



Chapter 04

함수 (functions)



함수

함수의 정의

■ 함수(functions)

- 반복적으로 실행되는 구문들을 미리 정의하여 호출을 통해 수행하는 방법
 - 프로그램의 구조화와 모듈화
 - 코드의 재사용
 - 유지보수의 용이성
- 함수의 구분
 - 내장 함수
 - 자바스크립트 엔진에 포함되어 있어 자바스크립트가 기본적으로 제공하는 함수
 - 별도의 선언 없이 호출만 하면 사용 가능
 - 사용자 정의 함수
 - 사용자가 필요에 의해 직접 함수를 작성하는 함수를 의미
 - 익명 함수와 선언적 함수가 있음

기본 함수 정의문

■ 선언 함수

- 이름을 가지는 함수를 의미
- 선언 함수의 생성

```
function 함수명( ) {  
  
    함수의 실행 문장들;  
}
```

```
<script type="text/javascript">  
    function myFunc( ) {  
        var output = prompt("숫자를 입력하세요.", "숫자");  
        alert(output);  
    }  
</script>
```

- 선언 함수의 실행
 - 함수의 이름을 호출함으로써 실행

```
함수이름();
```

```
myFunc();
```

기본 함수 정의문

4-function-name-1.htm

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="UTF-8">

  <script type="text/javascript">

    function myFunc() {

      document.write("hello", "<br>");
      document.write("welcome", "<br>");

    }

    myFunc();
    myFunc();

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

```
hello
welcome
hello
welcome
```

기본 함수 정의문

4-function-name-2.htm (p. 134)

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="UTF-8">
  <script type="text/javascript">

    var color=["white", "yellow", "aqua", "purple"];

    var i=0;
    function changeColor() {
      i++;
      if(i >= color.length) {
        i = 0;
      }

      var bodyTag = document.getElementById("theBody");
      bodyTag.style.backgroundColor = color[i];
    }

  </script>
</head>
<body id="theBody">
  <button onclick="changeColor();">배경색 바꾸기</button>
</body>
</html>
```

기본 함수 정의문

■ 익명 함수

- 이름을 가지지 않는 함수를 의미
- 익명 함수의 생성

```
함수참조변수 = function( ) {  
  
    함수의 실행 문장들;  
  
}
```

```
<script type="text/javascript">  
    var myFunc = function( ) {  
        var output = prompt("숫자를 입력하세요.", "숫자");  
        alert(output);  
    }  
</script>
```

- 익명 함수의 실행

- 참조 변수의 호출에 의해 실행됨

```
함수참조변수();
```

```
myFunc();
```

기본 함수 정의문

4-function-noname.htm

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="UTF-8">

  <script type="text/javascript">

    var myFunc = function() {

      document.write("hello", "<br>");
      document.write("welcome", "<br>");

    }

    myFunc();
    myFunc();

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

```
hello
welcome
hello
welcome
```

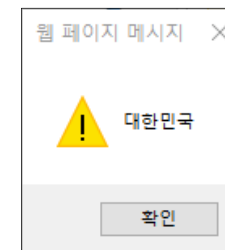

기본 함수 정의문

■ 선언 함수와 익명 함수의 차이

• 선언 함수

- 호이스팅(hoisting) 기술을 지원
 - 함수 정의문보다 호출문이 먼저 나와도 정상적으로 함수를 실행함

```
<script type="text/javascript">
    myFunc();
    function myFunc( ) { alert( ' 대한민국' ); }
</script>
```



• 익명 함수

- 호이스팅(hoisting) 기술을 지원하지 않음
 - 함수 정의문보다 호출문이 먼저 나오면 오류 발생

```
<script type="text/javascript">
    myFunc();
    var myFunc = function( ) { alert('대한민국'); }
</script>
```

오류발생

함수의 매개 변수

■ 매개 변수가 있는 함수

- 함수를 호출할 때 함수의 실행에 사용되는 값을 함수의 내부에 전달할 수 있음
 - 전달하는 값은 함수를 호출할 때 전달하며, 함수의 매개변수를 통해 전달됨

```
function 함수명( 매개변수1, 매개변수2, ... 매개변수n ) {           // 함수의 정의
    자바스크립트 코드
}

함수명( 데이터1, 데이터2, ..., 데이터n )                       // 함수의 호출
```

함수의 매개 변수

4-function-param-1.htm (p.136)

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="UTF-8">

  <script type="text/javascript">

    function myFnc(name, area) {
      document.write("안녕하세요. " + name + "입니다.", "<br>");
      document.write("사는곳은 " + area + "입니다.", "<br><br>");
    }

    myFnc("홍당무", "서울");

    myFnc("깍두기", "부산");

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

안녕하세요. 홍당무입니다.
사는곳은 서울입니다.

안녕하세요. 깍두기입니다.
사는곳은 부산입니다.

함수의 매개 변수

4-function-param-2.htm (p.136)

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="UTF-8">
  <script type="text/javascript">

    var rightId = "korea";
    var rightPw = "1234";
    function login(id, pw){
      if(id == rightId){
        if(pw == rightPw){
          document.write(id + "님 방문을 환영합니다");
        }else{
          alert("잘못된 비밀번호입니다.");
        }
      } else {
        alert("존재하지 않는 아이디입니다.");
      }
    }
    var userId = prompt("아이디를 입력하세요.", "");
    var userPw = prompt("패스워드를 입력하세요.", "");
    login(userId, userPw);

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

localhost:8080 내용:

아이디를 입력하세요.

확인

취소

localhost:8080 내용:

패스워드를 입력하세요.

확인

취소

korea님 방문을 환영합니다

함수의 매개 변수

■ arguments 매개변수

- 자바스크립트의 모든 함수가 기본적으로 가지는 변수
- 일반적으로 함수의 매개변수와 함수에 전달되는 값은 1:1 대응 되어야 함
- 함수의 매개변수가 없는 상태에서 값을 전달할 수도 있음
 - 전달되는 값은 arguments라는 배열로 저장됨
 - 함수는 arguments 배열의 요소를 참조해 전달된 값을 사용할 수 있음

```
function 함수명() {  
    arguments;  
    자바스크립트 코드  
}  
  
함수명( 데이터1, 데이터2, ..., 데이터n )
```

// 매개변수가 존재하지 않음
// 전달된 값은 arguments 배열에 저장됨
// 값은 arguments 배열 요소를 참조할 수 있음
// 함수를 호출할 때 값을 전달함

함수의 매개 변수

4-function-arguments.htm (p.138)

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="UTF-8">
  <script type="text/javascript">

    function sum() {
      var sum = 0;

      document.write("매개변수의 수 : ", arguments.length , "<br>" )
      for( var i=0; i<arguments.length; i++ ) {
        sum += arguments[i];
      }
      document.write("합계 : ", sum);
    }

    sum(10, 20, 30);

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

매개변수의 수 : 3
합계 : 60

return 문

■ 데이터를 반환하는 return 문

- 함수의 수행 결과는 반환하는 역할을 수행

```
function 함수명( ) {  
    함수의 실행 문장들;  
    return 데이터(값);  
}
```

```
// 함수를 호출한 위치로 결과 값을 반환  
var 변수 = 함수명( );
```

```
<!DOCTYPE html>  
<html>  
<head>  
    <meta charset="UTF-8">  
    <script type="text/javascript">  
  
        function myFnc(a, b) {  
            var num=a*b;  
            return num;  
        }  
        var result = myFnc(10, 3);  
  
        document.write(result);  
  
    </script>  
</head>  
<body></body>  
</html>
```

return 문

■ 강제 종료 역할을 수행하는 return 문

- 수행중인 함수를 중단시키는 역할 수행

```
function 함수이름( ) {  
    함수의 실행 문장 - 1;  
    return;  
    함수의 실행 문장 - 2;  
}  
  
var 변수 = 함수이름( );  
  
// 함수의 실행 문장 - 2는 수행되지 않음
```

```
<!DOCTYPE html>  
<html>  
<head>  
    <meta charset="UTF-8">  
    <script type="text/javascript">  
  
        function rtn(){  
            document.write("Hello");  
            return;  
            document.write("World");  
        }  
  
        rtn();  
  
    </script>  
</head>  
<body></body>  
</html>
```


return 문

4-function-return.htm (p.140)

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="UTF-8">
  <script type="text/javascript">

    function testAvg(arrData){
      var sum = 0;
      for(var i = 0; i < arrData.length; i++) {
        sum += Number(prompt(arrData[i] + " 점수는?", "0"));
      }

      var avg = sum / arrData.length;
      return avg;
    }

    var arrSubject = ["국어", "수학"];
    var result = testAvg(arrSubject);

    document.write("평균 점수는 " + result + "점 입니다");

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

국어 점수는?

확인

취소

수학 점수는?

확인

취소

평균 점수는 92점 입니다

재귀 함수

■ 재귀 함수

- 함수 내에서 함수 자신을 호출하는 방법
 - 반복문과 같이 함수 자신을 여러 번 호출해 실행하기 위해 사용됨

```
function myFunc() {  
    자바스크립트 코드;  
    myFunc();  
}  
  
myFunc();
```

재귀 함수

4-function-recursive.htm (p.143)

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="UTF-8">
  <script type="text/javascript">

    var num = 0;

    function testFnc(){
      num++;
      document.write(num, "<br>");
      if(num == 10) return;

      testFnc();
    }

    testFnc();

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

1
2
3
4
5
6
7
8
9
10

■ 지역 변수와 전역 변수

• 지역 변수

- 특정한 모듈(예를 들어 함수 내)에서만 사용될 수 있는 변수

• 전역 변수

- 해당 문서 전체에서 사용될 수 있는 변수

• 지역 변수와 전역 변수의 선언 방법

- 특정 영역의 외부에서 선언하는 경우
 - var 키워드를 사용하거나 사용하지 않아도 모두 전역 변수로 선언됨
- 특정 영역의 내부에서 선언하는 경우
 - var 키워드를 사용하지 않고 선언하면 전역 변수로 선언됨
 - var 키워드를 사용해 선언하면 지역 변수로 선언됨

함수의 스코프

4-function-scope.htm (p.143)

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="UTF-8">
  <script type="text/javascript">

    var planet = "지구";           // 전역 변수
    cont = "아시아";             // 전역 변수
    function test1() {
      nation="한국";             // 전역 변수
      var prov = "충북";         // 지역 변수
      document.write("planet : " + planet + "<BR>");
      document.write("cont : " + cont + "<BR><BR>");
    }
    function test2() {
      document.write("planet : " + planet + "<BR>");
      document.write("cont : " + cont + "<BR>");
      document.write("nation : " + nation + "<BR>");
      document.write("prov : " + prov + "<BR>");
    }
    test1();
    test2();

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

planet : 지구
cont : 아시아

planet : 지구
cont : 아시아
nation : 한국

객체 생성자 함수의 활용

객체 생성자 함수

■ 생성자 함수

- new 연산자를 사용해 객체를 생성할 수 있는 함수
 - this 키워드를 사용해 속성에 값을 지정하거나 함수를 등록함

```
function 함수명( 매개변수1, 매개변수2, ..., 매개변수n ) {  
    this.속성명 = 값;  
    this.함수명 = function() {  
        자바스크립트 코드  
    }  
    .....  
}  
  
var 인스턴스 = new 함수명();
```

```
var student = new Student();  
    인스턴스      생성자 함수
```

객체 생성자 함수

4-constructor.htm (p.149)

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="UTF-8">
  <script type="text/javascript">

    function CheckWeight(name, height, weight) {

      this.userName = name;
      this.userHeight = height;
      this.userWeight = weight;
      this.minWeight;
      this.maxWeight;

      this.getInfo = function() {
        var str = ""
        str += "이름: " + this.userName + ", ";
        str += "키: " + this.userHeight + ", ";
        str += "몸무게: " + this.userWeight + "<br>";
        return str;
      }
    }
  }
```


객체 생성자 함수

```
        this.getResult = function( ) {  
            this.minWeight = (this.userHeight - 100) * 0.9 - 5;  
            this.maxWeight = (this.userHeight - 100) * 0.9 + 5;  
            if(this.userWeight >= this.minWeight &&  
                this.userWeight <= this.maxWeight) {  
                return "정상 몸무게입니다";  
            } else if(this.userWeight < this.minWeight) {  
                return "정상 몸무게보다 미달입니다";  
            } else {  
                return "정상 몸무게보다 초과입니다";  
            }  
        }  
    }  
}  
  
var jang = new CheckWeight("장보리", 168, 62);  
var park = new CheckWeight("박달재", 180, 88);  
  
console.log(jang);  
console.log(park);  
  
document.write(jang.getInfo());  
document.write(jang.getResult());  
</script>  
</head>  
<body> </body>  
</html>
```

객체 생성자 함수

이름: 장보리, 키: 168, 몸무게: 62
정상 몸무게입니다

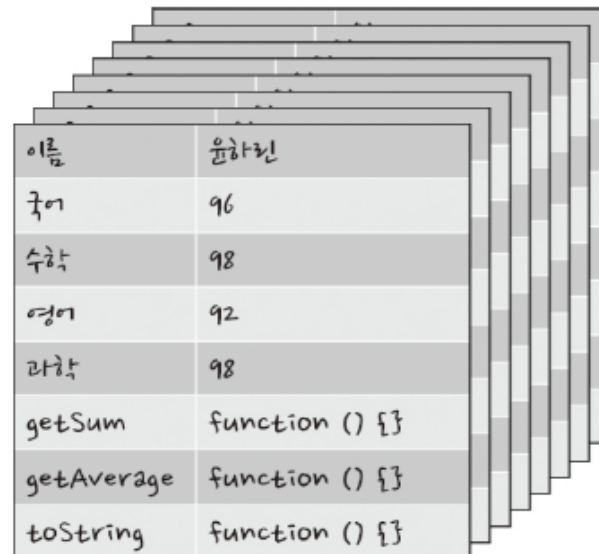
The screenshot shows a web browser's developer console with the 'Console' tab selected. The breadcrumb path is 'top'. There are two log entries, both of which are collapsed 'CheckWeight' objects. The first object has the following properties: `getInfo` (function), `getResult` (function), `maxWeight`: 66.2, `minWeight`: 56.2, `userHeight`: 168, `userName`: "장보리", `userWeight`: 62, and `__proto__`: Object. The second object has the following properties: `getInfo` (function), `getResult` (function), `userHeight`: 180, `userName`: "박달재", `userWeight`: 88, and `__proto__`: Object.

```
▼ CheckWeight ⓘ  
  ▶ getInfo: f ()  
  ▶ getResult: f ()  
    maxWeight: 66.2  
    minWeight: 56.2  
    userHeight: 168  
    userName: "장보리"  
    userWeight: 62  
  ▶ __proto__: Object
```

```
▼ CheckWeight ⓘ  
  ▶ getInfo: f ()  
  ▶ getResult: f ()  
    userHeight: 180  
    userName: "박달재"  
    userWeight: 88  
  ▶ __proto__: Object
```

■ 생성자 함수를 통해 생성되는 객체들의 문제점

- 생성되는 객체들의 속성과 메서드
 - 생성되는 객체들은 모두 서로 다른 속성 값을 가짐
 - 하지만 생성되는 객체들은 모두 같은 메서드를 가짐
- 생성되는 객체가 많을 경우 중복되는 메서드로 인한 메모리 낭비 초래

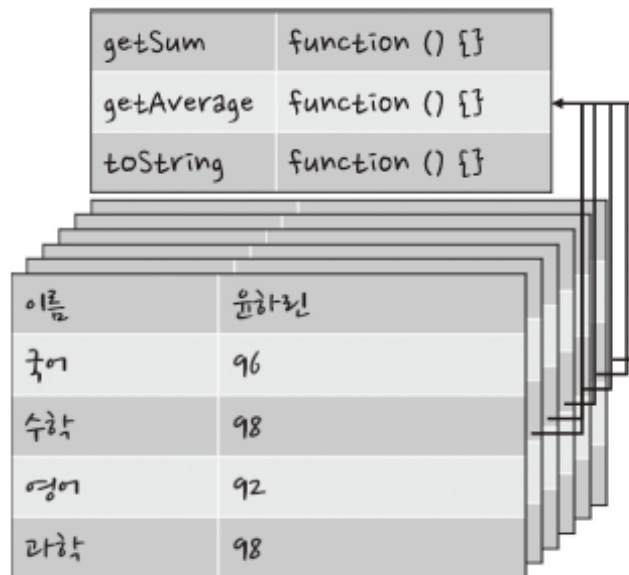


이름	윤하린
국어	96
수학	98
영어	92
과학	98
getSum	function () {}
getAverage	function () {}
toString	function () {}

프로토타입

■ 프로토 타입(prototype)

- 생성자 함수로 생성된 객체가 공통으로 가지는 공간
- 생성자 함수 내에 메서드를 선언하지 않고 외부에 프로토타입으로 선언
 - 생성자 함수 내에는 중복되지 않는 속성만을 선언
 - 생성되는 객체들은 프로토타입으로 선언된 메서드를 공유함



■ 프로토타입 메서드의 선언

- 함수의 외부에서 prototype 키워드를 사용해 선언

```
function 함수명( 매개변수1, 매개변수2, ..., 매개변수n ) {  
    this.속성명 = 값;  
    .....  
}
```

```
함수명.prototype.메서드이름 = function() {  
    자바스크립트 코드;  
}
```

```
var 인스턴스 = new 함수명();
```

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="UTF-8">
  <script type="text/javascript">

    function CheckWeight(name, height, weight) {
      this.userName = name;
      this.userHeight = height;
      this.userWeight = weight;
      this.minWeight;
      this.maxWeight;
    }

    CheckWeight.prototype.getInfo = function() {
      var str = ""
      str += "이름: " + this.userName + ", ";
      str += "키: " + this.userHeight + ", ";
      str += "몸무게: " + this.userWeight + "<br>";
      return str;
    }
  }
```

프로토타입

```
CheckWeight.prototype.getResult = function( ) {  
    this.minWeight = (this.userHeight - 100) * 0.9 - 5;  
    this.maxWeight = (this.userHeight - 100) * 0.9 + 5;  
    if(this.userWeight >= this.minWeight &&  
        this.userWeight <= this.maxWeight) {  
        return "정상 몸무게입니다";  
    } else if(this.userWeight < this.minWeight) {  
        return "정상 몸무게보다 미달입니다";  
    } else {  
        return "정상 몸무게보다 초과입니다";  
    }  
}
```

```
var jang = new CheckWeight("장보리", 168, 62);  
var park = new CheckWeight("박달재", 180, 88);
```

```
document.write(jang.getInfo());  
document.write(jang.getResult(), "<br>");
```

```
document.write(jang.getResult === park.getResult);
```

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

이름: 장보리, 키: 168, 몸무게: 62
정상 몸무게입니다
true

자바스크립트 내장함수

자바스크립트 내장함수

■ 인코딩(encoding)과 디코딩(decoding)

• 인코딩

- 문자열을 컴퓨터에 저장하거나 통신에 사용할 목적으로 부호화하는 방법
- 인코딩을 해야 하는 이유
 - 웹에서 문자데이터를 전달할 때 한글과 같은 문자는 오류를 발생시킬 수 있기 때문

• 디코딩

- 인코딩에 의해 부호화 된 문자열을 다시 원래의 문자열을 복원하는 방법

함수	설명
encodeURIComponent()	알파벳, 숫자, 일부 특수문자(@, _ , +, ., ;, /, =, ?, &)를 제외하고 유니코드로 인코딩 <ul style="list-style-type: none">- 1바이트 문자 : %XX 형식- 2바이트 문자 : %uXXXX 형식
decodeURI()	encodeURIComponent() 메서드로 인코딩된 문자열의 디코딩
encodeURIComponent()	알파벳과 숫자를 제외한 모든 문자를 유니코드로 인코딩
decodeURIComponent()	decodeURIComponent() 메서드로 인코딩된 문자열의 디코딩을 수행

자바스크립트 내장함수

4-encode-decode.htm

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="UTF-8">
  <script type="text/javascript">

    var URI = "http://www.ut.ac.kr?name=심청";

    var output1 = encodeURI(URI);
    document.write("encodeURI() : ", output1, "<br>");
    document.write("decodeURI() : ", decodeURI(output1), "<br><br>");

    var output2 = encodeURIComponent(URI);
    document.write("encodeURIComponent() : ", output2, "<br>");
    document.write("decodeURIComponent() : ", decodeURIComponent(output2), "<br>");

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

encodeURI() : http://www.ut.ac.kr?name=%EC%8B%AC%EC%B2%AD

decodeURI() : http://www.ut.ac.kr?name=심청

encodeURIComponent() : http%3A%2F%2Fwww.ut.ac.kr%3Fname%3D%EC%8B%AC%EC%B2%AD

decodeURIComponent() : http://www.ut.ac.kr?name=심청

자바스크립트 내장함수

■ parseInt() 함수

- 인자로 지정된 값을 정수형 데이터(10진수)로 변환
 - 사용자가 입력한 값이나 특정 변수의 값을 정수로 변환할 때 사용
 - 인자가 실수인 경우 정수형으로 변환
 - 인자가 0으로 시작하면 8진수, 0x로 시작하면 16진수로 인식하여 변환
 - parseInt(0x273) : 627 (16진수 273을 10진수로 변환)
 - parseInt(0273) : 187 (8진수 273을 10진수로 변환)
 - parseInt(08) : 8 (08은 10진수로 인식함)
 - parseInt(07) : 7 (07은 8진수로 인식함)
 - 함수의 두 번째 인자로 진법을 입력할 경우, 앞의 수를 해당 진법으로 인식
 - parseInt('FF', 16) : 255
 - parseInt('52', 10) : 52
 - parseInt('11', 8) : 9
 - parseInt('10', 2) : 2

자바스크립트 내장함수

4-parseInt.htm

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="UTF-8">
  <script type="text/javascript">

    document.write(parseInt(10.4),"<br>");
    document.write(parseInt(016),"<br>");
    document.write(parseInt(0x0c),"<br>");
    document.write(parseInt(07),"<br>");
    document.write(parseInt(08),"<br>");
    document.write(parseInt(010),"<br><br>");

    document.write(parseInt('FF', 16),"<br>");
    document.write(parseInt('11', 8),"<br>");
    document.write(parseInt('10', 2));

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

```
10
14
12
7
8
8

255
9
2
```

자바스크립트 내장함수

■ parseFloat() 함수

- 인자로 지정한 값을 실수형 데이터로 변환
 - 인자가 정수인 경우 정수로 표현하고, 실수인 경우 실수로 표현
 - 지수형 데이터도 변환 가능
- parseInt()나 parseFloat() 함수를 사용하는 이유
 - 입력 받은 값이나 변수의 값을 연산에 참여시키기 위해 데이터 타입을 정수나 실수로 변환시키기 위해 사용
 - 묵시적인 데이터 타입의 병환이 발생하나 확실하게 데이터 타입을 지정하기 위함

자바스크립트 내장함수

4-parseFloat.htm

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="UTF-8">
  <script type="text/javascript">

    document.write(parseFloat(10),"<br>");
    document.write(parseFloat(10.6),"<br>");
    document.write(parseFloat(52.273e5),"<br>");

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

```
10
10.6
5227300
```

자바스크립트 내장함수

■ isNaN() 함수

- 인자로 가지는 값이 숫자인지 조사하는 함수
 - 인자가 숫자이면 FALSE를 반환, 문자이면 TRUE를 반환
 - 주로 IF 조건문에서 조건으로 사용

자바스크립트 내장함수

4-isNaN.htm

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="UTF-8">
  <script type="text/javascript">

    var input=prompt("출력할 단을 입력하시오.", "");
    if(isNaN(input)) {
      alert("숫자를 입력해야합니다.");
      exit;
    }

    dan=parseInt(input,10);
    if(dan<2 || dan>9) {
      alert("2단부터 9단까지만 입력합니다.");
      exit;
    }

    for(var i=1; i<=9; i++){
      document.write(dan , " X " , i , " = " , dan*i);
      document.write("<BR>");
    }

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

localhost:8080 내용:

출력할 단을 입력하시오.

7

확인

취소

7 X 1 = 7
7 X 2 = 14
7 X 3 = 21
7 X 4 = 28
7 X 5 = 35
7 X 6 = 42
7 X 7 = 49
7 X 8 = 56
7 X 9 = 63

자바스크립트 내장함수

■ eval() 함수

- 문자열로 표현된 수식을 계산하여 그 결과를 반환하는 함수
 - 주로 prompt() 함수를 사용해 수식을 입력 받거나, 문자열로 형식의 수식을 계산하여 결과를 구하는데 사용

```
변수 = eval("계산할 인자")
```

- [예]

```
var a = "3 + 5";  
var b = eval(a);           // 8을 출력
```

- 인자를 자바스크립트 객체로 변경할 때 사용
 - 문자열로 된 자바스크립트 코드를 실제 자바스크립트 객체로 변경할 때 사용
 - 자바스크립트의 메서드나 속성을 적용할 수 있도록 하기 위해 변경

자바스크립트 내장함수

4-eval-1.htm

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="UTF-8">
  <script type="text/javascript">

    var str = "10 + 20";
    alert( eval(str) );

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

localhost:8080 내용:

30

확인

자바스크립트 내장함수

4-eval-2.htm

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="UTF-8">
  <script type="text/javascript">

    var willEval = "";

    willEval += 'var number = 10;';
    willEval += 'alert(number);';
    eval(willEval);

    var result = number + 30;
    alert(result);

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

localhost:8080 내용:

10

확인

localhost:8080 내용:

40

확인

자바스크립트 내장함수

■ 기타 내장함수

• String() 함수

- 인자로 가지는 데이터를 문자열로 변환하는 함수(숫자 데이터의 문자열로 변환하는데 사용)

```
변수 = String("데이터")
```

• Number() 함수

- 인자로 가지는 데이터를 숫자로 변환하는 함수
 - 키보드로 입력되는 데이터는 문자열로 간주되므로, 이를 숫자 데이터로 변경할 필요가 있을 때 사용

```
변수 = Number("데이터")
```

• Boolean() 함수

- 논리형 데이터를 반환

```
변수 = Boolean("데이터")
```

수고하셨습니다