5 함수

모던 웹을 위한 Javascript jQuery 입문

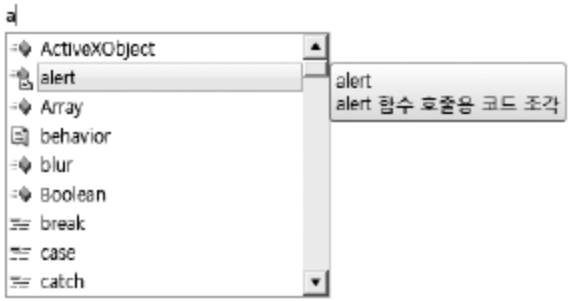
한빛미디어 Hanbit Media, Inc.

5. 함수

❖ 함수란?

- 코드의 집합
- 그림 5-1처럼 지금까지 사용하던 alert()나 prompt() 같은 것들

그림 5-1 Visual Web Developer Express의 함수 표시



5.1 함수 개요

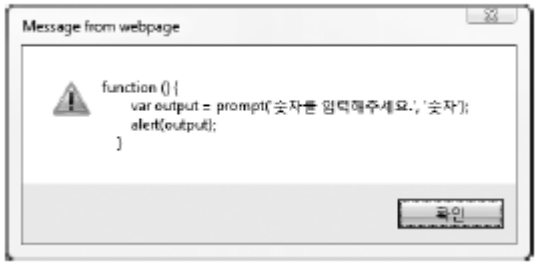
❖ 함수의 형태

- var 함수 = function ( ) { };
- 괄호 내부에 코드를 넣음
- 예제코드 5-1] 두 문장을 포함하는 함수를 생성하고 출력
  - 문자열처럼 보일 수 있지만 typeof 연산자를 사용하면 함수 자료형

코드 5-1 함수의 생성과 출력

```
<script>
    var 함수 = function ( ) {
        var output = prompt('숫자를 입력해주세요.', '숫자');
        alert(output);
    };
    alert(함수);
</script>
```

그림 5-2 함수의 출력



5.1 함수 개요

❖ 익명 함수

- function () {} 형태는 함수지만 이름을 가지고 있지 않음
- 이름이 없으므로 변수에 넣어 사용해야 함

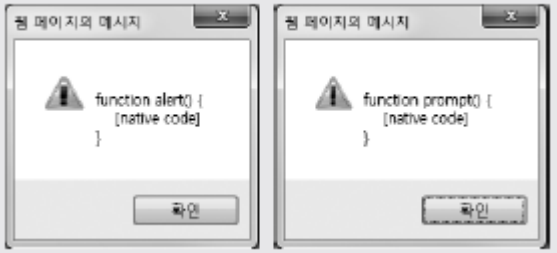
❖ 내장 함수의 출력

- 모든 브라우저는 내장하고 있는 함수의 소스를 볼 수 없게 막아놓음
- '선언적 함수'
  - 그림 5-3의 함수는 alert.prompt라는 이름을 가지고 있음
  - 이름을 가지고 있는 함수

코드 5-2 내장 함수 출력

```
<script>
    alert(alert);
    alert(prompt);
</script>
```

그림 5-3 실행 결과

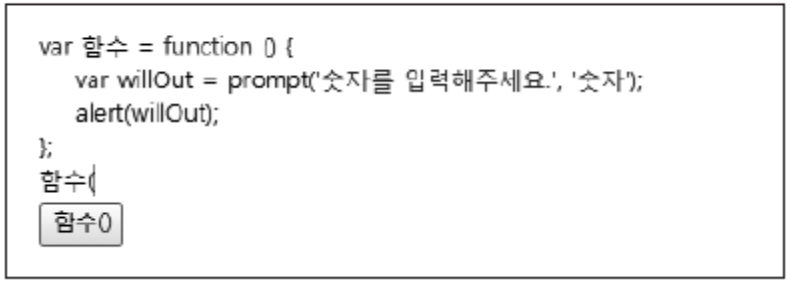


5.1 함수 개요

❖ 함수 호출

- 함수는 자료형이지만 뒤에 괄호를 열고 닫음으로써 코드 실행
- 함수를 실행하는 것

그림 5-4 함수의 호출



코드 5-3 함수의 호출

```
<script>
    // 함수를 만듭니다.
    var 함수 = function () {
        var output = prompt('숫자를 입력해주세요.', '숫자');
        alert(output);
    };

    // 함수를 호출합니다.
    함수();
</script>
```

5.2 선언적 함수

❖ 일반적인 함수의 형식

- '선언적 함수'라 표현
  - function 함수( ) {
  - }
- 아래 위 두 함수는 같은 기능을 수행함
  - var 함수 = function ( ) { };
- 익명 함수와 마찬가지 방법으로 만들고 사용

❖ 예제 코드 5-4

- 이름이 같은 두 개의 함수
- 호출 결과는 ? 뒤쪽의 함수 결과 출력

코드 5-4 선언적 함수의 재정의 (1)

```
<script>
    function 함수() { alert('함수 A'); }
    function 함수() { alert('함수 B'); }
    함수();
</script>
```

그림 5-5 재정의된 함수의 실행

코드 5-5 익명 함수의 재정의 (1)

```
<script>
    var 함수 = function () { alert('함수 A'); }
    var 함수 = function () { alert('함수 B'); }
    함수();
</script>
```

5.2 선언적 함수

❖ 예제 코드 5-6

- 오류가 발생해 실행되지 않음
  - 변수를 선언하기 이전에 변수를 사용했기 때문

코드 5-6 익명 함수의 재정의 (2)

```
<script>
    함수();
    var 함수 = function () { alert('함수 A'); }
    var 함수 = function () { alert('함수 B'); }
</script>
```

5.2 선언적 함수

❖ 예제 코드 5-7]

- 선언적 함수를 사용하면 정상적으로 코드 실행
- 웹 브라우저는 선언적 함수 먼저 읽음
  - 코드 5-7는 2번째 줄 → 3번째 줄 → 1번째 줄의 순서로 실행
  - 문자열 '함수 B' 출력

코드 5-7 선언적 함수의 재정의 (2)

```
<script>
    함수();
    function 함수() { alert('함수 A'); }
    function 함수() { alert('함수 B'); }
</script>
```

5.3 매개 변수와 리턴값

❖ 매개 변수

- 함수를 호출할 때 괄호 안에 적는 것

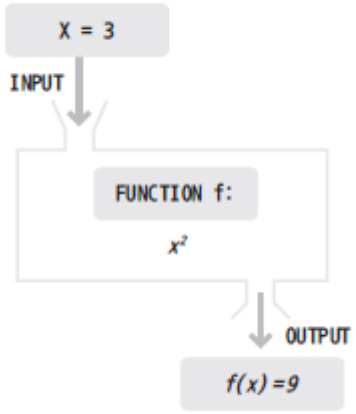
❖ 리턴값

- 함수를 호출하고 함수가 변환되는 값
  - prompt() 함수를 사용하면 사용자가 입력한 문자열로 변환

그림 5-6 prompt() 함수의 매개 변수


```
function 함수이름(매개 변수, 매개 변수, 매개 변수) {
    // 함수 코드
    return 리턴값;
}
```

그림 5-7 함수



5.3 매개 변수와 리턴값

❖ 예제 코드 5-8]

- 5-7의 함수를 자바스크립트로 코드화
- 매개 변수로 x를 넣으면 x2을 리턴하는 함수

코드 5-8 매개 변수와 리턴값

```
<script>
    function f(x) { return x * x; }
    alert(f(3));
</script>
```

그림 5-8 함수 호출 결과

5.4 매개 변수

❖ 매개 변수란?

- 함수를 호출하는 쪽과 호출된 함수를 연결하는 매개가 되는 변수
- 자바스크립트는 함수를 생성할 때 지정한 매개 변수보다 많거나 적은 매개 변수를 사용하는 것 허용
- 예제 코드 5-9]
  - alert() 함수와 prompt() 함수의 매개 변수 입력 테스트
    - **원래 함수에서 선언된 매개 변수보다 많게 사용하면?**
      - » **추가된 매개 변수 무시**

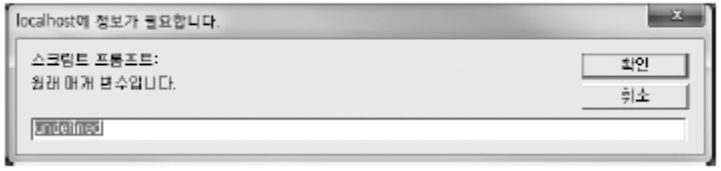
코드 5-9 비유효 매개변수

```
<script>
    alert('원래 매개 변수입니다.', '추가된 매개 변수입니다.');
    prompt('원래 매개 변수입니다.');
</script>
```

❖ 매개 변수란?

- alert() 함수와 prompt() 함수의 매개 변수 입력 테스트
  - **원래 함수에서 선언된 매개 변수보다 적게 사용하면?**
    - » **지정하지 않는 매개 변수는 undefined**

그림 5-10 prompt('원래 매개 변수입니다.')의 실행 결과



❖ 예제 코드 5-10] Array() 함수

- 지정한 매개 변수보다 많거나 적게 매개 변수를 사용하는 점 이용

코드 5-10 Array() 함수

```
<script>
    // 배열을 생성합니다.
    var array1 = Array();
    var array2 = Array(10);
    var array3 = Array(273, 103, 57, 32);

    // 배열을 사용합니다.
    alert(array1 + '\n' + array2 + '\n' + array3 + '\n');
</script>
```

그림 5-11 Array() 함수의 형태

그림 5-12 같은 함수지만 서로 다른 실행 결과

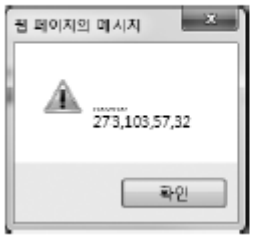


표 5-1 Array() 함수의 매개 변수에 따른 차이

함수 형태	설명
Array()	빈 배열을 만듭니다.
Array(number)	매개 변수만큼의 크기를 가지는 배열을 만듭니다.
Array(mixed, ... , mixed)	매개 변수를 배열로 만듭니다.

5.5 가변인자 함수

❖ 가변인자 함수란?

- 매개 변수의 개수가 변할 수 있는 함수
- 협의로는 매개 변수를 선언된 형태와 다르게 사용했을 때도 매개 변수를 모두 활용하는 함수
  - 5.4절 Array() 함수 참고

5.5 가변인자 함수

❖ sumAll() 함수

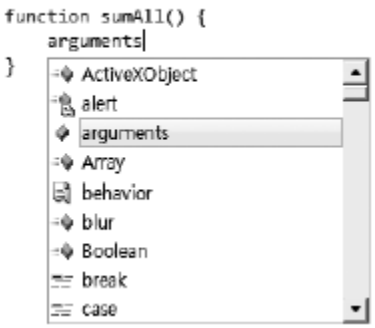
- 매개 변수로 입력된 숫자를 모두 더하는 함수

```
<script>
    function sumAll() {

    }
</script>
```

- 자바스크립트의 모든 함수는 내부에 자동으로 변수 arguments 가짐

그림 5-13 함수 내에 무조건 존재하는 변수 arguments



5.5 가변인자 함수

❖ 예제 코드 5-11]

- arguments 객체의 자료형과 배열의 길이 출력
  - 함수를 호출할 때 아홉 개의 매개 변수 입력
    - **arguments 배열의 length 속성은 9**

코드 5-11 arguments 객체

```
<script>
    function sumAll() {
        alert(typeof (arguments) + ': ' + arguments.length);
    }
    // 함수를 호출합니다.
    sumAll(1, 2, 3, 4, 5, 6, 7, 8, 9);
</script>
```

그림 5-14 arguments 객체

5.5 가변인자 함수

❖ 예제 코드 5-12]

- 변수 output을 만들고 반복문 사용
- output에 arguments 배열 내의 모든 요소 더함
- 함수를 호출해 출력 - 코드를 실행하면 45 출력

코드 5-12 가변 인자 함수

```
<script>
    function sumAll() {
        var willReturn = 0;
        for (var i in arguments) {
            willReturn += arguments[i];
        }
        return willReturn;
    }
    // 함수를 호출합니다.
    alert(sumAll(1, 2, 3, 4, 5, 6, 7, 8, 9));
</script>
```

5.5 가변인자 함수

❖ 함수의 매개 변수 숫자가 다를 때 처리하는 예제

- 배열 arguments의 요소 개수에 따라 조건 설정

```
<script>
    function 이렇게함수() {
        // 매개 변수의 개수를 가져옵니다.
        var length = arguments.length;

            // 조건을 나누어줍니다.
            if (length == 0) {
                // 매개 변수가 없을 때
            } else if (length == 1) {
                // 매개 변수가 한 개일 때
        } else {
                // 매개 변수가 두 개일 때
        }
    }
</script>
```

5.6 리턴값

❖ 리턴값 활용

- return 키워드를 사용해 함수를 호출한 곳으로 값 넘김
- return 키워드의 의미?
  - 함수가 실행되는 도중 함수를 호출한 곳으로 돌아가라는 의미
  - return 키워드 사용시 값을 지정하지 않아도 함수를 호출한 곳으로 돌아감
  - 예제에서는 return 키워드 '문장 B' 이전에 사용
    - **'문장 A' 만 출력**

코드 5-13 return 키워드

```
<script>
    // 함수를 선언합니다.
    function returnTest() {
        alert('문장 A');
        return;
        alert('문장 B');
    }
    // 함수를 호출합니다.
    returnTest();
</script>
```

5.6 리턴값

❖ 예제 코드 5-14]

- returnTest()의 리턴값 받고 출력

코드 5-14 return 키워드 뒤에 값을 입력하지 않을 경우

```
<script>
    // 함수를 선언합니다.
    function returnTest() {
        alert('문장 A');
        return;
        alert('문장 B');
    }
    // 함수를 호출합니다.
    var value = returnTest();
    alert(typeof (value) + ' : ' + value);
</script>
```

그림 5-15 함수에서 아무 값도 리턴하지 않았을 경우의 자료형과 값

5.7 내부 함수

❖ 내부 함수란?

- 함수 내부에 선언한 함수

```
function 외부 함수() {
function 내부 함수 1() {
    // 함수 코드
    }
function 내부 함수 2() {
    // 함수 코드
    }

// 함수 코드
}
```

5.7 내부 함수

❖ 피타고라스의 정리를 구현하는 함수 pythagoras()

- 밑변 width, 높이height가 매개 변수
  - 빗변hypotenuse의 길이 리턴

그림 5-16 피타고라스의 정리

$$hypotenuse = \sqrt{width^2 + height^2}$$

코드 5-15 피타고라스의 정리 함수 (1)

```
<script>
    function pythagoras(width, height) {
        return Math.sqrt(width * width + height * height);
    }
</script>
```

코드 5-16 피타고라스의 정리 함수 (2) – 함수 추출

```
<script>
    // 제곱을 구하는 함수
    function square(x) { return x * x; }

    // 피타고라스 함수
    function pythagoras(width, height) {
        return Math.sqrt(square(width) + square(height));
    }
</script>
```

❖ 함수 이름이 충돌하는 경우

- 다른 구성원이 같은 이름으로 함수를 만들었다면?

코드 5-17 함수 이름 충돌

```
<script>
    /* 윤씨가 만든 함수 */
    // 제곱을 구하는 함수
    function square(x) { return x * x; }
    // 피타고라스 함수
    function pythagoras(width, height) {
        return Math.sqrt(square(width) + square(height));
    }
    // 피타고라스 함수를 호출합니다.
    alert(pythagoras(3, 4));

    /* 연씨가 만든 함수 */
    // 삼각형이 직각인지 확인하는 함수
    function square(width, height, hypotenuse) {
        if (width * width + height * height == hypotenuse * hypotenuse) {
            return true;
        } else {
            return false;
        }
    }
</script>
```

그림 5-17 뒤에 위치하는 함수가 실행됨으로써 발생한 문제

5.7 내부 함수

❖ 함수 이름이 충돌하는 경우

- 예제 코드 5-18]
  - 내부 함수를 사용하면 외부에 이름이 같은 함수가 있어도 내부 함수 우선
- 내부 함수는 내부 함수가 포함되는 함수에서만 사용 가능

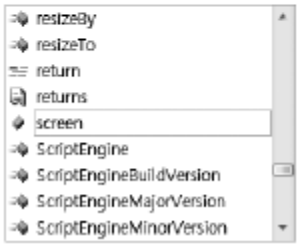
코드 5-18 내부 함수

```
<script>
    function pythagoras(width, height) {
        function square(x) {
            return x * x;
        }
        return Math.sqrt(square(width) + square(height));
    }
</script>
```

그림 5-18 내부 함수는 외부에서 사용할 수 없습니다.

```
function pythagoras(width, height) {
    function square(x) {
        return x * x;
    }
    return Math.sqrt(square(width) + square(height));
}

square
```



5.8 함수를 매개 변수로 받는 함수

❖ 예제 코드 5-19]

- callFunctionTenTimes() 함수
  - 함수를 매개 변수로 받아 해당 함수 10번 호출

코드 5-19 익명 함수를 매개 변수로 전달

```
<script>
    // 함수를 10번 호출하는 함수입니다.
    function callFunctionTenTimes(otherFunction) {
        for (var i = 0; i < 10; i++) {
            otherFunction();
        }
    }

    // callFunctionTenTimes() 함수를 호출합니다.
    callFunctionTenTimes(function () {
        alert('Hello World..!');
    });
</script>
```

5.8 함수를 매개 변수로 받는 함수

❖ 예제 코드 5-19]

- callFunctionTenTimes() 함수
  - 함수를 매개 변수로 받아 해당 함수 10번 호출 (2)

코드 5-20 선언적 함수를 매개 변수로 전달

```
<script>
    // 함수를 10번 호출하는 함수입니다.
    function callFunctionTenTimes(otherFunction) {
        for (var i = 0; i < 10; i++) {
            otherFunction();
        }
    }

    // justFunction() 함수입니다.
    function justFunction() {
       alert('Hello World..!');
    }

    // callFunctionTenTimes() 함수를 호출합니다.
    callFunctionTenTimes(justFunction);
</script>
```

5.9 함수를 리턴하는 함수와 클로저

❖ 예제 코드 5-21]

- 함수를 리턴하는 함수
  - 코드를 실행하면 문자열 'HelloWorld .. !' 출력

코드 5-21 익명 함수를 리턴하는 함수

```
<script>
    function outerFunction() {
        return function () {
    alert('Hello World .. !');
        };
    }

    outerFunction()();
</script>
```

5.9 함수를 리턴하는 함수와 클로저

❖ 예제 코드 5-22]

- 코드에 오류가 있어 웹 페이지 오류가 발생하거나 경고창 미출력
- 함수 안에 있는 변수는 지역 변수이므로 외부에서 사용할 수 없음
- 예외 >
  - 클로저를 사용하면 이 규칙을 위반할 수 있음

코드 5-22 지역 변수의 유효 범위

```
<script>
    // 함수를 선언합니다.
    function test(name) {
        var output = 'Hello ' + name + ' .. !';
    }

    // 출력합니다.
    alert(output);
</script>
```

5.9 함수를 리턴하는 함수와 클로저

❖ 예제 코드 5-23]

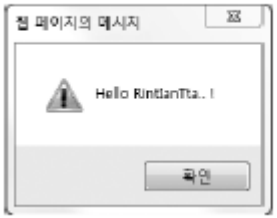
- 지역 변수 output은 함수 outerFunction을 실행할 때 생성
  - 지역 변수이므로 함수가 종료됨과 동시에 사라져야 정상
  - 코드는 그림 5-19처럼 정상적으로 실행
    - **자바스크립트 스스로 아직 지역 변수 output을 지우면 안 된다는 것을 인식하고 남겨두므로 발생하는 특성**

코드 5-23 클로저 (1)

```
<script>
    function outerFunction(name) {
        var output = 'Hello ' + name + '.. !';
        return function () {
            alert(output);
        };
    }

    outerFunction('RintIanTta')();
</script>
```

그림 5-19 실행 결과



5.9 함수를 리턴하는 함수와 클로저

❖ 클로저란?

- 지역 변수를 남겨두는 현상
- 함수 outerFunction()로 인해 생성된 공간
  - 함수 outerFunction() 내부의 변수들이 살아있음
- 리턴되는 함수 자체
- 살아남은 지역 변수 output

5.9 함수를 리턴하는 함수와 클로저

❖ 클로저의 사용

- 리턴된 클로저 함수를 사용해서만 지역 변수 output 사용 가능
- 클로저 함수로 인해 남는 지역 변수는 각각의 클로저의 고유한 변수

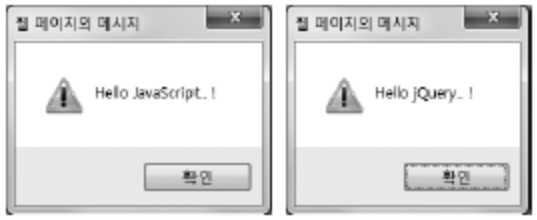
코드 5-24 클로저 (2)

```
<script>
    // 함수를 선언합니다.
    function outerFunction(name) {
        var output = 'Hello ' + name + '.. !';
        return function () {
            return output;
        };
    }

    // 변수를 선언합니다.
    var first = outerFunction('JavaScript');
    var second = outerFunction('jQuery');

    // 출력합니다.
    alert(first());
    alert(second());
</script>
```

그림 5-20 실행 결과



5.10 자바스크립트 내장 함수 (1)

❖ 내장 함수란?

- 자바스크립트에서 자체 제공하는 기본 내장 함수
- 인코딩, 디코딩과 관련된 내장 함수
  - 인코딩 - 문자를 컴퓨터에서 저장하거나 통신에 사용할 목적으로 부호화
    - **한글 같은 유니코드 문자**
  - 디코딩 - 부호화된 문자를 원래대로 되돌리는 것

표 5-2 자바스크립트의 인코딩, 디코딩과 관련된 내장 함수

함수 이름	설명
escape( )	적절한 정도로 인코딩합니다.
unescape( )	적절한 정도로 디코딩합니다.
encodeURI(uri)	최소한의 문자만 인코딩합니다.
decodeURI(encodedURI)	최소한의 문자만 디코딩합니다.
encodeURIComponent(uriComponent)	대부분의 문자를 인코딩합니다.
decodeURIComponent(encodedURI)	대부분의 문자를 디코딩합니다.

5.10 자바스크립트 내장 함수 (1)

❖ 내장 함수 간 비교

- escape()
  - 영문 알파벳, 숫자, 일부 특수 문자(@, *, -, _, +, ., /)를 제외한 모든 문자
  - 1바이트 문자는 %XX의 형태로, 2바이트 문자는 %uXXXX의 형태로 변환
- encodeURI()
  - escape() 함수에서 인터넷 주소에 사용되는 일부 특수 문자(:, ;, /, =, ?, &)는 변환하지 않음
- encodeURIComponent()
  - 알파벳과 숫자를 제외한 모든 문자 인코딩
  - UTF-8 인코딩과 같음

5.10 자바스크립트 내장 함수 (1)

❖ 출력 예제

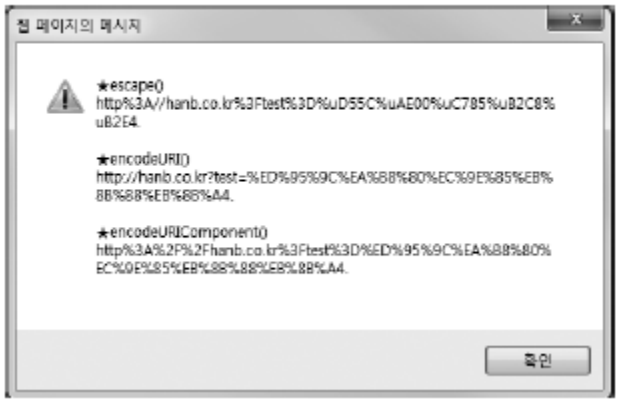
코드 5-25 인코딩 관련 내장 함수

```
<script>
    // 인코딩할 URL을 만듭니다.
    var URI = 'http://hanb.co.kr?test=한글입니다.';

    // 출력할 문자열을 만듭니다.
    var output = '';
    output += '★escape()\n'
    output += escape(URI) + '\n\n';
    output += '★encodeURI()\n'
    output += encodeURI(URI) + '\n\n';
    output += '★encodeURIComponent()\n'
    output += encodeURIComponent(URI) + '\n\n';

    // 출력합니다.
    alert(output);
</script>
```

그림 5-21 각각의 함수로 인코딩된 문자열



5.11 자바스크립트 내장 함수 (2)

❖ 자바스크립트 기본 내장 함수

표 5-3 자바스크립트의 기본 내장 함수

함수 이름	설명
eval(string)	string을 자바스크립트 코드로 실행합니다.
isFinite(number)	number가 무한한 값인지 확인합니다.
isNaN(number)	number가 NaN인지 확인합니다.
parseInt(string)	string을 정수로 바꿉니다.
parseFloat(string)	string을 유리수로 바꿉니다.

5.11 자바스크립트 내장 함수 (2)

❖ eval() 함수

- 문자열을 자바스크립트 코드로 변환해 실행하는 함수

코드 5-26 eval() 함수 (1)

```
<script>
    // eval() 함수에 사용할 문자열을 만듭니다.
    var willEval = '';
    willEval += 'var number = 10;';
    willEval += 'alert(number);';

    // eval() 함수를 사용합니다.
    eval(willEval);
</script>
```

코드 5-27 eval() 함수 (2)

```
<script>
    // eval() 함수에 사용할 문자열을 만듭니다.
    var willEval = '';
    willEval += 'var number = 10;';
    willEval += 'alert(number);';

    // eval() 함수를 사용합니다.
    eval(willEval);
    alert(number);
</script>
```

5.11 자바스크립트 내장 함수 (2)

❖ isFinite() 함수 와 isNaN() 함수

- 0으로 숫자를 나누면 자동으로 오류가 발생하며 프로그램 종료
  - 자바스크립트는 0으로 숫자를 나누면 infinity
- NaN(Not a Number)- 숫자지만 자바스크립트가 표현할 수 없는 숫자

코드 5-28 Infinity와 NaN

```
<script>
    var number1 = 10 / 0;
    var number2 = 10 / 'A';
    alert(number1 + ' : ' + number2);
</script>
```

코드를 실행하면 그림 5-22처럼 Infinity와 NaN이 출력됩니다.

그림 5-22 Infinity와 NaN을 출력

5.11 자바스크립트 내장 함수 (2)

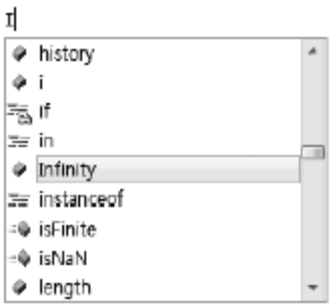
❖ 예제 코드

- isFinite() 함수는 isFinite(유한한 수)면 true 리턴
- isNaN() 함수는 isNaN이면 true 리턴
- 자바스크립트에는 Infinity, NaN이 변수로 존재

코드 5-29 숫자 관련 내장 함수

```
<script>
    var number1 = 10 / 0;
    var number2 = 10 / 'A';
    alert(isFinite(number1) + ' : ' + isNaN(number2));
</script>
```

그림 5-23 Infinity가 변수로 존재



5.11 자바스크립트 내장 함수 (2)

❖ number1 == Infinity하면 무한대?

- 무한대의 수인지 구분할 때는 꼭 isFinite() 함수 사용
- 음수를 0으로 나누면 –Infinity가 되므로 비교 불가
  - 결과는 value is not Infinity

코드 5-30 Infinity 값의 비교

```
<script>
    // 변수를 선언합니다.
    var value = -10 / 0;

    // 출력합니다.
    if (value == Infinity) {
        alert('value is Infinity');
    } else {
        alert('value is not Infinity');
    }
</script>
```

5.11 자바스크립트 내장 함수 (2)

❖ NaN은 비교문이 가능한가?

- NaN은 스스로를 비교할 수 없어 불가
- 자바스크립트는 NaN == NaN을 거짓으로 인식
  - NaN을 확인할 때에는 무조건 isNaN() 함수 사용

코드 5-31 NaN 값의 비교

```
<script>
    if (NaN == NaN) {
        alert('NaN == NaN');
    } else {
        alert('NaN != NaN');
    }
</script>
```

5.11 자바스크립트 내장 함수 (2)

❖ parseInt() 함수와 parseFloat() 함수

- 두 함수 모두 문자열을 숫자로 변경하는 함수
- Cf. Number() 함수는 숫자로 바꿀 수 없으면 무조건 NaN 리턴

코드 5-32 Number() 함수

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

코드 5-33 parseInt() 함수와 parseFloat() 함수

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

그림 5-24 parseInt() 함수(왼쪽)와 parseFloat() 함수(오른쪽)의 실행 결과

5.11 자바스크립트 내장 함수 (2)

❖ parseInt() 함수와 parseFloat() 함수 사용시 주의점

- 0으로 시작하거나 0x로 시작하면 8진수, 16진수로 생각
  - 10진수로 변환
- parseInt() 함수의 두 번째 매개 변수에 진법 입력
  - 앞의 수를 해당 진법의 수로 인식하고 10진수로 출력

```
• parseInt('0x273') ⇨ 627
• parseInt('273') ⇨ 273
• parseInt('0273') ⇨ 187

• parseInt('FF', 16) ⇨ 255
• parseInt('52', 10) ⇨ 52
• parseInt('11', 8) ⇨ 9
• parseInt('10', 2) ⇨ 2
```

- parseFloat() 함수 - 중간에 e가 들어가면 제곱으로 인식
  - parseFloat('52.273e5') ⇨ 5227300

Thank You !

모던 웹을 위한 Javascript jQuery 입문

한빛미디어 Hanbit Media, Inc.