■ 함수

● 명령문(코드)의 집합

- 형태 1 (Function Declarations)

```
<script>
    function myFunc() {
    }
</script>
```

- 형태 2 (Function Expressions)

```
<script>
    var myFunc = function() {
    }
</script>
```

■ 함수

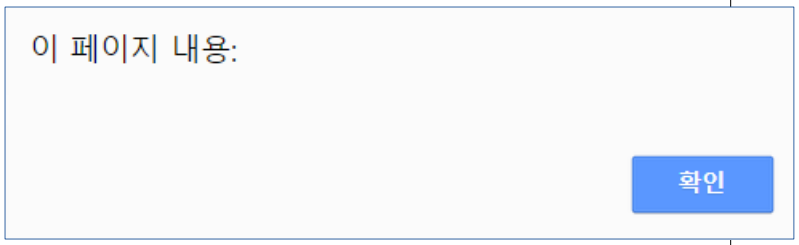
● 함수 사용 (호출) - 1

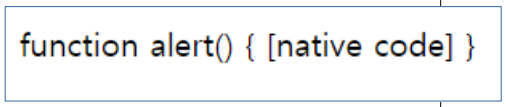
- 괄호 () 를 사용하여 함수 호출

```
<script>
    var func = alert;

    // 함수 내부 코드 출력
    document.write(func);

    // 함수 실행
    func();
</script>
```





■ 함수

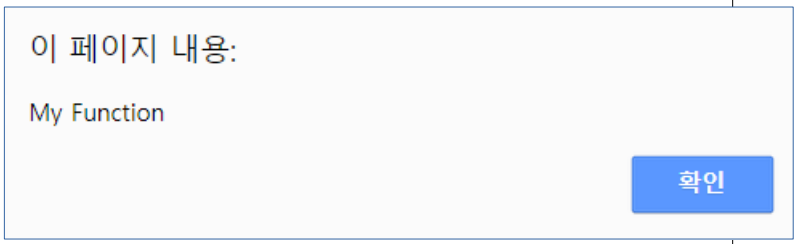
● 함수 사용 (호출) – 2

– 괄호 () 를 사용하여 함수 호출

```
<script>
    var func = function() {
        alert('My Function');
    };

    document.write(func);

    func();
</script>
```



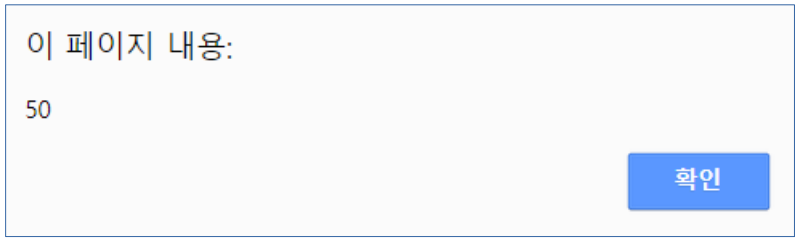

```
function () { alert('My Function'); }
```

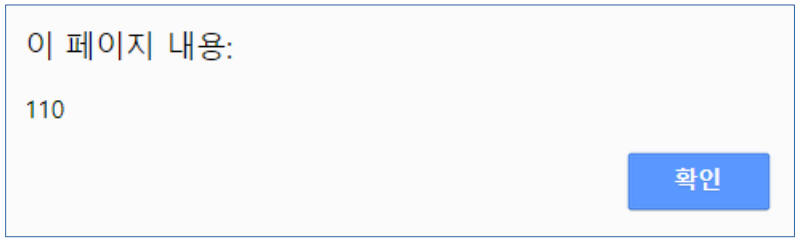
■ 함수

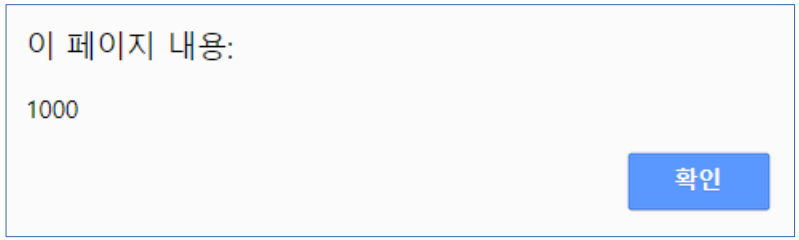
● 전역 / 지역 변수

- 함수 안에서 선언된 변수는 지역 변수로 인식
- 선언되지 않은 변수를 사용하면 전역 변수로 자동 생성 후 사용

```
<script>
    var i = 100;
    function funcA() {
        var i = 50;
        alert(i); // 지역변수 50
    }
    function funcB() {
        i = i + 10;
        alert(i); // 전역변수 110
    }
    function funcC() {
        j = 1000; // 선언하지 않은 변수는 전역변수로 취급
        alert(j); // 전역변수 1000
    }
    funcA();
    funcB();
    funcC();
    alert(j); // 전역변수 1000
</script>
```







■ 함수

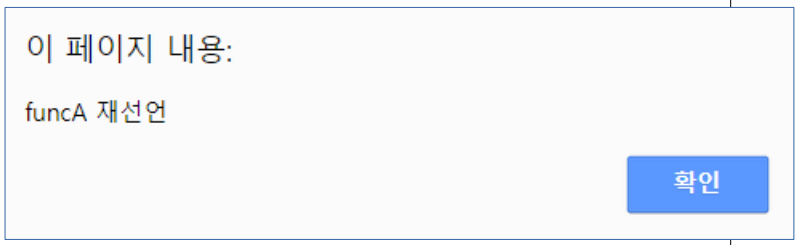
● 함수 중복 선언 – 1

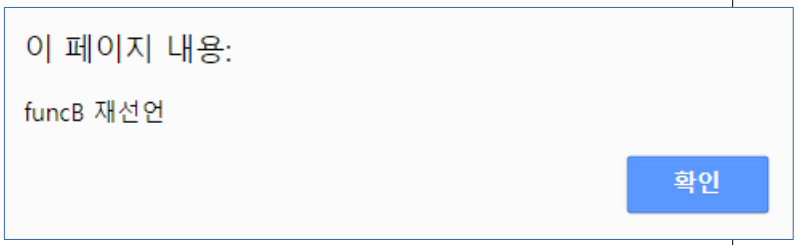
– 최종 선언된 함수가 동작

```
<script>
    function funcA() {
        alert('funcA');
    }
    function funcA() {
        alert('funcA 재선언');
    }

    var funcB = function() {
        alert('funcB');
    };
    var funcB = function() {
        alert('funcB 재선언');
    };

    funcA();
    funcB();
</script>
```





■ 함수

● 함수 중복 선언 – 2

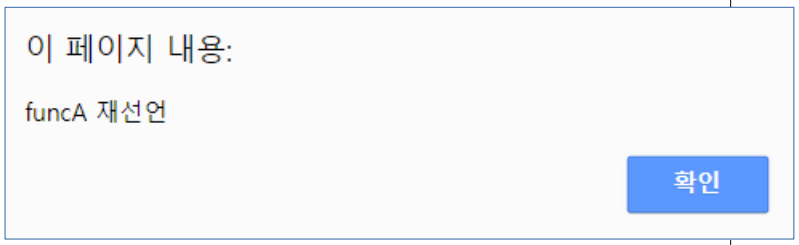
– 함수 선언 방식과 함수 표현 방식을 같이 사용하면서 중복되는 경우

• 함수 표현 방식으로 작성된 코드 동작

```
<script>
    var funcA = function() {
        alert('funcA 재선언');
    };

    function funcA() {
        alert('funcA');
    }

    funcA();
</script>
```



■ 함수

● Function Declarations vs Function Expressions

– 함수 선언 방식은 코드 위치가 상관없지만

함수 표현 방식은 사용하고자 하는 코드보다 항상 먼저 작성

```
<script>
    funcA(); // 정상 동작
    funcB(); // 오류

    function funcA() {
        alert('funcA');
    }

    var funcB = function() {
        alert('funcB');
    };
</script>
```

■ 함수

● Function Declarations vs Function Expressions

- 아래 코드 출력 결과는?

```
<script>
    funcA();

    var funcA = function() {
        function funcB() {
            console.log('java');
        }

        return funcB();

        function funcB() {
            console.log('script');
        }
    };
</script>
```

■ 함수

● Function Declarations vs Function Expressions

- 아래 코드 출력 결과는?

```
<script>
    funcA();

    function funcA() {
        var funcB = function() {
            console.log('java');
        }

        return funcB();

        var funcB = function() {
            console.log('script');
        }
    };
</script>
```

■ return

- 함수가 실행되는 도중에 함수가 호출된 위치로 복귀
- 함수 내의 코드로 작업한 결과를 반환

```
<script>
    function sum(num) {
        var sum = 0;
        for(var i = 1; i <= num; i++) {
            sum += i;
        }
        return sum;
    }

    var t = sum(10);
    document.write('1 ~ 10 까지의 합 : ' + t);
</script>
```

1 ~ 10 까지의 합 : 55

■ return

● 반환할 값을 지정하지 않으면 호출된 위치로 복귀되는 현상만 발생

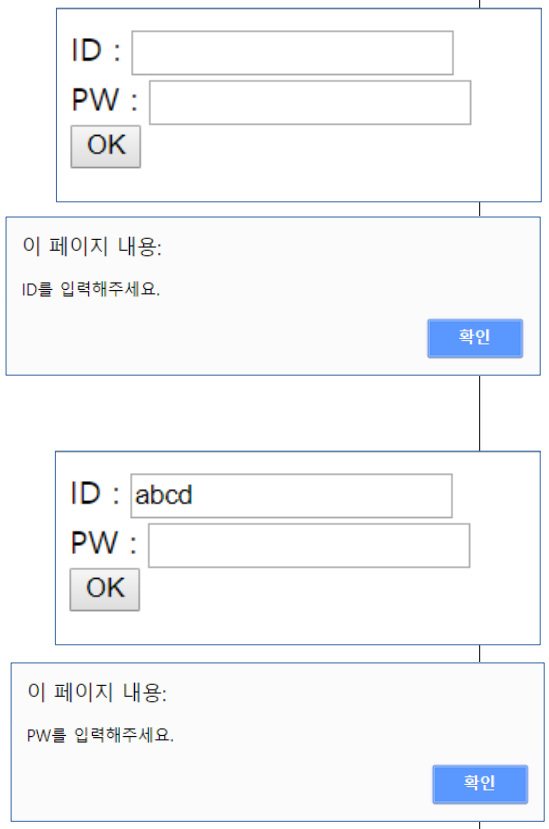
– 함수가 실행되는 도중에 더 이상 진행될 필요가 없는 경우

```
<script>
    function check() {
        var id = document.forms[0].user_id.value;
        var pw = document.forms[0].user_pw.value;

        if(id == '') {
            alert('ID를 입력해주세요.');
            return;
        }

        if(pw == '') {
            alert('PW를 입력해주세요.');
            return;
        }
    }
</script>

<form>
    ID : <input type='text' name='user_id'><br>
    PW : <input type='text' name='user_pw'><br>
    <input type='button' value='OK' onclick='check()'>
</form>
```



■ return

● 함수 반환

- 함수가 실행되기 전 원형코드 상태로 반환
- 함수를 호출하면서 ()를 붙여줘야 실행

```
<script>
    function funcA() {
        return function() {
            alert("함수A");
        }(); // 마지막에 ()를 붙여줘야 실행
    }

    funcA();
    // 함수 return 에서 ()를 붙이지 않고 funcA()(); 괄호 두번 호출도 가능
</script>
```

■ 가변인자

- 자바스크립트에서는 오버로딩이 없음
- 함수를 호출하면서 원하는 개수의 인자를 넘겨도 가능

```
<script>
    function func() {
        document.write("넘겨받은 인자의 개수 : " + arguments.length);
        for(var i = 0; i < arguments.length; i++) {
            document.write("<br>" + i + "번 : " + arguments[i]);
        }
        document.write("<br>==========<br>");
    }
    func(10);
    func("a", "b", "c");
    func();
</script>
```

■ 클로저 (Closure)

- 지역 변수가 사라지지 않고 계속 기억되는 현상
- 저장되어 있는 값을 임의로 수정하지 못하게 하기 위해서

- 온라인 시험 부정행위 확인
  - 전역변수를 사용해서 횟수 지정

```
<script>
    var count = 5;

    function decreaseCount() {
        count--;
        document.body.innerHTML = '남은횟수 : ' + count;
        if(count == 0) {
            alert('부정행위로 시험을 종료합니다.');
        }
    }
</script>
<body onblur='decreaseCount()'></body>
```

■ 클로저 (Closure)

● 온라인 시험 부정행위 확인

- 지역변수를 사용해서 횟수 지정

```
<script>
    function decreaseCount() {
        var count = 5;

        count--;
        document.body.innerHTML = '남은횟수 : ' + count;
        if(count == 0) {
            alert('부정행위로 시험을 종료합니다.');
        }
    }
</script>
<body onblur='decreaseCount()'></body>
```

■ 클로저 (Closure)

● 온라인 시험 부정행위 확인

- 클로저 사용

```
<script>
    function getCount() {
        var count = 5;

        return function() {
            return count--;
        }
    }

    var nowCount = getCount();

    function decreaseCount() {
        var check = nowCount();
        document.body.innerHTML = '남은횟수 : ' + check;
        if(check == 0) {
            alert('부정행위로 시험을 종료합니다.');
        }
    }
</script>
<body onblur='decreaseCount()'></body>
```

■ 클로저 (Closure)

● 클로저 사용 – 1

```
<script>
    function outerFunction(num) {
        var name = "a";

        return function() {
            name = name + num;
            num++;
            return name;
        };
    }
    var result1 = outerFunction(1);
    document.write(result1() + "<br>");
    document.write(result1() + "<br>");
    document.write(result1() + "<br>");
    document.write(result1() + "<br>");
    document.write(result1() + "<br>");
    document.write(result1() + "<br>");
</script>
```

a1
a12
a123
a1234
a12345
a123456

■ 클로저 (Closure)

● 클로저 사용 – 2

```
<script>
    function sequence() {
        var seq = 0;

        return function() {
            return ++seq;
        };
    }
    var seq = sequence();
    document.write(seq() + "<br>");
    document.write(seq() + "<br>");
    document.write(seq() + "<br>");
</script>
```

1
2
3

■ 클로저 (Closure)

● 클로저 사용 – 3

```
<script>
    function foo(x) {
        return function (y) {
            document.write(x + " : "+ y + "<br>");
        }
    }

    var bar1 = foo("과목");
    bar1("국어");
    bar1("영어");
    bar1("수학");

    var bar2 = foo("성적");
    bar2(100);
    bar2(90);
    bar2(80);
</script>
```

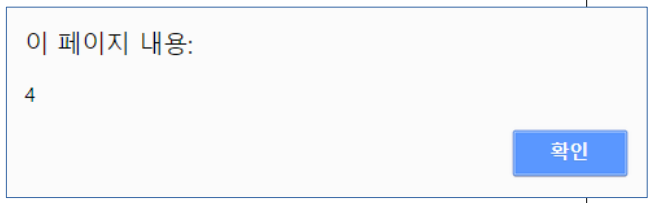
과목 : 국어
과목 : 영어
과목 : 수학
성적 : 100
성적 : 90
성적 : 80

■ 클로저 (Closure)

● 클로저 사용 – 4 (오류)

```
<script>
    function fnOnLoad() {
        for(var i = 1; i <= 3; i++) {
            document.getElementById("btn" + i).addEventListener("click", function() {
                alert(i);
            });
        }
    }
</script>

<body onload="fnOnLoad()">
    <input type="button" id="btn1" value="버튼1">
    <input type="button" id="btn2" value="버튼2">
    <input type="button" id="btn3" value="버튼3">
</body>
```



■ 클로저 (Closure)

● 클로저 사용 – 4 (오류 수정)

```
<script>
    function fnOnLoad() {
        for(var i = 1; i <= 3; i++) {
            (function(j) {
                document.getElementById("btn" + j).addEventListener("click", function() {
                    alert(j);
                });
            })(i);
        }
    }
</script>

<body onload="fnOnLoad()">
     <input type="button" id="btn1" value="버튼1">
     <input type="button" id="btn2" value="버튼2">
     <input type="button" id="btn3" value="버튼3">
</body>
```

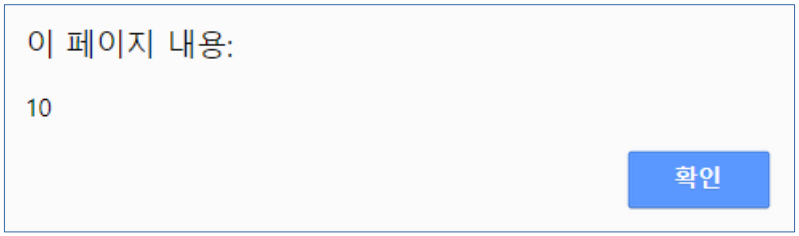
■ 내장함수

● eval()

- 문자열을 코드로 변환

```
<script>
    var code = '';
    code += 'var num = 10;';
    code += 'alert(num);';

    eval(code);
</script>
```



■ 내장함수

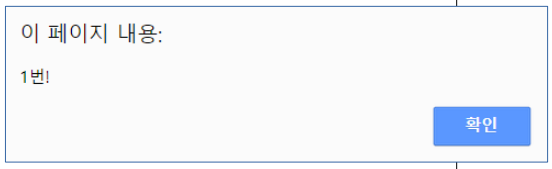
● eval()

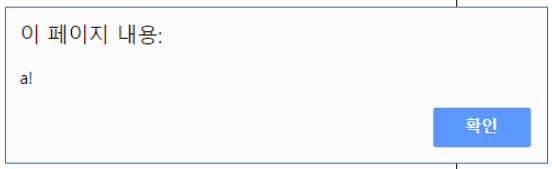
– 문자열을 코드로 변환

```
<script>
    function run() {
        var num = document.getElementById("num").value;
        eval("do" + num + "();");
    }

    function do1() { alert("1번!"); }
    function do2() { alert("2번!"); }
    function doa() { alert("a!"); }
</script>

<body>
    <input type="text" id="num">
    <input type="button" id="btn" value="실행" onclick="run()">
</body>
```



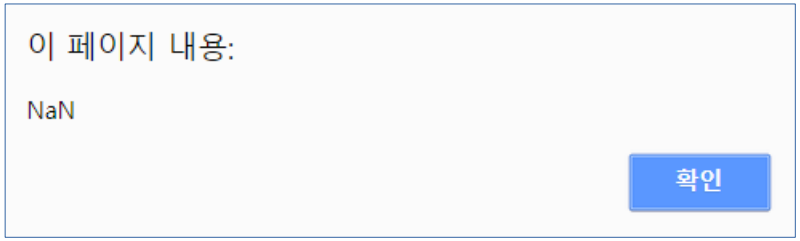


■ 내장함수

● isNaN() : Not a Number

- 숫자 판별

```
<script>
    var num = 10 / '가';
    alert(num);
</script>
```



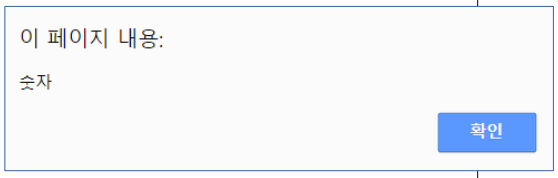
■ 내장함수

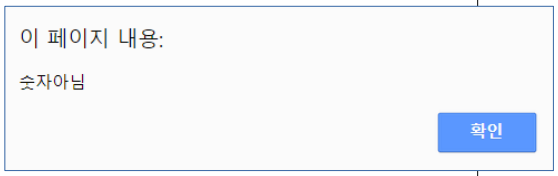
● isNaN() : Not a Number

- 숫자 판별

```
<script>
    function run() {
        var num = document.getElementById("num").value;
        var isNum = !isNaN(num);
        if(isNum) {
            alert("숫자");
        } else {
            alert("숫자아님");
        }
    }
</script>

<body>
    <input type="text" id="num">
    <input type="button" id="btn" value="실행" onclick="run()">
</body>
```





■ 내장함수

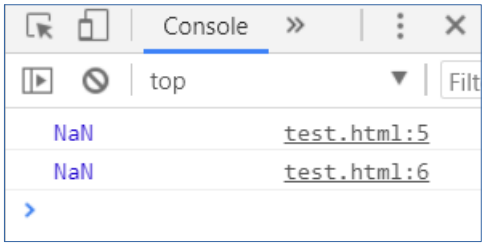
● parseInt() / parseFloat()

- 문자열을 숫자로 변환

- Number() 사용시

```
<script>
    var won = Number('1000원');
    var dollar = Number('1.5$');

    console.log(won);
    console.log(dollar);
</script>
```



- parseInt(), parseFloat() 사용시

```
<script>
    var won = parseInt('1000원');
    var dollar = parseFloat('1.5$');

    console.log(won);
    console.log(dollar);
</script>
```