day20-javascript

태그	
날짜	@2022년 10월 27일

Break & Continue

break는 반복문을 강제로 빠져나오기 위해 사용되고 continue는 반복 중 건너 뛰기 위해 사용된다.

```
break_test.html - 브레이크 예시

<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
</head>
<body>
    <script>
        var i;
        for(i=0; i<10; i++){
            if( i == 3) break;

            document.write(" i = " + i + " <br>");

        }


    </script>
</body>
</html>
```

i = 0
i = 1
i = 2

break_test.html - continue 예시

```
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
</head>
<body>
    <script>
        var i;
        for(i=0; i<10; i++){
            if( i == 3) continue;

            document.write(" i = " + i + " <br>");

        }


    </script>
</body>
</html>
```

```
i = 0
i = 1
i = 2
i = 4
i = 5
i = 6
i = 7
i = 8
i = 9
```

quiz1.html - 1에서 100까지의 홀수의 합 구하기 break사용

```
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
</head>
<body>
    <script>
        var i=0;
        var sum=0;
        while(true){
            i++;
            if( i % 2 == 1){
                sum += i;
```

```
            }
            if(i == 100){
                break;
            }
        }document.write(sum);
    </script>
</body>
</html>
```

```
quiz2.html - 1에서 100까지의 홀수의 합 구하기 continue사용

<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
</head>
<body>
    <script>
        var i=0;
        var sum=0;
        for(i = 0; i <= 100; i++){

            if( i % 2 == 0){
                continue;
            }
            sum += i;
        }document.write(sum);
    </script>
</body>
</html>
```

배열 array

변수는 1개의 값만 다룰 수 있다.

ex) 50명 학생들의 영어 점수 → 변수가 50개 필요 : 비효율적

효율적으로 해결하기 위해 배열 사용

배열 : 여러개의 값을 보관할 수 있는 특수 변수, 저장한 데이터를 꺼내서 사용할때는 index 값으로 접근한다.

```
기본형태

var 배열명 = [값1, 값2, 값3 .....]
```

```
array_test.html

<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
</head>
<body>
    <script>
    var arr = [1,2,3,4,5];
    document.write(arr[0] + "<br>");
    document.write(arr[1] + "<br>");
    document.write(arr[2] + "<br>");
    document.write(arr[3] + "<br>");
    document.write(arr[4] + "<br>");
    document.write(arr[5] + "<br>");


    </script>
</body>
</html>
```

```
1
2
3
4
5
undefined
```

```
array_test2.html - 배열을 for문으로 출력

<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
</head>
<body>
```

```
    <script>
    var arr = [1,2,3,4,5];

        for(var i = 0 ; i < arr.length; i++){
            document.write(arr[i] + "<br>");
        }

    </script>
</body>
</html>
```

quiz3.html - 배열 중 선물 뽑기

```
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
</head>
<body>
    <script>
        var gift = ["과일", "과자", "학용품", "용돈"]
        var a = prompt("1~4까지의 숫자를 입력하세요.");

        while (a > 4){
            a = prompt("다시입력하세여");
            var i = a-1;
        }
        document.write("당신은 " + gift[i] + "을 선물로 받게 됩니다.");

    </script>
</body>
</html>
```

array_test3.html - 자바스크립트에서의 배열 형태

```
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
</head>
<body>
    <script>
```

```
        var arr =[10, "dk", true];

        for(var i in arr){
            document.write("<br>");
            document.write(arr[i]);
        }
    </script>
</body>
</html>

// 자료형 상관없이 넣을 수 있고 출력도 가능하다.
// 값이 아닌 index가 나오는 개념이다.
```

```
10
dk
true
```

※ 자바에서는 for each 구문으로 배열에서 값을 하나씩 꺼낼 때 편리하게 사용했다.

자바스크립트에서는 for in 구문을 사용하여 반복문을 간단하게 처리할 수 있다.

Function 함수

함수는 프로그래밍의 격을 높여주는 기능이다.

함수덕분에 프로그래밍 기술이 급속도로 발전되었다.

효율적인 프로그래밍을 위한 출발점이 바로 함수의 사용이다.

Function은 기능을 뜻한다.

특별 프로그래밍에서는 함수가 클래스 안에 위치하면 매서드라고 부르고 클래스 밖에서 구현되면 함수라고 부르는 경향이 있다.

함수를 선언을 하고 사용을 위해 그 함수를 호출하게 된다.

기본적으로 코드의 반복작성을 피하기 위해 사용된다.

```
함수 선언의 기본 형태
```

```
function 함수이름(인수들){
  실행할 구문들;
}

함수 호출의 기본 형태

함수명(인수들);
```

```
fuction_test.html

<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
</head>
<body>
    <script>
    function greeting(){
        document.write("가입 완료<br>");
        document.write("다시 로그인 하세요<br>");
        document.write("방문을 환영합니다.<br>");
    }
    document.write("함수 호출로 출력 <br><br>");
    greeting();

    </script>
</body>
</html>
```

함수는 선언만으로는 결코 실행되지 않는다.

함수는 반드시 호출에 의해서만 실행된다.

호출은 다양한 방법으로 할 수 있다.

함수의 특징

1. 함수 선언은 function 으로 시작하여 뒤에 함수명을 기술하고 () 안에 인수를 기술하되 인수는 없을 수도 있다.
2. 함수는 함수를 호출했을때에만 실행이 되고 함수 호출은 “함수명()”로 한다.

3. 함수의 가장 기본적인 목적은 반복되는 소스를 줄이기 위함이다.

인수를 받아서 사용하는 함수

함수를 호출할 때 함수에 인수를 함께 넘겨서 각각 호출 할 때마다 다른 결과를 얻을 수 있게 된다.

```
function_test1.html

<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>

<!-- <script>
    function cal(op1, op2){
        document.write("op1 + op2 = " + (op1+op2) + "<br>");
        document.write("op1 - op2 = " + (op1-op2) + "<br>");
        document.write("op1 * op2 = " + (op1*op2) + "<br>");
        document.write("op1 / op2 = " + (op1/op2) + "<br>");
    }
    </script> -->
// 선언은 head에서 하고 호출은 body에서 해도 문제 없다.
</head>
<body>
    <script>

        function cal(op1, op2){
            document.write("op1 + op2 = " + (op1+op2) + "<br>");
            document.write("op1 - op2 = " + (op1-op2) + "<br>");
            document.write("op1 * op2 = " + (op1*op2) + "<br>");
            document.write("op1 / op2 = " + (op1/op2) + "<br>");
        }
        cal(10, 5);
        cal(20,10);
    </script>
</body>
</html>
```

결과 값을 반환하는 함수

함수 선언에서 return문을 사용하여 함수의 실행 결과 값을 함수를 호출한 그 위치로 돌려줄 수 있다.

```
function_test2.html

<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
</head>
<body>
    <script>
        function hap(op1, op2){
            return op1+op2;
        }
        function gop(op1, op2){
            var result = op1 * op2;
            return result;
        }

        document.write(hap(1,2) + "<br>");

        var r = gop(2,3)
        document.write(r + "<br>");
    </script>
</body>
</html>
```

※ 많이 사용되는 내장 함수

내장함수는 사용자가 직접 만들지 않아도 이미 사용할 수 있게끔 정의 해놓은 함수들이다.

자바스크립트는 다른 언어에 비해 내장함수가 많지 않은 편이다.

```
isNaN() : 숫자가 아니면 true 숫자이면 false를 반환.
parseInt() : 정수 문자를 정수로 변환한다. 정수처럼 보이지만 문자를 진짜 정수로 변환.
parseFloat() : 실수 문자를 실수로 변환. 실수처럼 보이지만 문자를 진짜 실수로 변환.
eval() : 문자열로 구성된 식을 계산하여 결과를 돌려준다. 식처럼 보이는 문자를 진짜 식으로 변환
```

```
function_test3.html
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
</head>
<body>
    <script>

        // isNaN 사용
        var num = "abc";
        if(isNaN(num)){
            document.write("num 문자를 기억합니다. <br>");
        } else {
            document.write("num은 숫자를 기억합니다.<br>")
        }

        var num1 = 123;
        if(isNaN(num1)){
            document.write("num1은 문자를 기억합니다.<br>");
        } else {
            document.write("num1은 숫자를 기억합니다.<br>")
        }

        // parseInt 사용
        var num2 = "2";
        document.write("<br> num2 + 1 = " + (parseInt(num2)+1));

        var num3 = "3.5";
        document.write("<br> num3+1 = " + (parseInt(num3)+1))


        // parseFloat 사용
        var num4 = "4.5";
        document.write("<br> num4+1 = " + (parseFloat(num4)+1));

        //eval 사용
        var num5 = "4+4+4";
        document.write("<br> num5 = " + eval(num5));
    </script>
</body>
</html>
```

전역 변수와 지역 변수

특히 함수를 사용할때는 변수의 생존기간이 아주 중요하다.

일반적인 변수는 프로그램이 실행되는 동안 메모리에 계속 생존하다가

프로그램이 종료되면 사라진다. 그렇지만 함수에서의 변수는 다르다.

함수 내부에서 선언된 변수는 함수의 실행이 끝날때 없어진다.
이런 변수를 지역변수 local variable 이라고 한다.

반면 함수 외부에서 선언된 변수는 프로그램 실행이 전부 끝나야 없어진므로
전역 변수 global variable 이라고 한다.

```
var_test.html

<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
</head>
<body>
    <script>
        function add() {
            var counter = 0;
            return counter += 1;
        }

        document.write(add() + "<br>");
        document.write(add() + "<br>");
        document.write(add() + "<br>");
    </script>
</body>
</html>

// add()함수를 호출할 때마다 새롭게 counter 변수가 0으로 초기화 되어 항상 같은 1을 반환하고 있다.
// 원래 의도는 호출할 때마다 하나씩 증가 시켜 그 값을 유지가 목적이었으나 의도한 것과는 다르게
// 동작하고 있다.
```

```
var_test.html

<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
```

```
</head>
<body>
    <script>

        var counter = 0;
        function add() {
           return counter += 1;
        }

        document.write(add() + "<br>");
        document.write(add() + "<br>");
        document.write(add() + "<br>");
    </script>
</body>
</html>

// 위처럼 counter 변수의 선언을 함수 밖에서 해야 프로그램이 종료되는 순간까지 계속 유지하게
// 됨으로 원하는 동작이 이루어 질 수 있다.
```

객체 Object

객체도 함수와 비슷하게 변형된 형식이지만 매우 중요하다.

자바를 비롯한 파이썬 C++ 등등 현재 대부분의 언어에서 사용되는 객체 지향 개념이다.

자바 스크립트도 객체 지향 언어 (Object Oriented Programming : OOP) 이다.

따라서 객체 지향에서는 모든 것이 객체로 취급된다. 숫자, 문자, 모든 데이터, 함수, 배열 전부 객체로 취급된다.

```
object.html - 객체에 접근하는 두가지 방법

<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
</head>
<body>
    <script>
        // 괄호가 없으면 프로퍼티 - 객체.프로퍼티
        // str객체의 속성을 알려준다.
        var str = "ABCDEF";
        document.write(str.length + "<br>"); //변수의 길이
        document.write("ABCDEF".length + "<br>"); //문자열의 길이
```

```
        // 둘 다 6으로 값은 같다.

        // 괄호가 있으면 매서드 - 객체.매서드()
        var num = 3.141592;
        document.write(num.toFixed(2) + "<br>");
        document.write(3.141592.toFixed(2) + "<br>");

    </script>
</body>
</html>


// 프로그램 상에서 조작이나 처리가 필요한 대상(객체)을 구분하고
// 그 대상들에 프로퍼티(속성) 매서드를 제공하여 프로그래머가
// 일일이 코드를 작성하지 않고 다양한 작업을 할수 있다.
```

문자열 객체

```
속성 property

  length : 문자열의 길이 속성을 알려준다.

메서드 method

  charAt() : 문자열 내에서 특정 위치의 문자를 반환

  concat() : 2개 이상의 문자열을 결합

  indexOf() : 문자열내의 첫번째 특정 문자의 위치값을 반환

  replace() : 특정 문자열을 다른 문자열로 대치

  split() : 하나의 문자열을 여러개의 문자열로 분할

  substr() : 특정 위치 부터 특정 개수의 문자열을 추출

  toLowerCase() : 문자열을 소문자로 변환

  toUpperCase() : 문자열을 대문자로 변환
```

```
property_method.html

<html lang="en">
<head>
    <meta charset="UTF-8">
```

```
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
</head>
<body>
    <script>
        var str = "abc def";

        // 스페이스바도 문자로 인식되기 때문에 7로 출력
        document.write(str.length + "<br>");

        document.write(str.charAt(1) + "<br>");

        document.write(str.replace("abc","XYZ") + "<br>");

        // space 기준으로 잘라라.
        document.write(str.split(" ") + "<br>");
        document.write(str.split("d") + "<br>");

        document.write(str.substr(1,6) + "<br>");

        document.write(str.toUpperCase(str) + "<br>");
    </Script>
</body>
</html>
```

배열 객체

length 배열의 길이를 반환

```
length : 배열의 길이를 반환
concat() : 2개 이상의 배열을 하나로 합침

indexOf() : 배열 내의 특정 요소에 인덱스 값을 구함

pop() : 배열의 마지막 요소 하나를 삭제

push() : 배열의 마지막에 요소를 하나 추가

reverse() : 배열 내 요소들을 역순으로 정렬

sort() : 배열 내 요소들의 순서를 기본 오름차순으로 정렬
```

```
array_method.html
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
</head>
<body>
    <script>
        var arr1 = ["Banana", "Lemon", "Apple"];
        var arr2 = ["119", "112", "114"];
        var arr3 = [10, 5, 1, 100, 80];

        document.write(arr1.length + "<br>");
        document.write(arr1.concat(arr2) + "<br>");
        document.write(arr2.indexOf("112") + "<br>");

        arr1.pop();
        document.write(arr1 + "<br>");

        arr1.push("Orange");
        document.write(arr1 + "<br>");

        document.write(arr1.reverse() + "<br>" );
        document.write(arr1.sort() + "<br>");

        document.write(arr1.sort().reverse() + "<br>");
        document.write(arr1.reverse(arr1.sort()));

            document.write(arr1 + "<br>");

        for(var i = arr1.length-1; i >= 0; i--){
            document.write(arr1[i]);
        }


    </script>
</body>
</html>
```