[JS-Deep Dive] 13장. 스코프

Status	시작 전

스코프

- 스코프

> - 식별자가 유효한 범위
> : 모든 식별자(변수이름, 함수이름, 클래스이름, 등은 자신이 선언된 위치에 의해 다른 코드가 식별자 자신을 참조할수 있는 유효 범위가 결정됨.
> -스코프라는 개념이 없다면 같은 이름을 갖는 변수는 충돌을 일으키므로 프로그램 전체에서 하나밖에 사용할수 없다.

- 식별자

> 식별자: 어떤 값을 구별할수 있어야 하므로 유일 해야함. 즉 하나의 값은 유일한 식별자에 연결 되어야함.

- 스코프의 종류

구분	설명	스코프 / 변수	

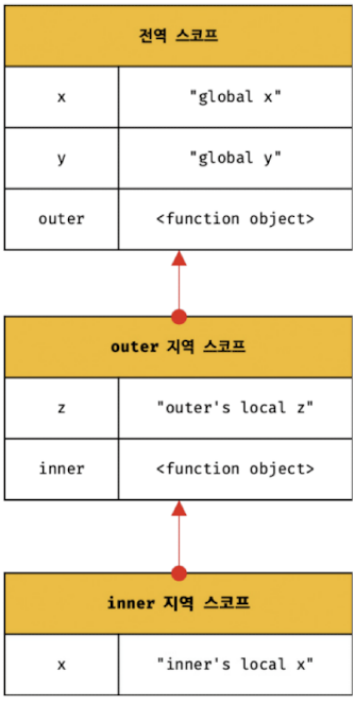
전역	코드의 가장 바깥 영역	전역/전역	어디서든지 참조가능
지역	함수 몸체 내부	지역/지역	자신의 지역 스코프 와 하위 지역 스코프 에서 유효

- 스코프체인

> 자바스크립트 엔진은 스코프 체인을 통해 참조할 변수를 검색한다
> 스코프는 함수의 중첩에 의해 계층적 구조를 갖는다
> 모든 지역 스코프의 최상위 스코프는 전역 스코프다

```
var x = "global x";
var y = "global y";

function outer() {
  var z = "outer's local z";

  console.log(x); // ① global x
  console.log(y); // ② global y
  console.log(z); // ③ outer's local z

  function inner() {
    var x = "inner's local x";

    console.log(x); // ④ inner's local x
    console.log(y); // ⑤ global y
    console.log(z); // ⑥ outer's local z
  }

  inner();
}

outer();

console.log(x); // ⑦ global x
console.log(z); // ⑧ ReferenceError: z is not defined
```

함수 몸체 내부에서 정의한 함수: 중첩함수
중첩함수를 포함하는 함수 : 외부함수



스코프가 계층적으로 연결된것 : 스코프 체인

- 자바 엔진은 코드를 실행하기에 앞서 위 그림과 유사한 구조인 렉시컬 환경을 실제로 생성함.

▼ 렉시컬 환경이란?

-식별자와 그에 대응하는 값, 주변 스코프에 대한 정보를 관리하는 구조

자바스크립트 입장의 스코프

- 자바스크립트 엔진 입장에서의 스코프

> -엔진입장에서의 스코프: 식별자를 검색할때 사용하는 규칙
> -자바 스크립트 엔진은 스코프를 통해 어떤 변수를 참조해야할지 결정
> ⇒ 식별자 결정

- 스코프 검색의 단계

> 1. 변수(혹은 함수) 선언이 실행되면 변수(함수) 식별자가 렉시컬 환경에 키 로 등록함
> 2. 할당이 일어나면 렉시컬 변수(함수) 식별자에 해당하는 값을 변경
> 3.스코프 체인을 통해 변수(함수) 를 참조하는 코드의 스코프에서 시작하여 상위 스코프 방향으로 이동하며 선언된 변수를 검색한다. (이를통해 상위 스코프에서 선언한 변수를 하위스코프에서도 참조 가능)
> 이때, 절대 하위스코프로 내려가면서 식별자를 검색하는 일은 없다.

함수레벨 스코프

var 키워드로 선언된 변수는 오로지 함수의 코드블록(함수의 몸체) 만을 지역스코프로 인정한다.

Dynamic scope & Lexical scope

```
var x = 1

function foo () {
    var x = 10
    bar()
}
function bar() {
    console.log(x)
}
foo()  // 1
bar() // 1
```

함수의 상위 스코프를 결정하는 두가지 방식, 프로그래밍 언어는 두가지 방식중 하나로 상위 스코프를 결정함.

Dynamic scope

> 동적 스코프
> 함수를 어디서 호출 했는가

lexical scope

> 정적 스코프
> -함수를 어디서 정의(선언)했는가
> 대부분의 프로그래밍 언어가 이 방식을 따른다.

아래부터는 추가 내용

최소한의 전역 변수 사용

- 전역 변수의 문제점

> 애플리케이션과 웹 페이지 내 모든 코드 사이에서 공유 되므로 실수를 범할수 있다
> 전역변수는 선언이 되면 프로그램 실행이 끝날때 까지 메모리 공간을 차지하므로, 메모리 부족의 원인이 될수 있다.

전역변수 사용의 최소화 방법

- 전역 변수 객체 만들기

애플리케이션에서 전역변수 사용을 위해 전역변수 객체를 만들어 놓는것

더글라스 크락포드 - 자바스크립트 핵심 가이드

```
var MYAPP = {};

MYAPP.student = {
  name: 'Lee',
  gender: 'male'
};

console.log(MYAPP.student.name);// 'Lee'
```

- 즉시 실행 함수를 이용
  - 즉시 실행함수는 즉시 실행되고 바로 전역에서 사라짐.

```
function () {
  var MYAPP = {};

  MYAPP.student = {
    name: 'Lee',
    gender: 'male'
  };
```

```
  console.log(MYAPP.student.name);
}());
```

참고:

- javaScript Patterns
- 더글라스 크락포드 - 자바스크립트 핵심 가이드