Scope

전역 / 지역 변수

변수는 크게 전역 / 지역 변수로 나뉩니다. 이 차이는 **변수가 선언된 블록 내에서만 사용 가능한지, 아니면 어디서나 사용 가능한지**의 차이입니다.

```
let globalVal = "Hello Global"

function myFunc() {
  let localVal = 'Hello local"
}

myFunc()
```

위 코드를 보면 globalVal은 **전역 변수**, localVal은 myFunc 함수에서 선언되어 myFunc 함수 내에서만 사용 가능한 **지역 변수**입니다. 그럼 다 전역변수로 만들어 버리면 되지 않나 라고 생각할 수도 있지만 어디서나 접근할 수 있고 호출될 수 있다는 것은 편의성 측면에서 장점 같지만 반대로 원하지 않는 변수가 언제 어디서나 값이 재할당 될 수 있다는 위험성을 갖는다.

스코프와 스코프 체인

스코프는 위에서 나타난 것처럼 **변수 또는 함수 등 식별자에 대한 유효 범위**입니다. **스코프 체인**이란 **변수를 찾기 위해 스코프를 안에서 바깥쪽으로 탐색하는 과정**을 의미합니다.

```
let globalVal = "Hello Global"

function myFunc() {
```

```
  let localVal = 'Hello local"
  alert(globalVal)
}

myFunc()
```

위 코드를 보면 myFunc()가 호출되면서 alert(globalVal) 이 실행되는데 이 때 js는 먼저 **가장 가까운 스코프에서부터 변수를 찾게 됩니다**. 즉, 가장 가까운 스코프인 myFunc 내부에서 globalVal을 찾고 여기서 **못 찾으면 상위 스코프로 올라가 전역 스코프에서 globalVal를 찾게 됩니다.**

이렇게 **식별자를 찾기 위해 스코프를 안에서 바깥쪽으로 탐색하는 과정**을 스코프 체이닝 이라고 합니다.

반면 반대로 아래와 같은 경우 스코프 탐색이 불가합니다.

```
function myFunc() {
  let localVal = 'Hello JavaScript"
  return localVal
}

let myVar = myFunc()

alert(localVal)
```

이는 localVal을 찾기 위해 바깥에서 안쪽으로 탐색하는 과정이 필요한데 이는 불가능한 방식입니다.

자바스크립트 스코프의 특징

자바스크립트의 스코프는 타 언어와 다른 특징을 가지고 있다. 대부분의 C-family language는 **블록 레벨 스코프**를 따른다. 즉, 코드 블록 내에서 유효한 스코프를 의미하는데 이를 C언어 코드를 보면서 이해해보려 한다.

```
int main(void) {
  if (1) {
    int x = 5;
    prinf("x = %d\n", x);
  }

  printf("x = %d\n", x); // Error

  return 0;
}
```

C언어는 **블록 레벨 스코프**를 따르기 때문에 if 라는 블록 내에서 선언된 변수 x는 그 바깥에서 참조가 불가능하다.

하지만 JavaScript는 타 언어와 달리 **함수 레벨 스코프**를 따른다. 함수 레벨 스코프란 코드 블록 내에서 선언된 변수는 함수 코드 블록 내에서만 유효하고 함수 외부에서는 참조할 수 없다는 것이다.

하지만, ES6 버전 이후부터 도입된 let은 블록 레벨 스코프를 사용할 수 있다.

```
var x = 0;
{
  var x = 1;
  console.log(x); // 1
}
console.log(x); // 1

let y = 0;
{
  let y = 1;
  console.log(y); // 1
}
console.log(y); // 0
```

렉시컬 스코프

렉시컬 스코프란 **정적 스코프**를 의미합니다. 렉시컬 스코프를 알아보기 전에 스코프의 생성 시점부터 알아보겠습니다.

```
let name = 'Jehyeok'

function sayHello() {
  let name = 'devandy'
  callMyName()
}

function callMyName() {
  alert('Hello ' + name)
}

sayHello()
```

위 코드는 Hello Jehyeok 이 실행되는 코드입니다. callMyName이 호출되는 시점이 아닌 선언 시점에 스코프가 생성되면 전역변수 name이 할당되면서 Jehyeok 을 가져와 출력한 것입니다.

이렇게 **스코프의 생성 시점은 호출 시점이 아닌 선언 시점**이라는 점을 꼭 기억해 둬야 합니다.

이제 렉시컬 스코프에 대해 알아보겠습니다.

```
var x = 1

function foo() {
  var x = 10
  bar()
}

function bar() {
  console.log(x)
}

foo() // 1
bar() // 1
```

위에서 공부한 내용을 토대로 둘 다 출력 값은 1입니다.

이렇게 **렉시컬 스코프**는 **함수를 어디서 호출하는지가 아니라 어디에 선언하였는지에 따라 결정**된다.

반대로 **함수의 호출 시점에 따라 결정되는 스코프**를 **동적 스코프**라고 한다.