4장 함수

1. 함수의 정의와 호출

함수란 필요한 기능을 구현해 놓은 단위이다.

① 함수 정의

```
function 함수명() {
	실행문;
}
```

함수명은 변수명 만드는 방법과 동일하다. 함수명은 동사+목적어의 형식으로 만든다.

로그인 처리 함수 : function doLogin() { }

우편번호 검색 팝업창 : function openZipCodePopupWindow() { }

회원 검색 : function searchMember() { }

② 함수 호출

```
함수명();
```

function-test1.html

```
<!DOCTYPE html>
<html lang="ko">
	<head>
		<meta charset="utf-8" />
		<title>Function Test</title>
		<script type="text/javascript">
			// 함수의 선언
			function sayHello() {
				document.write("<h1>안녕하세요.</h1>");
				document.write("<h1>자바스크립트</h1>");
			}
			// 함수의 호출 (여러 번 재사용 할 수 있다.)
			sayHello();
			sayHello();
		</script>
	</head>
	<body></body>
```

```
</html>
```

2) 함수 매개변수

```
function 함수명(매개변수) {
	실행문;
}

함수명(매개변수);
```

function-test2.html

```
<!DOCTYPE html>
<html lang="ko">
	<head>
		<meta charset="utf-8" />
		<title>Function Test</title>
		<script type="text/javascript">
			// 파라미터를 갖는 함수의 선언
			function f(x) {
				var y = x + 1;
				document.write("<h1>" + y + "</h1>");
			}
			// 함수의 호출 --> 파라미터를 전달한다.
			f(2);
			f(5);
			f(10);
		</script>
	</head>
	<body></body>
</html>
```

매개변수명과 같은 변수명을 함수 안에 선언하면 중복 에러가 발생한다.

```
function f(x) {
    var x =100;   // 변수 중복 에러
}
```

① 다중 매개변수

```
function 함수명(매개변수1, 매개변수2, ...) {
	실행문;
```

```
}
함수명(매개변수1, 매개변수2, ...);
```

function-test3.html

```
<!DOCTYPE html>
<html lang="ko">
	<head>
		<meta charset="utf-8" />
		<title>Function Test</title>
		<script type="text/javascript">
			// 파라미터를 갖는 함수의 선언
			function f(x, y) {
				var z = x + y;
				document.write("<h1>" + z + "</h1>");
			}
			// 함수의 호출 --> 파라미터를 전달한다.
			f(2, 1);
			f(5, 7);
			f(10, 5);
		</script>
	</head>
	<body></body>
</html>
```

2. 리턴값 사용

① 반환되는 값 정의

```
function 함수명(매개변수1, 매개변수2, ...) {
	실행문;
	return 변수명;
}
변수명 = 함수명(매개변수1, 매개변수2, ...);
```

② 함수의 실행 중단

```
function 함수명(매개변수1, 매개변수2, ...) {
	if( 조건 ) {
		return 값;
```

```
    }
    실행문;
    return 변수명;    // 생략 가능
}
```

function-test4.html

```
<!DOCTYPE html>
<html lang="ko">
        <head>
                <meta charset="utf-8" />
                <title>Function Test</title>
                <script type="text/javascript">
                        // 파라미터를 갖는 함수의 선언
                        function f(x, y) {
                                var z = x + y;

                                // 값이 10보다 작을 경우, 실행중단
                                if (z < 10) {
                                        return;  // 값이 없으면 실행 종료
                                }
                                return z;
                        }

                        // 함수의 호출 --> 파라미터를 전달한다.
                        var a = f(2, 1);    // undefined 출력
                        var b = f(5, 7);
                        var c = f(10, 5);

                        document.write("<h1>" + a + "</h1>");
                        document.write("<h1>" + b + "</h1>");
                        document.write("<h1>" + c + "</h1>");
                </script>
        </head>
        <body></body>
</html>
```

3. 다른 함수의 호출

```
function f1(x) {
    return x+1;
}
function f2(x) {
    return f1(x) * 2;
}
```

매개 변수는 함수 안에서만 유효하기 때문에 서로 다른 함수에는 매개변수 이름이 동일해도 에러가 발생하지 않는다.

function-test5.html

```
<!DOCTYPE html>
<html lang="ko">
	<head>
		<meta charset="utf-8" />
		<title>Function Test</title>
		<script type="text/javascript">
			function sum(x, y) {
				var z = x + y;
				return z;
			}

			function printResult(x, y) {
				// (우변) sum 함수를 호출하여 결과를 리턴받았다.
				// (좌변) 리턴받은 결과를 result에 대입하였다.
				var result = sum(x, y);
				document.write("<h1>" + result + "</h1>");
			}

			printResult(7, 10);
		</script>
	</head>
	<body></body>
</html>
```

5장 Javascript 내장 함수

웹 브라우저 개발사에서 미리 정의하여 놓은 함수로 개발자가 자유롭게 호출하여 사용할 수 있다.

1. 주요 내장 함수

내장 함수	함수의 기능
eval(문자열)	주어진 문자열을 수식으로 변환해서 리턴한다.
Number(문자열)	주어진 문자열를 숫자로 변환해서 리턴한다.
parseInt(문자열)	주어진 문자열를 숫자로 변환해서 리턴한다.
isNaN(문자열)	주어진 문자열이 숫자형식이 아니면 true, 숫자형식이면 false를 리턴한다.
alert(문자열)	주어진 문자열을 확인 대화창으로 표시한다.
confirm(문자열)	주어진 문자열이 표시되는 확인, 취소 대화창으로 표시한다.
prompt(문자열)	주어진 문자열이 제목으로 값을 입력받기 위한 대화창이 표시된다.

2. 문자열을 수식으로 변경

eval-test.html

```
<!DOCTYPE html>
<html lang="ko">
	<head>
		<meta charset="utf-8" />
		<title>eval Test</title>
		<script type="text/javascript">
			var str = "100 + 1";
			document.write("<h1>" + str + "</h1>");
			var result = eval(str);
			document.write("<h1>" + result + "</h1>");
		</script>
	</head>
	<body></body>
</html>
```

3. 문자열을 숫자로 변경

Number-test1.html

```
<!DOCTYPE html>
<html lang="ko">
	<head>
```

```
		<meta charset="utf-8" />
		<title>Number Test</title>
		<script type="text/javascript">
			var value = "100";
			var num1 = Number(value);
			var num2 = parseInt(value);

			// 글자 + 숫자 --> 글자 + 글자 --> 모두 합친다.
			var r1 = value + 1;
			// 숫자 + 숫자 --> 덧샘을 한다.
			var r2 = num1 + 1;
			var r3 = num2 + 1;
			document.write("<h1>r1=" + r1 + "</h1>");
			document.write("<h1>r2=" + r2 + "</h1>");
			document.write("<h1>r3=" + r3 + "</h1>");
		</script>
	</head>
	<body></body>
</html>
```

Number-test1.html

```
<!DOCTYPE html>
<html lang="ko">
	<head>
		<meta charset="utf-8" />
		<title>Number Test</title>
		<script type="text/javascript">
			var value = "100.6";

			// 문자열 그대로 100.6 이라는 실수값을 리턴한다.
			var num1 = Number(value);
			// 반올림이 아니라 내림 처리된다.
			var num2 = parseInt(value);

			var r1 = num1 + 1;
			var r2 = num2 + 1;

			document.write("<h1>r1=" + r1 + "</h1>");
			document.write("<h1>r2=" + r2 + "</h1>");
```

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

Number() 함수와 parseInt()함수는 정수형태의 문자열일 경우에는 같은 결과를 반환하지만 실수형태의 문자열은 서로 다른 결과를 반환한다.

4. 숫자인지 검사

Number() 함수와 parseInt()함수는 매개 변수로 전달된 문자열이 숫자 형태가 아닐 경우 NaN(Not a Number)를 반환한다.

isNaN-test1.html

```
<!DOCTYPE html>
<html lang="ko">
	<head>
		<meta charset="utf-8" />
		<title>isNaN Test</title>
		<script type="text/javascript">
			// 결과 : "NaN"
			var a = Number("a");
			document.write("<h1>a=" + a + "</h1>");

			// "NaN" + 1 지만 "NaN1"이 아닌 "NaN"이 된다.
			var a_sum = a + 1;
			document.write("<h1>a_sum=" + a_sum + "</h1>");

			// "NaN" + "str"이므로 "NaNstr"이 된다.
			var a_str = a + "str";
			document.write("<h1>a_str=" + a_str + "</h1>");
		</script>
	</head>
	<body>
	</body>
</html>
```

isNaN-test2.html

```
<!DOCTYPE html>
```

```
<html lang="ko">
	<head>
		<meta charset="utf-8" />
		<title>isNaN Test</title>
		<script type="text/javascript">
			var v1 = "가나다라";
			var v2 = "123456";

			var is_num1 = !isNaN(v1);
			var is_num2 = !isNaN(v2);

			if (is_num1) {
				document.write("<h1>`" + v1 + "`은(는) 숫자 입니다.</h1>");
			} else {
				document.write("<h1>`" + v1 + "`은(는) 숫자가 아닙니다.</h1>");
			}

			if (is_num2) {
				document.write("<h1>`" + v2 + "`은(는) 숫자 입니다.</h1>");
			} else {
				document.write("<h1>`" + v2 + "`은(는) 숫자가 아닙니다.</h1>");
			}
		</script>
	</head>
	<body></body>
</html>
```

isNaN-test3.html

```
<!DOCTYPE html>
<html lang="ko">
	<head>
		<meta charset="utf-8" />
		<title>isNaN Test</title>
		<script type="text/javascript">
			function sumInt(x, y) {
				var isStrX = !isNaN(x);
				var isStrY = !isNaN(y);

			// 둘 중 하나라고 숫자 모양이 아니라면, 실행을 중단하고 메시지를 리턴
```

```
                        if (!isStrX || !isStrY) {
                                return "숫자 형태이거나 숫자값이 전달되어야 합니다.";
                        }
                        var result = parseInt(x) + parseInt(y);
                        return result;
                }

                var sum1 = sumInt("a", 1);
                var sum2 = sumInt("1", "2");
                var sum3 = sumInt(100, "5.1");
                var sum4 = sumInt(-1, "-100");

                document.write("<h1>" + sum1 + "</h1>");
                document.write("<h1>" + sum2 + "</h1>");
                document.write("<h1>" + sum3 + "</h1>");
                document.write("<h1>" + sum4 + "</h1>");
        </script>
    </head>
    <body>
    </body>
</html>
```

5. 대화 상자

① 사용자에게 메시지 표시

```
alert("대화 상자에 표시할 메시지");
```

② 사용자의 결정 결과를 반환

```
var is_ok = confirm("표시할 메시지");
if( is_ok ) {
    true 일 때 실행문;
} else {
    false 일 때 실행문;
}
```

③ 사용자가 입력한 값을 반환

```
var input = prompt("표시할 메시지");
document.write(input);
```

messageBox-test.html

```
<!DOCTYPE html>
<html lang="ko">
	<head>
		<meta charset="utf-8" />
		<title>messageBox Test</title>
		<script type="text/javascript">
			var user_id = prompt("아이디를 입력하세요.", "");

			// 빈 문자열이 아닌 경우에만 실행
			if (user_id != "") {
				alert("안녕하세요. " + user_id + "님");

				var result = confirm("정말 로그아웃 하시겠습니까?");

				if (result) {
					alert("로그아웃 되셨습니다.");
				} else {
					alert("로그아웃이 취소되었습니다.");
				}
			}
		</script>
	</head>
	<body></body>
</html>
```

6장 이벤트 처리

1. 이벤트 개요

이벤트란 사용자가 웹 페이지에서 행한 일련의 동작.

① 이벤트의 종류

이벤트 이름	설 명
onAbort	이미지를 로딩하는 작업이 사용자의 행동으로 취소되었을 때
onBlur	문서나 윈도우, 폼 요소에서 현재 입력 포커스가 사라질 때
onChange	텍스트 필드나 텍스트 영역, 파일 업로드, 선택 항목이 변경되어 현재 입력 포커스가 사라질 때
onClick	링크나 클라이언트 쪽 이미지 맵 영역, 폼 요소가 클릭되었을 때
onDbClick	링크나 클라이언트 쪽 이미지 맵 영역, 폼 요소가 더블 클릭되었을 때
onDragDrop	드래그 된 객체가 윈도우의 영역에 드롭되었을 때
onError	이미지나 문서를 로딩하는 동안 에러가 발생 했을 때
onFocus	문서나 폼 요소에 입력 포커스가 놓였을 때
onKeyDown	키를 누를 때
onKeyPress	키를 눌렀다 놓았을 때
onKeyUp	키를 놓았을 때
onLoad	이미지나 문서가 로딩될 때
onMouseDown	마우스 버튼을 누를 때
onMouseMove	마우스를 이동했을 때
onMouseOver	문서의 태그영역에 마우스 커서가 들어갈 때
onMouseOut	문서의 태그영역에 마우스 커서가 벗어날 때
onMouseUp	마우스 버튼을 놓았을 때
onMove	사용자가 윈도우에서 이동할 때
onReset	폼의 리셋 버튼을 클릭하여 리셋시킬 때

이벤트 이름은 전부 소문자로 쓴다.

② 이벤트의 사용

```
function sayHello() {
    alert("Hello Event");
}
```

```
<input type="botton" value="결과값" 이벤트이름="sayHello()" />
```

2. HTML 태그 요소를 획득

제어할 특정 태그 요소를 지정하기 위해서는 Javascript 구문과 HTML 태그의 id 속성값을 연결해야 한다.

```
document.getElementById("아이디값");
```

```
<span id="value"> 요소 내용 </span>

var myTag = document.getElementById("value");
```

① HTML 태그 요소 제어

```
var myTag = document.getElementById("value");
myTag.innerHTML = "동적으로 적용될 새로운 요소 내용";
```

```
document.getElementById("value").innerHTML = "동적으로 적용될 새로운 요소 내용";
document.getElementById("value").innerHTML = "";
var tagContent = document.getElementById("value").innerHTML;
```

3. 이벤트 활용

① onclick 이벤트

event-test1.html

```
<!DOCTYPE html>
<html lang="ko">
	<head>
		<meta charset="utf-8" />
		<title>Event Test</title>
		<script type="text/javascript">
			/** printResult() 함수 안에서 호출된다. */
			// 두 수를 더한 값을 리턴하여 주는 함수
			function sum(x, y) {
				var z = x + y;
				return z;
			}

			/** input type="button"이 클릭되었을 때 호출된다. */
			function printResult() {
```

```
                        // sum 함수를 호출하여 10+50의 결과를 리턴받는다.
                        var result = sum(10, 50);

                        // document.getElementById("question")
                        // --> id가 "question"인 태그요소
                        var mytag = document.getElementById("question");

                        // ~의 안에 result값을 넣는다. --> ~.innerHTML = result;
                        // 이 요소의 안에는 "?"가 들어 있지만, 새롭게 지정된 result값이
                        // "?"를 지우고 자신이 그 자리를 차지한다.
                        mytag.innerHTML = result;
                    }
                </script>
            </head>
            <body>
                <h1>10 + 50 = <span id="question">?</span></h1>
                <!-- 클릭했을 때, printResult() 함수를 호출한다. -->
                <input type="button" value="결과보기" onclick="printResult()" />
            </body>
</html>
```

② onmouseover 이벤트와 onmouseout 이벤트

event-test2.html

```
<!DOCTYPE html>
<html lang="ko">
        <head>
                <meta charset="utf-8" />
                <title>Event Test</title>
                <script type="text/javascript">
                        /** showResult() 함수 안에서 호출된다. */
                        // 두 수를 더한 값을 리턴하여 주는 함수
                        function sum(x, y) {
                                var z = x + y;
                                return z;
                        }

                        /** (1) id가 question인 요소안에 10+50의 결과를 지정한다. */
                        function showResult() {
```

```
                        var result = sum(10, 50);
                        var red_result = "<font color='red'>" + result + "</font>";
                        document.getElementById("question").innerHTML = red_result;
                    }

                    /** (2) id가 question인 요소안에 "?"를 지정한다. */
                    function hideResult() {
                        document.getElementById("question").innerHTML = "?";
                    }
                </script>
            </head>
            <body>
                <!-- (3) <h1> 태그 위에 마우스를 올렸을 때, showResult() 호출 -->
                <!-- (4) <h1> 태그 위에 마우스를 내렸을 때, hideResult() 호출 -->
                <h1 onmouseover="showResult()" onmouseout="hideResult()">10 + 50 = <span
id="question">?</span></h1>
                <p>결과를 보시려면 수식 위에 마우스를 올리세요.</p>
            </body>
</html>
```

③ onload 이벤트

event-test3.html

```
<!DOCTYPE html>
<html lang="ko">
    <head>
        <meta charset="utf-8" />
        <title>Event Test</title>
        <script type="text/javascript">
            function insertQuestion() {
                // 내용을 입력받는 대화상자
                var q = prompt("사칙연산의 수식을 입력하세요. (예: 100+50)", "");

                // 입력받은 문자열을 수식으로 변환한다.(내장함수 사용)
                var answer = eval(q);

                // 사용자가 입력한 질문(q)와 계산결과 (answer)를
                // 하나의 문자열로 연결한다.
                var result = q + "=" + answer;
```

```
                        // 출력
                        document.getElementById("question").innerHTML = result;
                    }
                </script>
        </head>
        <!-- 페이지가 처음 열려서 웹 브라우저에게 HTML태그의 인식이 완료된 후,
                자동으로 insertQuestion() 함수가 호출되도록 함 -->
        <body onload="insertQuestion()">
                <h1 id="question"></h1>
        </body>
</html>
```

4. Javascript 소스의 실행 시점

Javascript는 특정 HTML 태그 요소를 제어하는 기능을 한다. 웹 브라우저는 HTML 파일을 한 줄씩 읽어서 해석한다. Javascript 코드가 HTML 문서 안에서 실행되는 시점을 알아본다.

whyonload-test.html

```
<!DOCTYPE html>
<html lang="ko">
        <head>
                <meta charset="utf-8" />
                <title>Event Test</title>
                <script type="text/javascript">
                /** Javascript 구문의 실행 시점
                -------------------------------------
                1) HTML페이지는 웹 브라우저에 의해서 맨 처음부터 한 줄씩 읽혀서 실행된다.
                2) doPrint()라는 함수를 호출하라는 명령어를 만나면 그 즉시, 해당 함수를 호출한다.
                3) doPrint() 함수 안에서는 id속성값이 "myid"인 태그요소를 제어하려고 한다.
                4) 하지만 웹 브라우저에게는 아직 HTML태그가 읽혀진것이 아닌 시점이기 때문에 그 요소를 찾지 못하고 에러가 난다.
                5) doPrint() 함수가 온전하게 실행되기 위해서는 이 함수에서 제어하려는 "myid"요소가 웹 브라우저에게 인식된 이후에 호출되어야 한다.

                결론 : 제어 하려는 태그가 인식된 이후에 Javascript 함수를 호출해야 한다는 제약은, Javascript소스와 HTML태그가 반드시 순서를 지켜가면서, 섞여서 코딩해야 한다는 제약을 가져온다. (스파게티 소스) onload 이벤트는 이러한 제약을 해결하고 Javascript 코드와 HTML 태그를 분리할 수 있게 해 준다.
```

```
		*/
			function doPrint(num) {
				var msg = "doPrint()가 " + num + "회 실행되었습니다.";
				alert(msg);
				document.getElementById("myid").innerHTML = msg;
			}

			// (1)~(4)
			doPrint(1);
		</script>
	</head>
	<body>
		<script type="text/javascript">
			// (1)~(4)
			doPrint(2);
		</script>

		<h1 id="myid">안녕하세요.</h1>

		<script type="text/javascript">
			// (5)
			doPrint(3);
		</script>
	</body>
</html>
```

HTML 태그 요소를 제어하기 위한 Javascript 구문이 실행되는 시점은 제어를 위한 요소가 웹 브라우저에서 읽힌 후이다. 때문에 소스 코드를 한 줄씩 읽어 들이는 웹 브라우저의 특성상 Javascript 코드는 HTML 태그 이후에 기술되어야 한다.

효율적인 관리를 위해 HTML 태그와 Javascript 코드를 각기 다른 블록으로 분리해야 한다. HTML 태그가 모두 웹 브라우저에서 읽혀지기 전까지 실행을 대기해야 하므로 <body> 태그에 onload 이벤트로 Javascript 함수를 연결해야 정상적으로 실행되는 경우가 있다.

좀더 나은 방법은 Javascript 소스를 외부 파일로 만들어서 연결하는 것이 좋다.

5. HTML 태그의 id 속성 Javascript에서 접근하기

Javascript에서는 HTML 태그의 id값을 document.getElementById("id값")의 형식으로 찾는다. 기본으로 하나의 웹 문서에는 id 값이 중복되어는 안되지만 중복된 id값이 존재할 경우에 Javascript는 가장 첫 번째로 기술된 요소를 제어한다.

id-test.html

```
<!DOCTYPE html>
<html lang="ko">
	<head>
		<meta charset="utf-8" />
		<title>Id Test</title>
		<style type="text/css">
			/** 스타일시트는 중복 아이디에도 적용이 된다. */
			#myid {
				background: #ff0;
				font-size: 20px;
				font-weight: bold;
				color: #00f;
			}
		</style>

		<script type="text/javascript">
			function doPrint() {
				// 자바스크립트는 중복아이디를 인식하지 못하고,
				// 가장 먼저 나타나는 id값만 인식한다.
			document.getElementById("myid").innerHTML = "스크립트가 실행되었습니다.";
			}
		</script>
	</head>
	<body>
		<h1 id="myid">안녕하세요.</h1>
		<h1 id="myid">자바스크립트 입니다.</h1>
		<h1 id="myid">강의 참 재미있죠?</h1>
		<a href="#" onclick="doPrint()">자바스크립트 함수 호출하기</a>
	</body>
</html>
```

위 예제에서 Javascript는 동일한 id값을 갖는 요소들 중에서 가장 먼저 등장하는 첫 번째 요소에만 메시지를 출력한다.

CSS에서는 HTML 태그에 중복된 id값이 존재하는 것을 허용하지만, Javascript에서는 중복된 id값은 의미가 없다. 따라서 HTML 태그의 id값은 유일한 식별자로 정한다.