 주소를 기억
  - 변수, 함수, 클래스 등의 이름은 모두 식별자
  - 메모리 상에 존재하는 어떤 값을 식별할 수 있는 이름은 모두 식별자
- 변수 선언
  - 값을 저장하기 위한 메모리 공간 확보
  - 변수를 사용하려면 반드시 선언이 필요
    - var. let, const 키워드 사용

    ▼ var

        - 단점
            - 함수 레벨 스코프를 지원해서 의도치 않게 전역 변수가 선언되어 부작용이 발생하기도 한다.
    - 변수 선언에 의해 확보된 메모리 공간에는 비어 있지 않고 `undefined` 값이 할당 되어 초기화 한다.
        - Js만의 독특한 특징
    - 변수 선언은 2단계를 걸쳐간다
        - 선언 단계: 변수 이름을 등록해 변수의 존재를 알린다.
        - 초기화 단계: 메모리 공간을 확보하고 `undefined` 를 할당
    - 변수 이름은 실행 컨텍스트에 등록된다.
    - 쓰레기 값
        - 초기화 단계를 거치지 않으면 이전의 값이 남는 것
    - `ReferenceError`
        - 선언하지 않은 식별자에 접근할 경우 에러 발생

```
var score;
```

- 키워드
    - 자신이 수행해야 할 약속된 동작을 수행
    - ex) var 키워드는 변수를 선언해야 한다.
- 호이스팅
    - 런타임
        - JS는 인터프리티에 의해 한 줄씩 순차적으로 실행하는 시점
    - 변수 선언은 런타임 이전 단계에서 먼저 실행된다.
    - JS는 모든 선언문을 소스코드에서 찾아내 먼저 실행한다.
    - 변수 선언이 소스코드 어디에 있든 상관없이 먼저 실행한다.
    - 변수 선언문이 코드의 선두로 끌어 올려진 것처럼 동작하는 것
    - var, let, const, function, function*, class* 키워드를 사용해서 선언하면 호이스팅된다.

    - 선언문은 런타임 이전 단계에서 먼저 실행

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

var score; // 변수 선언
```

- 값의 할당
    - 변수에 값을 저장하는 것
    - 변수 선언과 값의 할당을 하나의 문으로 사용 가능

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

var score = 80; // 합친거
```

    - 변수 선언과 값의 할당의 실행 시점이 다르다.
    - 할당 할떄 unedefind와 같은 메모리 공간에 저장이 아니라 새로운 메모리 공간에 값을 할당한다.
- 값의 재할당
    - 이미 값이 할당되어 있는 변수에 새로운 값을 또다시 할당하는 것

```
var score = 80; // 선언과 할당
score = 90; // 재할당
```

    - 변수 키워드에 따라 다를 수 있다.
        - `const` 재할당이 금지되어 있다.
    - 불필요한 값들은 가비지 콜렉터에 의해 메모리에서 자동 삭제된다.
        - 가비지 콜렉터
            - 사용하지 않는 메모리를 해제 하는 기능
            - 메모리 누수를 방지
            - Js에 내장되어 있는 매니지드 언어
    - 삭제되는 시점은 예측할 수 없다.
- 식별자 네이밍 규칙

- 카멜 케이스 (camelCase)

```
var firstName
```

- 스네이크 케이스 (snake_case)

```
var first_name
```

- 파스칼 케이스 (PascalCase)

```
var FirstName
```

- 변수나 함수의 이름에는 카멜 케이스를 사용하고 생성자 함수, 클래스의 이름에는 파스칼 케이스를 사용한다.