4장 변수

정의

하나의 값을 저장하기 위해 확보한 메모리 공간 자체 또는 그 메모리 공간을 식별하기 위해 붙인 이름

알쓸신잡

- var의 단점
    - 블록 레벨 스코프를 지원하지 않고 함수 레벨 스코프를 지원함 → 의도치 않게 전역 변수로 선언됨
- 변수를 선언 후 아직 변수에 값을 할당하지 않으면 undefined라는 값이 암묵ㅈㄱ으로 할당되어 초기화됨
- undefinded란?
    - JS에서 제공하는 원시 타입의 값 → 6장에서 소개
- **선언하지 않은 변수 참조 시 ReferenceError(참조 에러)'**
- 할당

```
var a = 1;
```

단어 정리

- 식별자 === 변수이름
- 할당 === 대입
- 키워드 === JS 코드를 해석하고 실행하는 JS 엔진이 수행할 동작을 규정한 일종의 명령어

변수 선언 실행 시점과 호이스팅

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

- 해당 부분은 **ReferenceError** 가 야기될 것 같지만 발생하지 않고 undefined가 나옴

  → 변수 선언이 소스 코드가 한 줄씩 순차적으로 실행되는 시점, 런타임이 아니라 그 이전 단계에서 먼저 실행되기 때문임

  왜?

  JS 엔진은 소스 코드를 한 줄씩 순차적으로 실행하기에 앞서 먼저 소스 코드의 평가과정을 거치면서 소스 코드를 실행하기 위한 준비를 함. 이때 소스 코드 실행을 위한 준비 단계인 소스 코드의 평가 과정에서 JS 엔진은 변수 선언을 포함한 모든 선언문을 소스 코드에서 찾아내 먼저 실행함

  해당 평과 과정이 끝나면 비로소 변수 선언을 포함한 모든 선언문을 제외하고 소스 코드를 한 줄씩 순차적으로 실행함

  해당 이유는 변수 선언이 소스코드가 순차적으로 실행되는 런타임 이전 단계에서 먼저 실행된다는 증거이며 변수 선언문이 코드의 선두로 끌어 올려진 것처럼 동작하는 JS고유의 특징을 **변수 호이스팅**이라고 함

> 💡 아래의 코드들을 실행한 값을 알아오시오. 그리고 그 이유도

```
// 각각의 console.log의 출력 값을 알아오시오.
console.log(a);
var a;
a= 11;
console.log(a);
```

```
// 각각의 console.log의 출력 값을 알아오시오.
console.log(a);
a = 10;
var a;
console.log(a);
```

GC(Gabage Collector)가 있는가?

- JS는 내장하고 있는 매니지드 언어임

매니지드 언어

메모리의 할당 및 해제를 위한 메모리 관리 기능을 언어 차원에서 담당하고 개발자의 직접적인 메모리 관리 제어를 허용하지 않음. 사용하지 않은 메모리에 대해서 GC가 직접 해제를 수행하며 이 또한 개발자가 관여할 수 없음. 어느 정도의 메모리 매니징에 대한 관리가 되지만 성능면에서의 단점을 감수해야하는 언어

언매니지드 언어

C언어와 같은 언어는 개발자가 명시적으로 메모리를 할당하고 해제하기 위해 저 수준 메모리 제어 기능을 제공. 해당 언어는 개발자의 역량에 의해 최고의 매니징이 될 수도 있으며 역대급으로 치명적인 매니징이 될 수도 있음

JS 식별자(변수이름)네이밍 규칙

- 특수문자를 제외한 문자, 숫자, 언더스코어(_), 달러 기호($)를 포함할 수 있음
- 식별자는 특수문자를 제외한 문자, 언더스코어, 달러기호로 시작해야하며 숫자로 시작하는 것을 허용하지 않음
- 예약어 사용 금지

예약어

await	beark	case	catch	class	const
continue	debugger	default	delete	do	else
enum	export	extends	false	finally	for
function	if	`implments`	import	in	instanceof
`interface`	`let`	new	null	package	private
`protected`	`public`	return	super	`static`	switch
this	throw	true	try	typeof	var
void	while	with	`yield`		

- 코드 표시된 단어들은 네이밍 가능하나 strict mode에서는 사용 불가능

네이밍 케이스

- 카멜 케이스(cameCase)

```
var firstName;
```

- 스네이크 케이스(snake_case)

```
var first_name;
```

- 파스칼 케이스(PascalCase)

```
var FirstName
```

> 💡 헝가리안 케이스에 대해서 알아오시오

→ 대부분 ECMAScript 표준을 따라 카멜 케이스 || 파스칼 케이스를 따르고 있음