[참고] JavaScript 변수 범위, 호이스트, 클로저

- **변수 범위**
  - 변수가 존재하는 컨텍스트
  - 변수에 엑세스 할 수 있는 위치와 해당 컨텍스트에서 변수에 액세스 할 수 있는지 여부를 지정
  - 지역 범위(local scope), 전역 범위(global scope)를 가짐

▪ 지역 변수(local variables = function-level scope)

- 블록 수준 범위(block-level scope)를 지원하지 않음

```
var name = "Richard";    // 전역 변수
if (name) {
    name = "Jack";       // 전역 변수
    console.log(name);   // Jack
}
console.log(name);       // Jack
```

- 함수 수준 범위(function-level scope)을 지원
  - 함수 안에 정의된 변수는 지역 범위(local scope)를 가지며, 해당 함수와 내부함수에서만 접근이 가능
  - 지역 변수는 함수 내에서 전역 변수 보다 높은 우선 순위를 가짐

```
var name = "Richard";    // 전역 변수
function showName() {
    var name = "Jack";   // 지역 변수. showName() 함수에서만 접근 가능
    console.log(name);   // Jack
}
console.log(name);       // Richard
```

▪ 지역 변수(local variables = function-level scope)

- **var 키워드 없이 변수를 선언하면 전역 범위(global scope)를 가지게 됨**

```
var name = "Michael Jackson";      // 전역 변수
function showCelebrityName() {
    console.log(name);
}
function showOrdinaryPersonNameGood() {
    var name = "Hong Gildong";     // 지역 변수. 전역 변수를 덮어쓰는 것을 방지
    console.log(name);
}
function showOrdinaryPersonNameBad() {
    name = "Johnny Evers";         // 전역 변수 오염
    console.log(name);
}

showCelebrityName();               // Michael Jackson
showOrdinaryPersonNameGood();      // Hong Gildong
showCelebrityName();               // Michael Jackson
showOrdinaryPersonNameBad();       // Johnny Evers
showCelebrityName();               // Johnny Evers
```

▪ 전역 변수(global variables)

- 함수 밖에 선언된 모든 변수는 전역 범위(global scope)를 가짐
- 브라우저에서 전역 컨텍스트 또는 범위는 window 객체 또는 전체 HTML 문서를 가리킴
  ➔ 전역 변수는 전체 어플리케이션에서 사용 가능
  ➔ 전역 변수는 window 객체를 통해 접근이 가능

```
/* 전역 변수 선언 */
var myName = "Richard";            // myName = "Richard"; 또는 var myName; 또는 myName;
console.log(window.myName);        // Richard
console.log("myName" In window);   // true
```

- **var 키워드 없이 변수를 선언하면 자동으로 전역 컨텍스트에 추가**

```
function showAge() {
    age = 90;                      // 전역 변수
    console.log(age);              // 90
}
var firstName = "Richard";         // 전역 변수
{
    var firstName = "Bob";         // 전역 변수를 참조 ➔ 자바 스크립트는 블록 범위를 지원하지 않음
}
showAge();                         // 90
console.log(age);                  // 90
console.log(firstName);            // Bob
```

▪ 전역 변수(global variables)

- 전역 범위 오염 예 1

```
for (var i = 1; i <= 10; i++) { // 변수 i는 전역 변수
    console.log(i);             // 1~10까지 출력
}
function aNumber() {
    console.log(i);             // 11
}
aNumber();                      // 11
```

- 전역 범위 오염 예 2

```
var highValue = 200;
var constantVal = 2;
var myObj = {
    highValue: 20,
    constantVal: 5,
    calculateIt: function() {
        setTimeout(function() {
            console.log(this.constantVal * this.highValue); // this 객체는 myObj가 아니라 window 객체를 참조
        }, 2000);
    }
}
myObj.calculateIt();            // 400 = 200 * 2
```

▪ 전역 변수(global variables)

- **전역 변수 오염 방지 ➔ 가급적 전역 범위의 변수가 생성되지 않도록 한다.**

```
/* BAD */
var firstName, lastName;                              // 전역 변수
function fullName() {
     console.log("Full Name : " + firstName + " " + lastName);
}

/* GOOD */
function fullName() {
     var firstName = "Michael", lastName = "Jackson";  // 지역 변수
     console.log("Full Name : " + firstName + " " + lastName);
}
```

▪ 호이스트(hoist)

- 변수의 선언이 끌어 올려지는 것
- 변수의 선언이 함수 안에 있는 경우 함수의 최상위로, 함수 밖에 있는 경우 전역 컨텍스트의 최상위로 끌어 올려진다.
- 변수의 선언이 초기화나 할당할 때 발생하는 것이 아니라, 함수 또는 전역 컨텍스트의 최상위로 끌어 올려져서 처리된다.

```
function showName() {
    console.log("First Name : " + name);    // First Name : undefined  ➔ 지역변수 name이 호이스트 되었음
    var name = "Ford";
    console.log("Last Name : " + name);     // Last Name : Ford
}
showName();

/* 자바스크립트 엔진의 해석 */
function showName() {
    var name;                               // 변수 name이 호이스트 됨. 할당은 이후에 이루어지므로 undefined 상태
    console.log("First name : " + name);    // First Name : undefined
    name = "Ford";                          // 변수 name에 값을 할당
    console.log("Last Name : " + name);     // Last Name : Ford
}
```

▪ 클로저(closure)

- 외부함수의 변수에 접근할 수 있는 내부함수
- 클로저는 세가지 스코프 체인(scope chain)을 가짐
  - 자신의 블록 내에 정의된 변수에 대한 접근
  - 외부함수의 변수(파라미터 포함)에 대한 접근
  - 전역 변수에 대한 접근

- 기본적인 클로저 예

```
function showName(firstName, lastName) {
    var nameIntro = "Your name is ";
    function makeFullName() {
        return nameIntro + firstName + " " + lastName;    // 외부함수의 변수 뿐 아니라 파라미터까지 사용
    }
    return makeFullName();
}
showName("Michael", "Jackson");                           // Your name is Michael Jackson
```

▪ 클로저(closure)

- 비동기, none-blocking 아키텍처의 핵심 기능으로 활용

```
$(function() {
    var selections = [];
    $(".niners").click(function() {                    // 이 클로저는 외부함수의 selections 변수에 접근
        selections.push(this.prop("name"));            // 외부함수의 selections 변수를 갱신
    });
});
```

- 함수 팩토리로 활용 – 특정한 값을 함수의 인자에 덧붙일 수 있는 함수를 생성

```
function makeAdder(x) {
    return function(y) {
        return x + y;
    };
}
var add5 = makeAdder(5);
var add10 = makeAdder(10);
console.log(add5(2));      // 7
console.log(add10(2));     // 12
```

▪ 클로저 규칙과 부수 효과

- **클로저는 외부함수가 반환된 이후에도 외부함수의 변수에 접근이 가능하다.**

```
function celebrityName(firstName) {
    var nameIntro = "This is celebrity is ";
    function lastName(theLastName) {
        return nameIntro + firstName + " " + theLastName; // 내부함수는 외부함수의 변수와 파라미터에 접근이 가능
    }
    return lastName;
}
var mjName = celebrityName("Michael");   // 외부함수 반환
mjName("Jackson");                       // This celebrity is Michael Jackson
                                         // 외부함수 반환 후 클로저(lastName) 호출
                                         // 클로저는 외부함수 반환 후에도 외부함수의 변수와 파라미터에 접근이 가능
```

▪ 클로저 규칙과 부수 효과

- **클로저는 외부함수의 변수에 대한 참조를 저장한다.**

```
function celebrityID() {
    var celebrityID = 999;
    return {                          // 내부함수를 가진 객체를 리턴
        getID: function() {
            return celebrityID;       // celebrityID의 현재값을 리턴
        },
        setID: function(theNewID) {
            celebrityID = theNewID;   // celebrityID의 값을 변경
        }
    }
}
var mjID = celebrityID();             // celebrityID 외부함수 반환
mjID.getID();                         // 999
mjID.setID(567);                      // 외부함수의 변수값을 변경
mjID.getID();                         // 567
                                      // 변경된 외부함수의 변수값을 리턴
```

▪ 클로저 규칙과 부수 효과

- **모듈 패턴 : 클로저를 이용해서 프라이빗 변수와 함수에 접근하는 퍼블릭 함수를 정의**

```
var makeCounter = function() {
    var privateCounter = 0;
    function changeBy(val) {
        privateCounter += val;
    }
    return {
        increment: function() {
            changeBy(1);
        },
        decrement: function() {
            changeBy(-1);
        },
        value: function() {
            return privateCounter;
        }
    };
};
```

```
var counter1 = makeCounter();
var counter2 = makeCounter();

console.log(counter1.value()); /* 0 */
console.log(counter2.value()); /* 0 */

counter1.increment();
counter1.increment();
console.log(counter1.value()); /* 2 */
console.log(counter2.value()); /* 0 */

counter1.decrement();
console.log(counter1.value()); /* 1 */
console.log(counter2.value()); /* 0 */
```