- 2회차 : 조건문, 전역변수와 지역변수, 스코프, 호이스팅, 반복문 -

/* if (조건문) */

- 조건문이란 주어진 조건에 따라 결과값을 출력하는 것을 의미한다.
- 조건문의 조건을 만들기 위해서 보통 비교 연산자, 논리 연산자를 사용한다.

```
// if
var a = 20;
var b = 40;

if ( a < b ) {
    console.log("a는 b보다 작다.");
}


// if ~ else
if ( a > b ) {
    console.log("a는 b보다 크다.");
} else  {
    console.log("a는 b보다 작다.");
}


// else if {
var c = 60;

if ( a > b ) {
    console.log("a는 b보다 크다.");
} else if ( b > c ) {
    console.log("b는 c보다 크다.");
} else if ( a < c ) {
    console.log("a는 c보다 작다.");
} else if ( b < c ) {
    console.log("b는 c보다 작다.");
} else  {
    console.log("모든 조건을 만족하지 않는다.");
}


// 중첩 if
if ( a !== b ) {
    if (a > b) {
        console.log("a는 b보다 크다");
    } else {
        console.log("a는 b보다 작다");
    }
} else {
    console.log("a와 b는 같다");
}
```

- 참고 내용 : 논리 연산자로 IF문 효과 구현하기

```
var test = true;
var isTrue = function() {
    console.log("Test is true.");
};



var isFalse = function() {
    console.log("Test is false.")
};



// if 문에 해당
( test && isTrue());



// else 문에 해당
test = false;
( test || isFalse());



// 응용 코드
function theSameOldFoo(name) {
    name = name || "Bar";
    console.log("My best friend's name is " + name);
}


theSameOldFoo();
theSameOldFoo("Beau");
```

/* 삼항 연산자 */

```
// condition ? expr1 : expr2
var age = 15;

if(age >= 18) {
    console.log("You are an adult!");
} else {
    console.log("You are a kid");
}

// IF 문을 삼항 연산자로 변경하기
var result = (age >= 18) ? "You are an adult!" : "You are a kid";
console.log(result);




// Tip
var stop;

age >= 18 ? (
    console.log("Ok, you can go."),
    stop = false
) : (
    console.log("Sorry, you are much too young!"),
    stop = true
);




var firstCheck = false,
    secondCheck = false,
    access = firstCheck
        ? "Access denied"
        : secondCheck
            ? "Access denied"
            : "Access granted";

console.log(access);
```

/* 전역변수(Global Variable)와 지역변수(Local Variable) */

- 변수는 유효 범위(scope)에 따라 전역변수와 지역변수로 구분된다
- 전역변수는 함수 외부에서 선언된 변수로 모든 영역에서 접근할 수 있다.
- 지역변수는 함수 내부에서 선언된 변수로 함수가 실행되면 생성되고, 함수가 종료되면 소멸하게 된다. 단, 함수 외부에서는 접근이 불가능하다.

```
var gv = "전역변수";

function func() {
    var lv = "지역변수";

    console.log(gv);
    console.log(lv);
}

func();
// console.log(gv);
// console.log(lv);
```

```
var gv = "전역변수";
console.log("1 " + gv);

function func() {
    gv = "함수 내부 변수”;    // 함수 내부에서 var를 사용하지 않으면, 전역변수의 값을 변경
    console.log("2 " + gv);
}

func();

console.log("3 " + gv);
```

```
var gv = "전역변수";

if(true) {
    var lv = “지역변수??????”;

    console.log(gv);
    console.log(lv);
}

// console.log(gv);
// console.log(lv);
```

/* 전역변수와 지역변수를 쉽게 구분하는 방법 */

- 모든 전역변수는 window 객체의 프로퍼티이다.

```
// 크롬 console 탭에서 확인
var gv = "전역변수";

window.gv;

// hasOwnProperty() : 해당 객체의 프로퍼티로 존재하는지 확인
console.log(window.hasOwnProperty(“gv”));
```

/* 스코프 (Scope) */

- 스코프란 변수에 접근하는 범위를 결정짓는 요소를 말한다.
- 스코프는 전역 스코프와 지역 스코프, 두 종류가 있다.
- 변수가 함수 바깥에 선언되어있다면, 전역 스코프라고 한다.

```
var gv = "전역 변수";

function func() {
    console.log(gv);
}

func();
```

- 자바스크립트 안에서 지역 스코프는 함수 스코프다.

```
function func() {
    var lv = "지역 스코프";
    console.log(lv);
}

func();

// console.log(lv);
```

- 함수는 서로의 스코프에 접근할 수 없다

```
function funcOne() {
    var func1 = "Func One 안에 있는 변수";
}

function test() {
    funcOne();
    console.log(func1);
}

test();
```

- **스코프 체인 (Scope Chain = 중첩 스코프 = 스코프 버블)**
- 함수가 다른 함수 내부에서 정의되었다면, 내부 함수는 외부 함수의 변수에 접근할 수 있지만, 외부 함수는 내부 함수의 변수에 접근할 수 없다.

```
function outerFunc () {
    var outer = 'Outer Func';

    function innerFunc() {
        var inner = 'Inner Func';
        console.log(outer)
    }

    innerFunc();
    console.log(inner);
}

outerFunc();
```

```
var a = 10;

function func1() {
    var b = 20;

    function func2() {
        var c = 30;
        console.log(a + b + c);
    }

    func2();

}

func1();
```

- 글로벌 스코프 안에는 변수 a
- func1 스코프 안에 변수 b
- func2 스코프 안에 변수 c

- 위 코드가 실행될 때 스코프 체인은 func2 안쪽에서부터 위로 올라가게 된다.
- 스코프에 대한 검색은 기본적으로 아래에서 위는 가능하지만, **위에서 아래로는 불가능하다.**

- **렉시컬 스코핑**
- 스코프는 함수를 호출할 때가 아닌 함수가 **선언될 때 생성된다.**

```
var name = 'inkwon';

function func1() {
    console.log(name);
}

function func2() {
    name = 'jun';
    func1();
}

func2();
```

```
var name = 'inkwon';

function func1() {
    console.log(name);
}

function func2() {
    var name = 'jun';
    func1();
}

func2();
```

- 함수 func1과 함수 fun2의 상위 스코프는 전역 스코프이다.

/* 변수 호이스팅 */

- 모든 변수 선언은 호이스트된다. 호이스트란, 변수의 영향력이 스코프에 따라 선언과 할당으로 분리되는 것을 의미한다. 만약 변수가 함수 바깥에 정의되어 있다면, 전역 영역의 최상위로, 함수 내에 정의되어 있다면 함수의 최상위로 호이스트가 변경된다.

```
console.log(a);


// console.log(a);
var a;
a = "I am A!";
// console.log(a);



// console.log(b);
var b = "I am B!";
// console.log(b);
```

/* 함수 호이스팅 */

```
func();
function func() {
    console.log("Func!");
}


funcVar();
// console.log(funcVar);
var funcVar = function() {
    console.log(“Func Var!!”);
};
```

/* JavaScript 수학 연산 (Math) */

```
// 절대값을 반환
var num_abs = Math.abs(-3);
console.log(num_abs);


// 정수로 올림
var num_ceil = Math.ceil(0.3);
console.log(num_ceil);


// 정수로 내림
var num_floor = Math.floor(10.9);
console.log(num_floor);


// 랜덤한 숫자를 추출
var num_random = Math.random();
console.log(num_random);   //  0 ~ 1 사이 값을 추출
```

/* 문자를 숫자로 변환하는 함수 */

```
// 문자열 숫자를 정수로 변환
var str1 = "20.6";
var num1 = parseInt(str1);

console.log(num1);


// 문자열 숫자를 소수로 출력
var str2 = "20.1";
var num2 = parseFloat(str2);

console.log(num2);
```

/* Loops (반복문) */

- 반복문 : 컴퓨터에게 반복적인 어떤 작업을 지시할 때 사용되는 구문
- 대표적인 반복문 : while, for

```
// while 문
var num = 0;

while (num < 10) {
    console.log(num);
    num++;
}

** while 문의 조건으로는 항상 true 가 삽입되어야 실행이 됨
** 그러나 제한이 없는 true 거나 false 일 경우에는 Endless Loops ( 무한루프 )가 발생할 수 있음



// do … while
// 조건문이 true 든 false 이든 일단 무조건 한번은 실행하고, 조건문이 false 가 될 때까지 반복
var i = 12;
do {
    console.log(i + " ");
    i++;
} while (i < 10);



//for 문
for (var i = 0; i < 10; i++) {
    console.log(i)
}


// 반복문을 탈출하고 싶을 때는 break 사용
for ( var i = 0; i < 10; i++ ) {
    if ( i === 5 ) {
        break;
    }
    console.log('Hello World ' + i);
}
```

- 객체에 적용할 수 있는 반복문

```
var student = {
    name : "inkweon",
    country : "Korea",
    age : 31,
    skills : ["JavaScript", "HTML", "CSS"],
    sum : function(num1, num2) {
        return num1 + num2;
    }
};


for (var prop in student) {
    console.log(student[prop]);
}
```

- 배열에 적용할 수 있는 반복문

```
var array = ['Apple', 'Banana', 'Tomato', 'Melon'];


for (var i = 0; i < array.length; i++) {
    console.log(array[i]);
}

for (var index in array) {
    console.log(array[index]);
}

array.forEach(function (element) {
    console.log(element);
});

array.map(function(num) {
    console.log(num);
});
```

- forEach, map 차이점 : forEach문은 return 문을 받지 못한다.

/* 배경 색상 변경 */

```
(function() {

    var hexValue = document.getElementById('hex-value');
    var btn = document.getElementById('btn');

    btn.addEventListener('click', createHex);

    function createHex() {
        var hexValues = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, "A", "B", "C", "D",
"E", "F"];
        var hexColor = '#';

        for(var i = 0; i < 6; i++) {
            var random = Math.floor(Math.random() * hexValues.length);
            hexColor += hexValues[random];
        }

        document.body.style.backgroundColor = hexColor;
        hexValue.textContent = hexColor;
    }

})();
```

- textContent : 텍스트 요소를 가져오거나 설정할 때 사용된다. (단, IE < 9에서는 지원되지 않는다.)

- **콜백함수**
- 콜백함수는 맛집 예약과 유사하다. 맛집에 자리가 없을 경우 대기자 명단에 이름을 쓴 후 자리가 날 때까지 기다리거나 주변을 둘러볼 수 있다. 만약 자리가 생기면 전화나 문자로 연락이 온다. 여기서 연락을 받는 시점이 콜백 함수가 호출되는 시점이다.
- 위 코드는 버튼을 클릭했을 때 (=전화로 연락을 받았을 때) 콜백 함수인 createHex가 호출된다.

/* 랜덤 인용 문구 */

```
(function () {

    var quotes = [……];

    var btn = document.getElementById('generate-btn');

    btn.addEventListener('click', function () {

        var random = Math.floor(Math.random() * quotes.length);
        document.getElementById('quote').textContent = quotes[random].quote;
        document.querySelector('.author').textContent = quotes[random].author;

    });

})();
```

/* 텍스트 전달하기 */

```
(function() {

    var message_form = document.getElementById('message-form');

    message_form.addEventListener('submit', function(e) {

        e.preventDefault();

        var message = document.getElementById('message').value;

        if(message === "") {

            var feedback = document.querySelector(‘.feedback');

            feedback.classList.add('show');
            setTimeout(function() {
                feedback.classList.remove('show');
            }, 2000)

        } else {

            document.querySelector('.message-content').textContent = message;
            message_form.reset();

        }

    })

})();
```

/* 슬라이드 효과 */

```
(function () {

    var customers = [……];
    var index = 0;

    document.querySelectorAll('.btn').forEach(function (item) {
        item.addEventListener('click', function (e) {
            e.preventDefault();

            // console.log(e.target);
            if (e.target.classList.contains('prevBtn')) {

                // 배열의 가장 앞 데이터에 접근했을 때
                if (index === 0) {
                    index = customers.length;
                }

                index--;
                document.getElementById('customer-img').src = customers[index].img;
                document.getElementById('customer-name').textContent =
customers[index].name;
                document.getElementById('customer-text').textContent =
customers[index].text;

            }

            if (e.target.classList.contains('nextBtn')) {
                // 배열의 가장 끝 데이터에 접근했을 때
                if (index === (customers.length - 1)) {
                    index = -1;
                }

                index++;
                document.getElementById('customer-img').src = customers[index].img;
                document.getElementById('customer-name').textContent =
customers[index].name;
                document.getElementById('customer-text').textContent =
customers[index].text;

            }
        })
    })
})();
```

/* 살펴보기 : 자바스크립트 Property & Method */
/* String Property & Method */

- 문자열 길이 (1) : length

```
var textLength = "Hello World”;

console.log(textLength.length); // 11 출력, testLength (객체), length ( 프로퍼티 )
```

- 문자열 길이 (2) : trim()

```
var textLength = " Hello World ";
console.log(textLength.length);
console.log(textLength.trim().length);
```

- 문자 추출 (1) : charAt()

```
var str = "Nice To Meet you";

console.log(str.charAt(0));
```

- 문자 추출 (2) : slice(start, and)

```
var str = "Nice To Meet you";

console.log(str.slice(0, 4));
```

- 문자 분할 : split()

```
var str = "Nice To Meet you";

console.log(str.split(' '));     // 띄어쓰기를 기준으로 문자 분할
```

- 문자 교체 : replace(search value, new value)

```
var str = "Nice To Meet you";

console.log(str.replace('Nice', 'Hello'));
```

- 문자 찾기 : indexOf()

```
var str = "Nice To Meet you";

console.log(str.indexOf('To'));  // 문자가 존재하지 않을 경우 -1을 반환
```

- 문자 대소문자 변화 : toUpperCase(), toLowerCase()

```
var str = "Nice To Meet you";

console.log(str.toUpperCase());
console.log(str.toLowerCase());
```

- 그 외 문자열 메서드 : https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/String

/* Array Property & Method */

- 배열 길이 : length

```
var fruit = ["apple", "banana", "tomato"];

console.log(fruit.length);
```

- 배열에 데이터 추가 (1) : push()

```
var fruit = ["apple", "banana", "tomato"];

fruit.push("strawberry", "grape");
console.log(fruit);
```

- 배열에 데이터 추가 (2) : unshift()

```
var fruit = ["apple", "banana", "tomato"];

fruit.unshift("strawberry", "grape");
console.log(fruit);
```

- 배열에 있는 데이터 삭제 (1) : pop()

```
var fruit = ["apple", "banana", "tomato"];

fruit.pop();
console.log(fruit);
```

- 배열에 있는 데이터 삭제 (2) : shift()

```
var fruit = ["apple", "banana", "tomato"];

fruit.shift();
console.log(fruit);
```

- 배열 안에 이는 요소 이어 붙이기 : join()

```
var fruit = ["apple", "banana", "tomato"];
var str = fruit.join(' / ');

console.log(str);
```

- **서로 다른 배열 이어 붙이기 : concat()**

```
var alpha = ['a', 'b', 'c'];
var numeric = [1, 2, 3];

var alphaNumeric = alpha.concat(numeric);
console.log(alphaNumeric);
```