7 기본 내장 객체

모던 웹을 위한 Javascript jQuery 입문

한빛미디어 Hanbit Media, Inc.

❖ 자바 스크립트의 기본 내장 객체의 기능

- w3schools(http://www.w3schools.com/jsref/default.asp) 사이트
  - 자바스크립트 기본 내장 객체의 속성과 메서드에 관한 설명 + 예제

그림 7-1 w3schools 사이트

7.1 기본 자료형과 객체의 차이

❖ 기본 자료형이란?

- 숫자, 문자열, 불리언

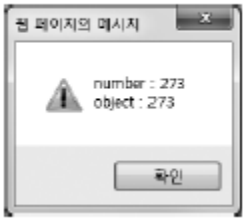
❖ 기본 자료형과 객체의 차이 예제 코드

- 기본 자료형과 객체의 자료형과 값 출력
  - 변수 primitiveNumber는 기본 자료형 숫자
  - 변수 objectNumber는 생성자 함수 사용했으므로 객체

코드 7-1 기본 자료형과 객체

```
<script>
    // 변수를 선언합니다.
    var primitiveNumber = 273;
    var objectNumber = new Number(273);

    // 출력합니다.
    var output = '';
    output += typeof (primitiveNumber) + ' : ' + primitiveNumber + '\n';
    output += typeof (objectNumber) + ' : ' + objectNumber;
    alert(output);
</script>
```

그림 7-2 자료형에서 차이

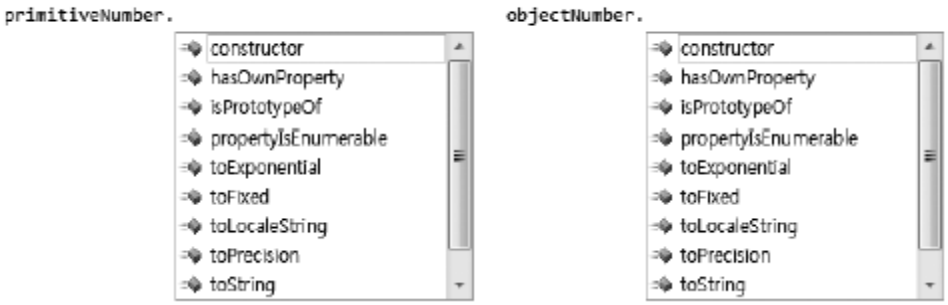


7.1 기본 자료형과 객체의 차이

❖ 기본 자료형과 객체의 차이 예제 코드

- 속성과 메서드는 객체만 가질 수 있음
- Cf. 기본 자료형의 속성이나 메서드 사용시
  - 기본 자료형이 자동 객체 변환

그림 7-3 기본 자료형과 객체의 메소드



7.1 기본 자료형과 객체의 차이

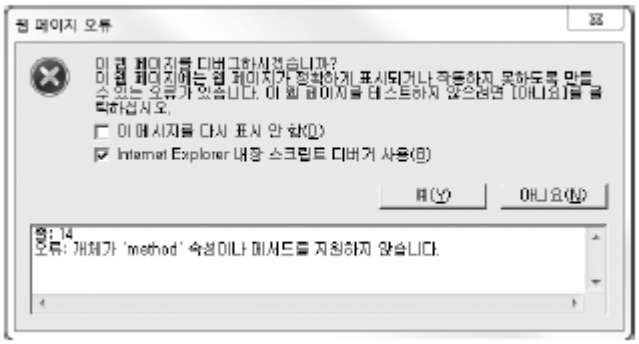
❖ 기본 자료형에 메서드 추가

- 일반적으로 1회 사용 의미. 함수에 추가하면 오류 발생

코드 7-2 기본 자료형에 메서드 추가

```
<script>
    // 변수를 만듭니다.
    var primitiveNumber = 273;

    // 메서드를 추가합니다.
    primitiveNumber.method = function () {
        return 'Primitive Method'
    };

    // 메서드를 실행합니다.
    var output = primitiveNumber.method();
    alert(output);
</script>
```

그림 7-4 웹 페이지 오류



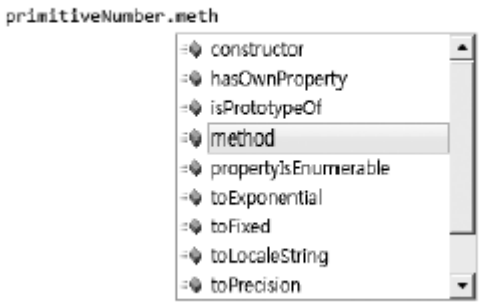
❖ 기본 자료형에 메서드 추가 (2)

- 함수의 프로토타입에 메서드 추가

코드 7-3 생성자 함수에 메서드 추가

```
<script>
    // 변수를 만듭니다.
    var primitiveNumber = 273;
    var objectNumber = new Number(273);

    // 메서드를 추가합니다.
    Number.prototype.method = function () {
        return 'Method on Prototype';
    };

    // 메서드를 실행합니다.
    var output = '';
    output += primitiveNumber.method() + '\n';
    output += objectNumber.method();
    alert(output);
</script>
```

그림 7-5 프로토타입을 사용한 기본 자료형의 메서드 추가



7.2 Object 객체 생성

❖ Object 객체

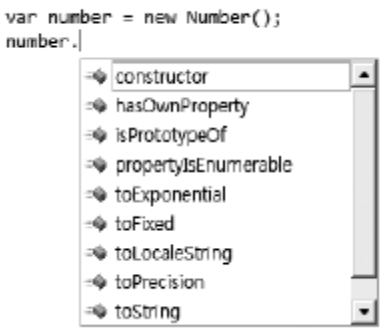
- 자바스크립트의 가장 기본적 내장 객체
- 정확히는 Object 생성자 함수로 만든 인스턴스

코드 7-4 Object 객체

```
var object = {};
var object = new Object();
```

그림 7-6 Object 객체의 속성과 메서드

그림 7-7 Number 객체의 속성과 메서드



7.3 Object 객체의 메서드

❖ Object 객체의 메서드

- hasOwnProperty() 메서드와 propertyIsEnumerable() 메서드
  - 항상 같은 값 리턴

표 7-1 Object 객체의 메서드

메서드 이름	설명
constructor( )	객체의 생성자 함수를 나타냅니다.
hasOwnProperty(name)	객체가 name 속성을 가지고 있는지 확인합니다.
isPrototypeof (object)	객체가 object의 프로토타입인지 검사합니다.
propertyIsEnumerable(name)	반복문을 사용해 열거할 수 있는지 확인합니다.
toLocaleString( )	객체를 호스트 환경에 맞는 언어의 문자열로 바꿉니다.
toString( )	객체를 문자열로 바꿉니다.
valueOf( )	객체의 값을 나타냅니다.

❖ Object 객체의 메서드

- hasOwnProperty() 메서드와 propertyIsEnumerable() 메서드 예제
  - property 속성을 검사한 것은 모두 true 출력
  - constructor 속성을 검사한 것은 모두 false 출력
  - propertyIsEnumerable() 메서드를 true로 가지는 속성만 for in 반복문 출력

코드 7-5 hasOwnProperty() 메서드와 propertyIsEnumerable() 메서드

```
<script>
    // 변수를 만듭니다.
    var object = { property: 273 };

    // 출력합니다.
    var output = '';
    output += 'HOP('property'): ' + object.hasOwnProperty('property') + '\n';
    output += 'HOP('constructor'): ' + object.hasOwnProperty('constructor') + '\n';
    output += 'PIE('property'): ' + object.propertyIsEnumerable('property') + '\n';
    output += 'PIE('constructor'): ' + object.propertyIsEnumerable('constructor');
    alert(output);

    // for in 반복문을 사용합니다.
    for (var key in object) {
        alert(object[key]);
    }
</script>
```

❖ Object 객체의 메서드

- toString() 메서드 - 객체를 문자열로 변환하는 메서드

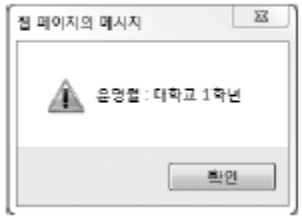
코드 7-6 toString() 메서드

```
<script>
    // 변수를 선언합니다.
    var object = new Object();

    // 출력합니다.
    alert(object);
    alert(object.toString());
</script>
```

코드 7-7 toString() 메서드 재선언

```
<script>
    // 변수를 선언합니다.
    var student = {
        name: '윤명월',
        grade: '대학교 1학년',
        toString: function () {
            return this.name + ' : ' + this.grade;
        }
    };

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

그림 7-8 toString() 메서드 자동 호출



7.4 Object 객체를 사용한 자료형 구분

❖ typeof 연산자의 문제점

코드 7-8 typeof 연산자의 문제점

```
<script>
    // 변수를 만듭니다.
    var numberFromLiteral = 273;
    var numberFromObject = new Number(273);

    // 출력합니다.
    var output = '';
    output += '1. ' + typeof (numberFromLiteral) + '\n';
    output += '2. ' + typeof (numberFromObject);
    alert(output);
</script>
```

그림 7-9 typeof 연산자를 사용한 자료형 비교

7.4 Object 객체를 사용한 자료형 구분

❖ typeof 연산자를 사용한 자료형 비교

코드 7-9 typeof 연산자를 사용한 자료형 비교

```
<script>
    // 변수를 선언합니다.
    var numberFromLiteral = 273;
    var numberFromObject = new Number(273);

    // 자료형을 확인합니다.
    if (typeof (numberFromLiteral) == 'number') {
        alert('numberFromLiteral은 숫자입니다.');
    }
    if (typeof (numberFromObject) == 'number') {
        alert('numberFromObject는 숫자입니다.');
    }
</script>
```

7.4 Object 객체를 사용한 자료형 구분

❖ constructor() 메서드 사용한 자료형 비교

코드 7-10 생성자 함수를 사용한 자료형 비교

```
<script>
    // 변수를 선언합니다.
    var numberFromLiteral = 273;
    var numberFromObject = new Number(273);

    // 자료형을 확인합니다.
    if (numberFromLiteral.constructor == Number) {
        alert('numberFromLiteral은 숫자입니다.');
    }
    if (numberFromObject.constructor == Number) {
        alert('numberFromObject는 숫자입니다.');
    }
</script>
```

7.5 Number 객체

❖ Number 객체

- 자바스크립트에서 가장 단순한 객체

코드 7-11 Number 객체의 생성

```
<script>
    // 변수를 선언합니다.
    var numberFromLiteral = 273;
    var numberFromConstructor = new Number(273);

    // 출력합니다.
    alert(typeof (numberFromLiteral));
    alert(typeof (numberFromConstructor));
</script>
```

- Object 객체의 메서드 이외에 추가 객체 가짐

표 7-2 Number 객체의 메서드

메서드 이름	설명
toExponential( )	숫자를 지수 표시로 나타내는 문자열을 만듭니다.
toFixed( )	숫자를 고정 소수점 표시로 나타낸 문자열을 만듭니다.
toPrecision( )	숫자를 길이에 따라 지수 표시 또는 고정 소수점 표시로 나타낸 문자열을 만듭니다.

7.5 Number 객체

❖ Number 객체의 메서드

코드 7-12 Number 객체의 메서드

```
<script>
    // 변수를 선언합니다.
    var number = 273.5210332;

    // 출력합니다.
    var output = '';
    output += number.toFixed(1) + '\n';
    output += number.toFixed(4);
    alert(output);
</script>
```

코드 7-13 메서드의 특이 사용

```
<script>
    // 변수를 선언합니다.
    var fixedNumber = (273.5210332).toFixed(2);

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

7.6 Number 생성자 함수의 속성

❖ 생성자 함수의 속성과 메서드 생성

코드 7-14 생성자 함수의 속성과 메서드 생성

```
<script>
    // 생성자 함수를 만듭니다.
    function Constructor(){ }
    Constructor.property = 273;
    Constructor.method = function () { };

    // 출력합니다.
    alert(Constructor.property);
</script>
```

그림 7-10 Number 생성자 함수의 속성

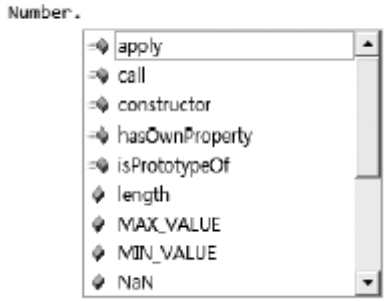


표 7-3 Number 생성자 함수의 속성

속성 이름	설명
MAX_VALUE	자바스크립트의 숫자가 나타낼 수 있는 최대 숫자
MIN_VALUE	자바스크립트의 숫자가 나타낼 수 있는 최소 숫자
NaN	자바스크립트의 숫자로 나타낼 수 없는 숫자
NEGATIVE_INFINITY	음의 무한대 숫자
POSITIVE_INFINITY	양의 무한대 숫자

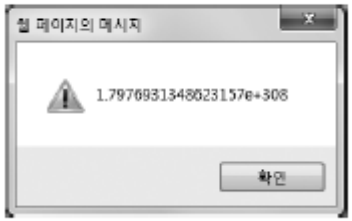
7.6 Number 생성자 함수의 속성

❖ Number 객체의 MAX_VALUE 속성

코드 7-15 Number 객체의 MAX_VALUE 속성

```
<script>
    // 변수를 선언합니다.
    var number = Number.MAX_VALUE + 1;

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

그림 7-11 MAX_VALUE



코드 7-16 Number.MAX_VALUE에서 Infinity로 변환

```
<script>
    // 변수를 선언합니다.
    var addNumber = Number('0.0000000000000001e+308');
    var number = Number.MAX_VALUE + addNumber;

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

7.7 String 객체

❖ String 객체

- 자바스크립트에서 가장 많이 사용하는 내장 객체

코드 7-17 String 객체의 생성

```
<script>
    // 변수를 선언합니다.
    var stringFromLiteral = 'Hello World..!';
    var stringFromConstructor = new String('Hello World..!');

    // 변수의 자료형을 출력합니다.
    var output = '';
    output += typeof (stringFromLiteral) + '\n';
    output += typeof (stringFromConstructor);
    alert(output);
</script>
```

7.7 String 객체

❖ String 객체의 length 속성

표 7-4 String 객체의 속성

속성 이름	설명
length	문자열의 길이를 나타냅니다.

코드 7-18 length 속성

```
<script>
    // 변수를 선언합니다.
    var characters = prompt('사용할 비밀번호를 입력해주세요.', '6글자 이상');

    // 출력합니다.
    if (characters.length < 6) {
        alert('6글자 이상으로 입력하세요.');
    } else {
        alert('잘했어요!');
    }
</script>
```

7.8 String 객체의 기본 메서드

❖ String 객체 메서드

- 기본 메서드와 HTML 관련 메서드로 구분
- 기본 메서드

표 7-5 String 객체의 메서드

메서드 이름	설명
charAt(position)	position에 위치하는 문자를 리턴합니다.
charCodeAt(position)	position에 위치하는 문자의 유니코드 번호를 리턴합니다.
concat(string, ... , string)	매개 변수로 입력한 문자열을 이어 리턴합니다.
indexOf(searchString, position)	앞에서부터 일치하는 문자열의 위치를 리턴합니다.
lastIndexOf(searchString, position)	뒤에서부터 일치하는 문자열의 위치를 리턴합니다.
match(regExp)	문자열 내에 regExp가 있는지 확인합니다.
replace(regExp, replacement)	regExp를 replacement로 바꾼 뒤 리턴합니다.
search(regExp)	regExp와 일치하는 문자열의 위치를 리턴합니다.
slice(start, end)	특정 위치의 문자열을 추출해 리턴합니다.
split(separator, limit)	separator로 문자열을 잘라 배열을 리턴합니다.
substr(start, count)	start부터 count만큼 문자열을 잘라서 리턴합니다.
substring(start, end)	start부터 end까지 문자열을 잘라서 리턴합니다.
toLowerCase( )	문자열을 소문자로 바꿔 리턴합니다.
toUpperCase( )	문자열을 대문자로 바꿔 리턴합니다.

7.8 String 객체의 기본 메서드

❖ 잘못된 String 객체 메서드 사용

코드 7-19 잘못된 String 객체의 메서드 사용

```
<script>
    // 변수를 선언합니다.
    var string = 'abcdefg';

    // 출력합니다.
    string.toUpperCase();
    alert(string);
</script>
```

그림 7-12 String 객체의 메서드의 잘못된 사용

- 자기 자신을 변화시키지 않고 리턴
  - 문자열을 대문자로 변화시키고 싶으면 리턴값 사용.

코드 7-20 올바른 String 객체의 메서드 사용

```
<script>
    // 변수를 선언합니다.
    var string = 'abcdefg';

    // 출력합니다.
    string = string.toUpperCase();
    alert(string);
</script>
```

7.9 String 객체의 HTML 관련 메서드

❖ String 객체의 HTML 관련 메서드

표 7-6 String 객체의 HTML관련 메서드

메서드 이름	설명
anchor( )	a 태그로 문자열을 감싸 리턴합니다.
big( )	big 태그로 문자열을 감싸 리턴합니다.
blink( )	blink 태그로 문자열을 감싸 리턴합니다.
bold( )	b 태그로 문자열을 감싸 리턴합니다.
fixed( )	tt 태그로 문자열을 감싸 리턴합니다.
fontcolor(colorString)	font 태그로 문자열을 감싸고 color 속성을 주어 리턴합니다.
fontsize(fontSize)	font 태그로 문자열을 감싸고 size 속성을 주어 리턴합니다.
italics( )	I 태그로 문자열을 감싸 리턴합니다.
link(linkRef)	a 태그에 href 속성을 지정해 리턴합니다.
small( )	small 태그로 문자열을 감싸 리턴합니다.
strike( )	strike 태그로 문자열을 감싸 리턴합니다.
sub( )	sub 태그로 문자열을 감싸 리턴합니다.
sup( )	sup 태그로 문자열을 감싸 리턴합니다.

7.9 String 객체의 HTML 관련 메서드

❖ String 객체의 HTML 관련 메서드

- Write() 메서드 사용

코드 7-21 String 객체의 HTML 관련 메서드

```
<script>
    // 문자열을 만듭니다.
    var string = 'JavaScript';
    var output = '';
    output += 'anchor: ' + string.anchor() + '<br/>';
    output += 'big: ' + string.big() + '<br/>';
    output += 'blink: ' + string.blink() + '<br/>';
    output += 'bold: ' + string.bold() + '<br/>';
    output += 'fixed: ' + string.fixed() + '<br/>';
    output += 'string: ' + string.fontcolor('Red') + '<br/>';
    output += 'fontsize: ' + string.fontsize(15) + '<br/>';
    output += 'italics: ' + string.italics() + '<br/>';
    output += 'link: ' + string.link('http://hanb.co.kr') + '<br/>';
    output += 'small: ' + string.small() + '<br/>';
    output += 'strike: ' + string.strike() + '<br/>';
    output += 'sub: ' + string.sub() + '<br/>';
    output += 'sup: ' + string.sup() + '<br/>';
    // 출력합니다.
    document.write(output);
</script>
```

그림 7-13 String 객체의 HTML 관련 메서드

anchor: JavaScript
big: JavaScript
blink: JavaScript
bold: **JavaScript**
fixed: JavaScript
string: JavaScript
fontsize: JavaScript
italics: *JavaScript*
link: JavaScript
small: JavaScript
strike: ~~JavaScript~~
sub: JavaScript
sup: JavaScript

7.10 메서드 체이닝

❖ 메서드 체이닝

- 연속해 메서드 사용

코드 7-22 기본 메서드 사용

```
<script>
    // 변수를 선언합니다.
    var output = 'Hello World .. !';

    // 메서드를 사용합니다.
    output = output.toLowerCase();
    output = output.substring(0, 10);
    output = output.anchor('name');

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

코드 7-23 메서드 체이닝

```
<script>
    // 변수를 선언합니다.
    var output = 'Hello World .. !';

    // 메서드 체이닝
    output = output.toLowerCase().substring(0,10).anchor('name');

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

7.11 Array 객체의 생성

❖ Array 객체의 생성

표 7-7 Array 생성자 함수

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

코드 7-24 Array 객체

```
<script>
    // 변수를 선언합니다.
    var array1 = [52, 273, 103, 57, 32];
    var array2 = new Array();
    var array3 = new Array(10);
    var array4 = new Array(52, 273, 103, 57, 32);
</script>
```

7.12 Array 객체의 속성과 메서드

❖ Array 객체의 속성과 메서드

- 배열은 몇 개의 요소를 가지고 있는지 나타내는 length 속성 가짐

표 7-8 Array 객체의 속성

속성 이름	설명
length	배열 요소의 개수를 알아냅니다.

코드 7-25 length 속성

```
<script>
    // 변수를 선언합니다.
    var array = ['A', 'B', 'C', 'D'];

    // 출력합니다.
    var output = '';
    for (var i = 0; i < array.length; i++) {
        output += i + ' : ' + array[i] + '\n';
    }
    alert(output);
</script>
```

7.12 Array 객체의 속성과 메서드

❖ Array 객체 메서드

표 7-9 Array 객체의 메서드

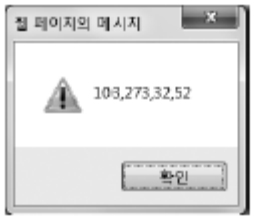
메서드 이름	설명
concat( )	매개 변수로 입력한 배열의 요소를 모두 합쳐 배열을 만들어 리턴합니다.
join( )	배열 안의 모든 요소를 문자열로 만들어 리턴합니다.
pop( )*	배열의 마지막 요소를 제거하고 리턴합니다.
push( )*	배열의 마지막 부분에 새로운 요소를 추가합니다.
reverse( )*	배열의 요소 순서를 뒤집습니다.
slice( )	배열 요소의 지정한 부분을 리턴합니다.
sort( )*	배열의 요소를 정렬하고 리턴합니다.
splice( )*	배열 요소의 지정한 부분을 삭제하고 삭제한 요소를 리턴합니다.

❖ Sort() 메서드 사용

코드 7-26 Array 객체의 메서드를 사용할 때 주의할 점

```
<script>
    // 변수를 선언하고 정렬합니다.
    var array = [52, 273, 103, 32];
    array.sort();

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

그림 7-14 문자열 오름차순 정렬



7.13 Array 객체의 정렬

❖ Array 객체의 sort() 메서드의 정렬

- 변화를 주고 싶을 때는 sort() 메서드의 매개 변수로 함수 넣음
- 함수 이용해 퀵소트나 머지 소트 같은 정렬 진행

```
array.sort(function (left, right) {

});
```

• 오름차순 정렬

```
function (left, right) {
    return left - right;
}
```

• 내림차순 정렬

```
function (left, right) {
    return right - left;
}
```

7.13 Array 객체의 정렬

❖ 오름차순 정렬 예제

코드 7-27 sort() 메서드의 정렬 방식 지정

```
<script>
    // 변수를 선언하고 정렬합니다.
    var array = [52, 273, 103, 32];
    array.sort(function (left, right) {
        return left - right;
    });

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

7.13 Array 객체의 정렬

❖ 학생 성적 정렬 예제

- 내림차순 정렬해 1등부터 3등까지 출력

코드 7-28 학생 성적 정렬

```
<script>
    // 생성자 함수를 선언합니다.
    function Student(name, korean, math, english, science) {
        // 속성
        this.이름 = name;
        this.국어 = korean;
        this.수학 = math;
        this.영어 = english;
        this.과학 = science;

        // 메서드
        this.getSum = function () {
            return this.국어 + this.수학 + this.영어 + this.과학;
        };
        this.getAverage = function () {
            return this.getSum() / 4;
        };
        this.toString = function () {
            return this.이름 + '\t' + this.getSum() + '\t' + this.getAverage();
        };
    }
```

7.13 Array 객체의 정렬

❖ 학생 성적 정렬 예제 (2)

```
    // 학생 정보 배열을 만듭니다.
    var students = [];
    students.push(new Student('윤하린', 96, 98, 92, 98));
    /* 생략 */
    students.push(new Student('최지웅', 92, 94, 88, 98));

    // 정렬하고 1등부터 3등까지만 배열에 남겨둡니다.
    students.sort(function (left, right) {
        return right.getSum() - left.getSum();
    });
    students = students.slice(0, 3);

    // 출력합니다.
    var output = '이름\t총점\t평균\n';
    for (var i in students) {
        output += students[i].toString() + '\n';
    }
    alert(output);
</script>
```

7.13 Array 객체의 정렬

❖ Sort() 메서드와 Slice() 메서드

- sort() 메서드의 매개 변수로 총점을 내림차순 정렬
- 0번째 인덱스부터 세 개만 변수 students에 다시 할당

코드 7-29 slice() 메서드

```
// 정렬하고 1등부터 3등까지만 배열에 남겨둡니다.
students.sort(function (left, right) {
    return right.getSum() - left.getSum();
});
students = students.slice(0, 3);
```

그림 7-15 sort() 메서드와 slice() 메서드의 사용

코드 7-30 Array 객체 메서드의 체이닝

```
// 정렬하고 1등부터 3등까지만 배열에 남겨둡니다.
students = students.sort(function (left, right) {
    return right.getSum() - left.getSum();
}).slice(0, 3);
```

7.14 Array 객체의 요소 제거

❖ Array 객체의 remove( ) 메서드

코드 7-31 Array 객체의 remove() 메서드

```
// Array생성자 함수의 프로토타입에 remove() 메서드를 추가합니다.
Array.prototype.remove = function (index) {
    this.splice(index, 1);
}
```

코드 7-32 잘못된 Array 객체의 요소 제거

```
<script>
    // Array 생성자 함수의 프로토타입에 remove() 메서드를 추가합니다.
    Array.prototype.remove = function (index) { this.splice(index, 1); }

    // 변수를 선언합니다.
    var array = [52, 273, 103, 32, 274, 129];

    // 반복문과 조건문을 사용해 100보다 큰 요소를 제거합니다.
    for (var i = 0; i < array.length; i++) {
        if (array[i] > 100) {
            array.remove(i);
        }
    }

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

그림 7-16 배열 요소를 잘못된 방법으로 제거한 결과

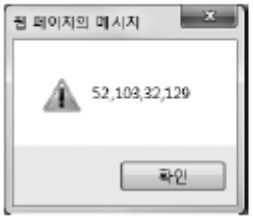
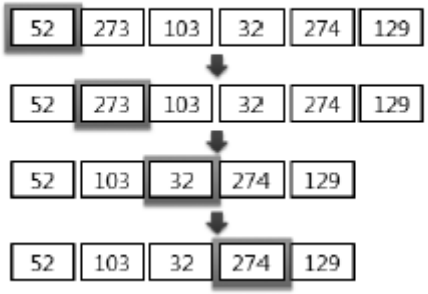


그림 7-17 잘못된 Array 객체의 요소 제거



7.14 Array 객체의 요소 제거

❖ Array 객체의 remove( ) 메서드

코드 7-33 올바른 Array 객체의 요소 제거

```
<script>
    // Array 생성자 함수의 프로토타입에 remove() 메서드를 추가합니다.
    Array.prototype.remove = function (index) { this.splice(index, 1); }

    // 변수를 선언합니다.
    var array = [52, 273, 103, 32, 274, 129];

    // 반복문과 조건문을 사용해 100보다 큰 요소를 제거합니다.
    for (var i = array.length; i >= 0; i--) {
        if (array[i] > 100) {
            array.remove(i);
        }
    }

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

그림 7-18 배열 요소를 올바른 제거한 결과

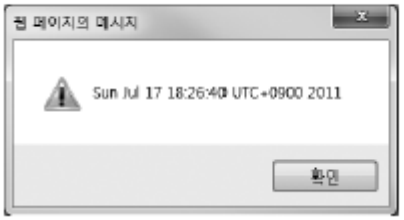
7.15 Date 객체의 생성

❖ Date 객체의 생성과 출력

코드 7-34 Date 객체의 생성과 출력

```
<script>
    // 변수를 선언합니다.
    var date = new Date();

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

그림 7-19 Date 객체를 사용한 현재 시간 구하기



코드 7-35 문자열을 사용한 Date 객체 생성

```
var date = new Date('December 9');
var date = new Date('December 9, 1991');
var date = new Date('December 9, 1991 02:24:23');
```

코드 7-36 숫자를 사용한 Date 객체 생성

```
var date = new Date(1991, 12, 9);
var date = new Date(1991, 12, 9, 2, 24, 23);
var date = new Date(1991, 12, 9, 2, 24, 23, 1);
```

코드 7-37 Unix Time을 사용한 Date 객체 생성

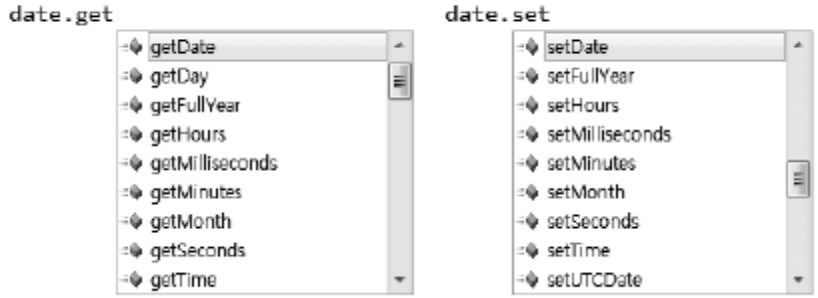
```
var date = new Date(2732741033257);
```

7.16 Date 객체의 메서드

❖ Date 객체의 메서드

- 게터와 세터

그림 7-20 Date 객체의 게터와 세터



- 일주일 후의 시간 구하기

코드 7-38 일주일 후의 시간 구하기

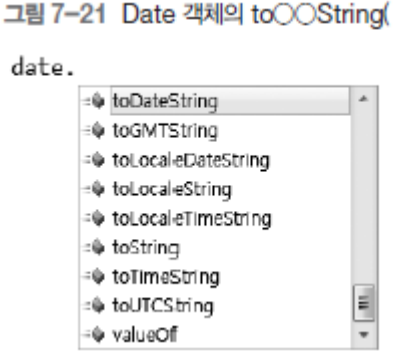
```
<script>
    // 변수를 선언합니다.
    var date = new Date();

    // 현재 시간에서 7일을 더합니다.
    date.setDate(date.getDate() + 7);

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

7.16 Date 객체의 메서드

❖ Date 객체의 메서드

- to○○String() 형태 메서드

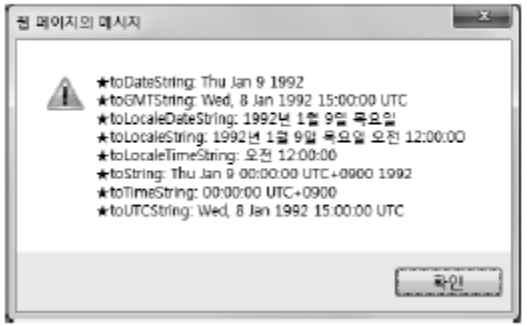
그림 7-21 Date 객체의 to○○String() 메서드



코드 7-39 Date 객체의 to○○String() 메서드

```
<script>
    // 변수를 생성합니다.
    var date = new Date(1991, 12, 9);
    // 출력합니다.
    var output = '';
    output += '★toDateString: ' + date.toDateString() + '\n';
    output += '★toGMTString: ' + date.toGMTString() + '\n';
    output += '★toLocaleDateString: ' + date.toLocaleDateString() + '\n';
    output += '★toLocaleString: ' + date.toLocaleString() + '\n';
    output += '★toLocaleTimeString: ' + date.toLocaleTimeString() + '\n';
    output += '★toString: ' + date.toString() + '\n';
    output += '★toTimeString: ' + date.toTimeString() + '\n';
    output += '★toUTCString: ' + date.toUTCString();
    alert(output);
</script>
```

그림 7-22 각각 to○○String() 메서드의 실행 결과



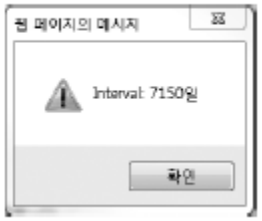
7.17 Date 객체를 사용한 시간 간격 구하기

❖ Date 객체를 사용한 시간 간격 구하기

코드 7-40 날짜 간격을 구하는 방법

```
<script>
    // 변수를 선언합니다.
    var now = new Date();
    var before = new Date('December 9, 1991');
    // 날짜 간격을 구합니다.
    var interval = now.getTime() - before.getTime();
    interval = Math.floor(interval / (1000 * 60 * 60 * 24));

    // 출력합니다.
    alert('Interval: ' + interval + '일');
</script>
```

그림 7-23 날짜 간격



7.17 Date 객체를 사용한 시간 간격 구하기

❖ Date 객체를 사용한 시간 간격 구하기 (2)

- 프로토 타입에 메서드 넣어 쉽게 날짜 간격 구하는 예제

코드 7-41 프로토타입에 날짜 간격을 구하는 메서드 추가

```
<script>
    // Date생성자 함수의 프로토타입에 메서드를 추가합니다.
    Date.prototype.getInterval = function (otherDate) {
        // 변수를 선언합니다.
        var interval;

        // 양수로 날짜 간격을 구하려고 조건문을 사용합니다.
        if (this> otherDate) {
            interval = this.getTime() - otherDate.getTime();
        } else {
            interval = otherDate.getTime() - this.getTime();
        }

        // 리턴합니다.
        return Math.floor(interval / (1000 * 60 * 60 * 24));
    };

    // 변수를 선언합니다.
    var now = new Date();
    var before = new Date('December 9, 1991');

    // 출력합니다.
    alert('Interval: ' + now.getInterval(before) + '일');
</script>
```

7.18 Math 객체

❖ Math 객체

- 자바스크립트의 기본 내장 객체 중 유일하게 생성자 함수 미사용 객체
  - 변수로 표시

그림 7-24 Math 객체는 변수

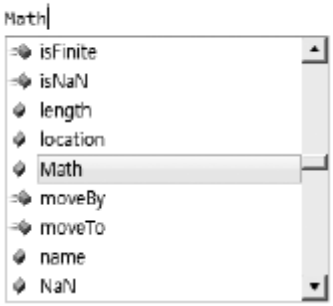


표 7-10 Math 객체의 속성

속성 이름	값
E	2.718281828459045
LN2	0.6931471805599453
LN10	2.302585092994046
LOG2E	1.4426950408889633
LOG10E	0.4342944819032518
PI	3.141592653589793
SQRT1_2	0.7071067811865476
SQRT2	1.4142135623730951

7.18 Math 객체

❖ Math 객체의 사용

- 생성자 함수가 아니라 그냥 객체

코드 7-42 Math 객체

```
<script>
    alert(Math.E);
    alert(Math.LN2);
    alert(Math.LN10);
    alert(Math.LOG2E);
    alert(Math.LOG10E);
    alert(Math.PI);
    alert(Math.SQRT1_2);
    alert(Math.SQRT2);
</script>
```

7.18 Math 객체

❖ Math 객체의 메서드

표 7-11 Math 객체의 메서드

메서드 이름	설명
abs(x)	x의 절대 값을 구합니다.
acos(x)	x의 아크 코사인 값을 구합니다.
asin(x)	x의 아크 사인 값을 구합니다.
atan(x)	x의 아크 탄젠트 값을 구합니다.
atan2(y, x)	x와 y의 비율로 아크 탄젠트 값을 구해 구합니다.
ceil(x)	x보다 크거나 같은 가장 작은 정수를 구합니다.
cos(x)	x의 코사인 값을 구합니다
exp(x)	자연 로그의 x 제곱을 구합니다.
floor(x)	x보다 작거나 같은 가장 큰 정수를 구합니다.
log(x)	x의 로그 값을 구합니다.
max(x,y,z,…,n)	매개 변수 중 가장 큰 값을 구합니다.
min(x,y,z,…,n)	매개 변수 중 가장 작은 값을 구합니다.
pow(x,y)	x의 y 제곱을 구합니다.
random( )	0부터 1까지의 임의의 수를 구합니다.
round(x)	x를 반올림하여 구합니다.
sin(x)	x의 사인 값을 구합니다.
squt(x)	x의 제곱근을 구합니다.
tan(x)	x의 탄젠트 값을 구합니다.

7.18 Math 객체

❖ Math 객체의 함수화

- 자바스크립트의 함수는 하나의 자료형
  - 변수에 저장 가능 → 함수처럼 사용

코드 7-43 메서드의 함수화

```
<script>
    // 변수를 선언합니다.
    var findMax = Math.max;

    // 출력합니다.
    alert(findMax(52, 273, 103, 57, 32));
</script>
```

7.19 ECMAScript 5 Array 생성자 함수의 메서드

❖ ECMAScript 5에서 추가된 Array 객체의 메서드

표 7-12 Array 생성자 함수의 메서드

메서드 이름	설명
Array.isArray( )	배열인지 확인합니다.

- Cf. typeof 키워드를 사용해서 Array 객체의 자료형 확인
  - 결과는 문자열 'object'

코드 7-44 Array.isArray( ) 메서드

```
<script>
    // 출력합니다.
    alert(Array.isArray([1, 2, 3]));
    alert(Array.isArray({}));
    alert(Array.isArray(1));
</script>
```

7.20 ECMAScript 5 Array 객체 (1)

❖ ECMAScript 5 Array 추가 객체

표 7-13 ECMAScript 5 Array 객체의 메서드(1)

메서드 이름	설명
filter( )	특정 조건을 만족하는 요소를 추출해 새로운 배열을 만듭니다.
forEach( )	배열의 각각의 요소를 사용해 특정 함수를 for in 반복문처럼 실행합니다.
every( )	배열의 요소가 특정 조건을 모두 만족하는지 확인합니다.
map( )	기존의 배열에 특정 규칙을 적용해 새로운 배열을 만듭니다.
some( )	배열의 요소가 특정 조건을 적어도 하나 만족하는지 확인합니다.
indexOf( )	특정 요소를 앞쪽부터 검색합니다.
lastIndexOf( )	특정 요소를 뒤쪽부터 검색합니다.

7.20 ECMAScript 5 Array 객체 (1)

❖ indexOf( ) 메서드와 lastIndexOf( ) 메서드

- 내부에 검색하려는 요소 있으면 해당 요소가 위치하는 인덱스 리턴
- 없으면 -1 리턴

코드 7-45 indexOf() 메서드와 lastIndexOf() 메서드

```
<script>
    // 변수를 선언합니다.
    var array = [1, 2, 3, 4, 5, 5, 4, 3, 2, 1];

    // 메서드를 사용합니다.
    var output1 = array.indexOf(4);
    var output2 = array.indexOf(8);
    var output3 = array.lastIndexOf(4);
    var output4 = array.lastIndexOf(8);

    // 출력합니다.
    var output = '';
    output += output1 + ' : ' + output2 + '\n';
    output += output3 + ' : ' + output4;
    alert(output);
</script>
```

그림 7-25 실행 결과

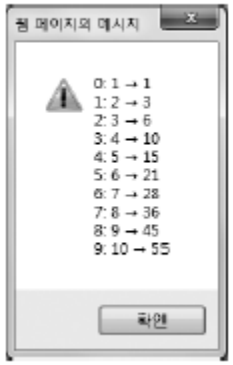
7.20 ECMAScript 5 Array 객체 (1)

❖ forEach( ) 메서드

코드 7-46 forEach() 메서드

```
<script>
    // 변수를 선언합니다.
    var array = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10];

    // 메서드를 사용합니다.
    var sum = 0;
    var output = '';

    array.forEach(function (element, index, array) {
        sum += element;
        output += index + ': ' + element + ' -> ' + sum +'\n' ;
    });

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

그림 7-26 forEach() 메서드



7.20 ECMAScript 5 Array 객체 (1)

❖ filter( ) 메서드

코드 7-47 filter() 메서드

```
<script>
    // 변수를 선언합니다.
    var array = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10];

    // 메서드를 사용합니다.
    array = array.filter(function (element, index, array) {
        return element <= 5;
    });

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

그림 7-27 실행 결과

7.20 ECMAScript 5 Array 객체 (1)

❖ every( ) 메서드와 some( ) 메서드

코드 7-48 every() 메서드와 some() 메서드

```
<script>
    // 변수를 선언합니다.
    var array = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10];

    // 함수를 선언합니다.
    function lessThanFive(element, index, array) {
        return element < 5;
    }

    function lessThanTwenty(element, index, array) {
        return element < 20;
    }

    // 메서드를 사용합니다.
    var output1 = array.every(lessThanFive);
    var output2 = array.every(lessThanTwenty);
    var output3 = array.some(lessThanFive);
    var output4 = array.some(lessThanTwenty);

    // 출력합니다.
    var output = ''
    output += output1 + ' : ' + output2 + '\n';
    output += output3 + ' : ' + output4;
    alert(output);
</script>
```

그림 7-28 실행 결과

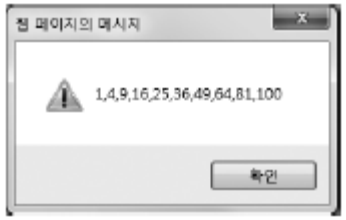
7.20 ECMAScript 5 Array 객체 (1)

❖ map( ) 메서드

코드 7-49 map( ) 메서드

```
<script>
    // 변수를 선언합니다.
    var array = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10];

    // 메서드를 사용합니다.
    var output = array.map(function (element) {
        return element * element;
    });

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

그림 7-29 실행 결과



7.21 ECMAScript 5 Array 객체 (2)

❖ ECMAScript 5 Array 객체 메서드

표 7-14 ECMAScript 5 Array 객체의 메서드 (2)

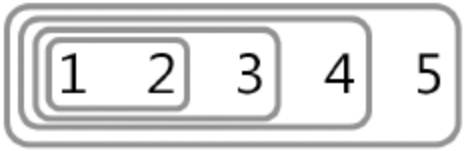
메서드 이름	설명
reduce( )	배열의 요소가 하나가 될 때까지 요소를 왼쪽부터 두 개씩 묶는 함수를 실행합니다.
reduceRight( )	배열의 요소가 하나가 될 때까지 요소를 오른쪽부터 두 개씩 묶는 함수를 실행합니다.

코드 7-50 reduce( ) 메서드

```
<script>
    // 변수를 선언합니다.
    var array = [1, 2, 3, 4, 5];

    // 출력합니다.
    var output = '';
    array.reduce(function (previousValue, currentValue, index, array) {
        output += previousValue + ' : ' + currentValue + ' : ' + index + '\n';
    });
    alert(output);
</script>
```

그림 7-30 reduce( ) 메서드와 reduceRight( ) 메서드

그림 7-31 reduce( ) 메서드의 실행 단계



7.21 ECMAScript 5 Array 객체 (2)

❖ ECMAScript 5 Array 객체 메서드

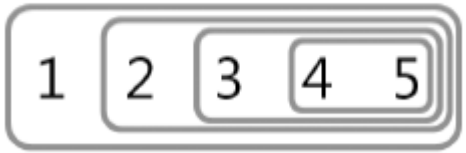
- Reduce() 메서드 활용

코드 7-51 reduce( ) 메서드 활용

```
<script>
    // 변수를 선언합니다.
    var array = [1, 2, 3, 4, 5];

    // 출력합니다.
    var result = array.reduce(function (previousValue, currentValue, index,
        array) {
        return previousValue + currentValue;
    });
    alert(result);
</script>
```

그림 7-32 실행 결과

그림 7-33 reduceRight( ) 메서드의 실행 단계



7.22 ECMAScript 5의 JSON 객체와 Date 객체

❖ JSON (JavaScript Object Notation) 이란?

- ECMAScript 5부터는 정식으로 JSON 객체 지원
- 자바스크립트 객체의 형태를 가지는 문자열의 의미

표 7-15 ECMAScript 5의 JSON 객체

메서드 이름	설명
JSON.parse( )	JSON 형식의 문자열을 자바스크립트 객체로 만듭니다.
JSON.stringify( )	자바스크립트 객체를 JSON 형식의 문자열로 만듭니다.

코드 7-52 JSON.stringify( ) 메서드

```
<script>
    // 변수를 선언합니다.
    var object = {
        name: 'RintIanTta',
        gender: 'Male'
    };

    // 출력합니다.
    alert(JSON.stringify(object));
</script>
```

그림 7-34 stringify( ) 메서드

7.22 ECMAScript 5의 JSON 객체와 Date 객체

❖ JSON.parse( ) 메서드

- JSON 문자열을 자바스크립트 객체로 바꾸는 역할 수행

코드 7-53 JSON.parse( ) 메서드

```
<script>
    // 변수를선언합니다.
    var object = {
        name: 'RintIanTta',
        gender: 'Male'
    };
    var copy = JSON.parse(JSON.stringify(object));

    // 출력합니다.
    alert(copy.name + ' : ' + copy.gender);
</script>
```

7.22 ECMAScript 5의 JSON 객체와 Date 객체

❖ toJSON( ) 메서드

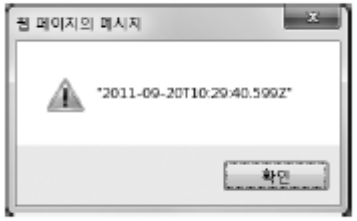
표 7-16 Date 객체의 메서드

메서드 이름	설명
toJSON()	JSON 문자열로 변경합니다.

코드 7-54 toJSON() 메서드

```
<script>
    // 변수를선언합니다.
    var date = new Date();

    // 출력합니다.
    alert(JSON.stringify(date.toJSON()));
</script>
```

그림 7-35 실행 결과



7.23 ECMAScript 5의 String 객체

❖ trim() 메서드

- ECMAScript 5부터는 String 객체가 trim() 메서드 지원
- 문자열의 양쪽 끝의 공백 제거하는 메서드

표 7-17 String 객체의 메서드

메서드 이름	설명
trim( )	문자열의 양 끝 공백을 제거합니다.
trimLeft( )	문자열의 왼쪽 끝의 공백을 제거합니다.
trimRight( )	문자열의 오른쪽 끝의 공백을 제거합니다.

코드 7-55

```
<script>
    // 변수를 선언합니다.
    var string = ' Hello World .. !  ';

    // 출력합니다.
    alert('"' + string.trim() + '"');
</script>
```

그림 7-36 trim() 메서드 실행 결과

7.24 HTML5 Audio 객체

❖ HTML5 Audio 객체

- HTML5부터는 audio 태그 사용해 쉽게 음악 재생
- 자바스크립트를 사용해서도 음악 재생 - Audio 객체 사용

그림 7-37 폴더 구성

코드 7-56 Audio 객체 생성(1)

```
<script>
    var audio = new Audio('Kalimba.mp3');
</script>
```

코드 7-57 Audio 객체 생성(2)

```
<script>
    var audio = new Audio();
    audio.src = 'Kalimba.mp3';
</script>
```

7.24 HTML5 Audio 객체

❖ HTML5 Audio 객체의 속성과 메서드

표 7-18 Audio 객체의 기본적인 속성

속성 이름	설명
src	재생하려는 음악 파일 경로
volume	음악의 음량
currentTime	음악의 현재 위치(초 단위)

표 7-19 Audio 객체의 기본적인 메서드

메서드 이름	설명
play( )	음악을 재생합니다.
pause( )	음악을 일시 정지합니다.

코드 7-58 간단한 음악 재생 프로그램

```
<!DOCTYPE html>
<html>
<head>
<script>
    // 변수를 선언합니다.
    var audio = new Audio('Kalimba.mp3');
</script>
</head>
<body>
    <button onclick="audio.play()">PLAY</button>
    <button onclick="audio.pause()">PAUSE</button>
    <input type="number" onchange="audio.currentTime=this.value" />
    <input type="number" onchange="audio.volume=this.value" />
</body>
</html>
```

그림 7-38 실행 결과

Thank You !

모던 웹을 위한 Javascript jQuery 입문

한빛미디어 Hanbit Media, Inc.