**Ch08 기본내장객체**

**w**3schools(https://www.w3schools.com/jsref) 자바스크립트의 기본 내장객체의 속성과 메소드

devdocs.io(http://devdocs.io/javascript ) 각각의 속성, 메소드에 대한 자세한 정보 제공

**Number : 숫자 객체**

**String : 문자열을 다루기 이한 객체**

**Array : 배열을 만들어 주는 객체**

**Date : 날짜와 시간을 처리하기 위한 객체**

**Math : 수학 계산을 위한 객체**

8.1 기본 자료형과 객체의 차이점

* 기본 자료형은 여섯 가지 자료형 중 숫자, 문자열, 불

<script>

var primitiveNumber = 273;

var objectNumber = new Number(273);

var output = '';

output += typeof (primitiveNumber) + ' : ' + primitiveNumber + '\n';

output += typeof (objectNumber) + ' : ' + objectNumber;

alert(output);

</script>

* 기본 자료형과 객체는 자료형이 분명 다르지만 두가지 모두 값을 출력한다. 각자의 속성과 메소드도 있다. 속성과 메소드는 객체만이 가질 수 있지만 기본 자료형에도 속성과 메서드가 있다. 기본 자료형의 속성이나 메소드를 사용하면 기본 자료형이 자동으로 객체로 변환된다. 이번 장에서 Number 객체라 부르면 객체와 기본 자료형 모두를 의미한다고 생각하세요. 굳이 차이점을 찾자면 기본 자료형은 객체가 아니므로 속성과 메서드를 추가할 수 없습니다. 기본 자료형에 메소드를 추가하고자 한다면 Number객체의 프로토타입에 메서드를 추가하면 기본 자료형에게도 새로운 메서드가 생성된다.

<script>

var primitiveNumber = 273;

var objectNumber = new Number(273);

Number.prototype.method = function () {

return 'Method on prototype';

};

var output = '';

output += primitiveNumber.method() + '\n';

output += objectNumber.method();

alert(output);

</script>

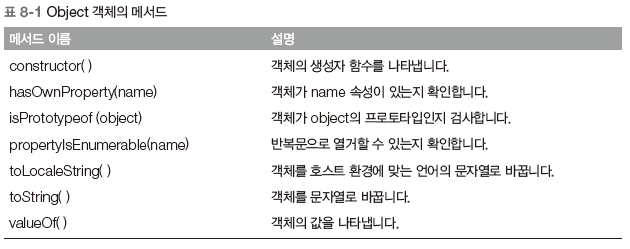
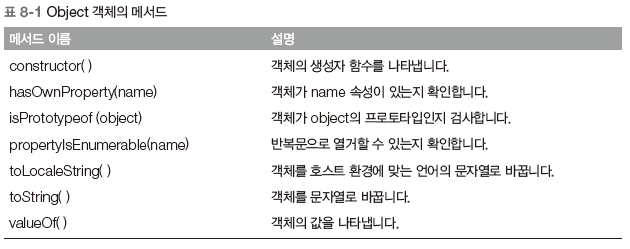
8.2 Object 객체 ; 자바 스크립트의 최상위 객체

8.2.1 생성

var object = {};

var object = new Object();

8.2.2 속성과 메소드



<script>

var object = { property: 273 };

var output = '';

output += "hasOwnProperty('property'): " + object.hasOwnProperty('property') + '\n';

output += "hasOwnProperty('constructor'): " + object.hasOwnProperty('constructor') + '\n';

output += "반복문으로 열거할 수 있는지('property'): " + object.propertyIsEnumerable('property') + '\n';

output += "반복문으로 열거할 수 있는지('constructor'): " + object.propertyIsEnumerable('constructor');

output += object.toString();

alert(output);

for (var key in object) {

alert(object[key]);

}

</script>

8.2.3 자료형 구분

* 자료형을 검사할 때는 typeof 연산자를 사용한다. 그러나 다음의 경우 자료형 검사에 문제가 발생한다.

<script>

var numberFromLiteral = 273;

var numberFromObject = new Number(273);

if (typeof (numberFromLiteral) == 'number') {

alert('numberFromLiteral은 숫자입니다.');

}

if (typeof (numberFromObject) == 'number') {

alert('numberFromObject는 숫자입니다.');

}

</script>

* 위의 경우 두 대상을 같은 자료형으로 취급하고 싶을 때 constructor()메서드를 사용한다. Constructor()메소드로 자료형을 비교한다.

<script>

var numberFromLiteral = 273;

var numberFromObject = new Number(273);

if (numberFromLiteral.constructor == Number) {

alert('numberFromLiteral은 숫자입니다.');

}

if (numberFromObject.constructor == Number) {

alert('numberFromObject는 숫자입니다.');

}

</script>

8.2.4 모든 객체에 메서드 추가

* Object 객체는 모든 자바 스크립트 객체의 최상위 객체이므로 Object 객체으 프로토타입에 속성 또는 메서드를 추가하면 모든 객체에서 활용할 수 있다.

<script>

Object.prototype.test = function () {

alert(this);

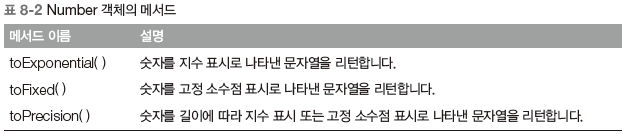
};

var number = 273;

number.test();

</script>

8.3 Number 객체



<script>

var number = 273.5210332;

var output = '';

output += number.toFixed(1) + '\n';

output += number.toFixed(4) + '\n';

output += Math.round(number \* 10) / 10;

output += number.toPrecision(5) + '\n';

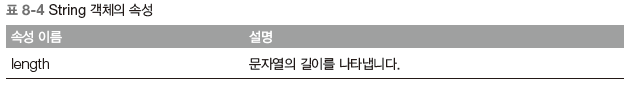
output += number.toExponential(5);

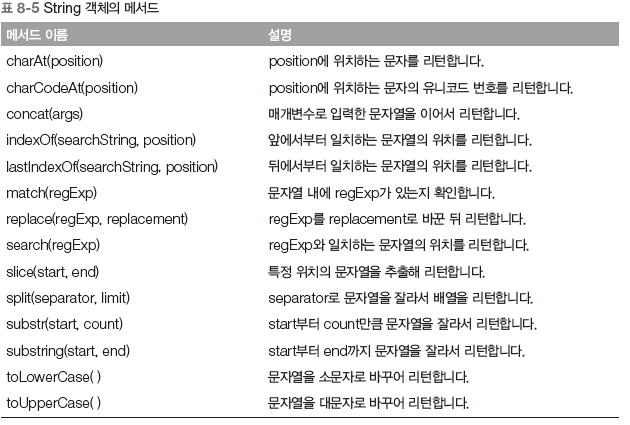
alert(output);

</script>

* 1.7976931348623157e+308까지 수를 나타낼 수 있고 그 이상은 무한한 수(POSITIVE\_INFINITY 혹은NEGATIVE\_INFINITY)로 인식한다.

8.4 String 객체





* String 객체의 메서드는 자기 자신을 변화시키지 않고 리턴하므로 문자열을 변화시키고 싶으면 아래와 같이 사용한다.

<!DOCTYPE html>

<html lang="en" xmlns="http://www.w3.org/1999/xhtml">

<head>

<meta charset="utf-8" />

<title></title>

<script>

var str = '@abcd@ef';

document.write('글씨길이 : ' + str.length + '<br>');

document.write('스트링 더하기 : ' + str.concat(str) + '<br>');

document.write('부분 글씨 : ' + str.substr(3, 2) + '<br>');

document.write('앞에서부터 일치하는 글자 위치 : '

+ str.indexOf('\*') + '<br>');

document.write('뒤에서부터 일치하는 글자위치'

+ str.lastIndexOf('@') + '<br>');

document.write('1번째부터 2번째 앞까지 :' + str.slice(1, 2)+'<br>');

document.write('str을 다 대문자로 : '+str.toUpperCase()+'<br>');

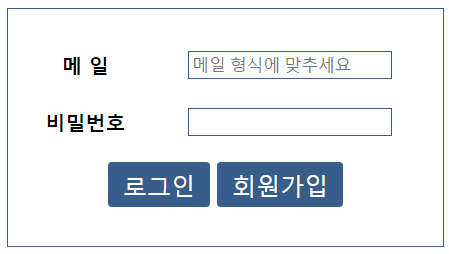
document.write('str='+str)

</script>

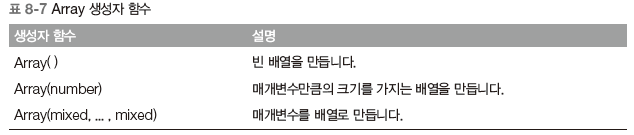
</head>

<body></body>

</html>



8.5 Array 객체

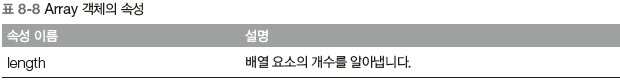


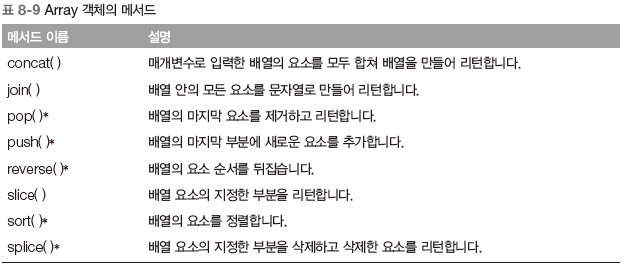
var array1 = [52, 273, 103, 57, 32];

var array2 = new Array();

var array3 = new Array(10);

var array4 = new Array(52, 273, 103, 57, 32);





<script>

var array = [52, 273, 103, 32];

array.sort(); //문자열 정렬

alert(array);

</script>

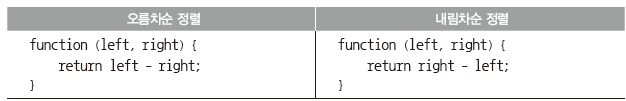
<script>

var array = ['a','z', 'p', 'd'];

array.sort();

alert(array);

</script>



<script>

var array = [52, 273, 103, 32];

array.sort(function (left, right) {

return left - right; //숫자 오름차순 정렬을 위해 추가 코드

});

alert(array);

</script>

- 학생 성적 정렬 코드 구현

<script>

function Student(name, kor, eng, mat, sci) {

this.name = name; this.kor = kor; this.eng = eng; this.mat = mat; this.sci = sci;

}

Student.prototype.getSum = function () {return this.kor + this.eng + this.mat + this.sci;};

Student.prototype.getAvg = function () {

return this.getSum() / 4;

};

Student.prototype.toString = function () {

return this.name + '\t' + this.getSum() + '\t' + this.getAvg()+'\n';

};

var students = [];

students.push(new Student('홍길동', 90, 99, 95, 96));

students.push(new Student('신길동', 87, 89, 93, 94));

students.push(new Student('김길동', 70, 88, 77, 66));

students.push(new Student('나선주', 90, 87, 78, 98));

students.push(new Student('윤아린', 95, 86, 97, 79));

// 총점기준으로 정렬하고 1등부터 3등까지만 배열에 남겨둡니다

students.sort(function (left, right) {

return right.getSum() - left.getSum();

});

students = students.slice(0, 3);

var output = '이름\t총점\t평균\n';

for (var idx in students) {

output += students[idx].toString();

}

alert(output);

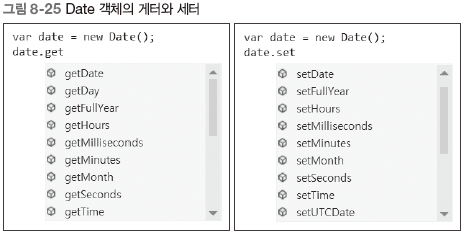
</script>

8.6 Date 객체

8.6.1 생성

- 특정한 날짜를 생성하기 위해서는, Date의 매개변수에 숫자를 연, 월-1, 일, 시, 분, 초, 밀리초 순서로 입력해 객체를 생성한다.

8.6.2 메서드



연도를 가져올 경우 getFullYear() 추천

<script>

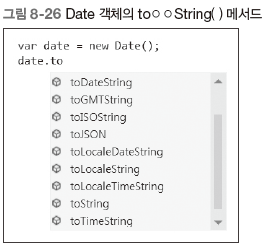
var date = new Date();

// 현재에서 7일을 더합니다.

date.setDate(date.getDate() + 7);

alert(date);

</script>



<script>

var date = new Date(1991, 11, 9);

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() + '\n';

alert(output);

</script>

8.6.3 날짜 간격 계산

- Date 객체는 D-Day를 구할 때도 활용할 수 있다. 날짜 간격을 구할 때는 getTime() 함수를 사용한다. getTime()함수는 1970년 1월1일 자정부터 지난 밀리초를 구한다. 이를 사용해 두 시간 사이의 초간격을 구하고 다시 날짜로 바꿀 수 있다.

<script>

var now = new Date();

var before = new Date(2018,2,5); // 3월5일

var interval = now.getTime() - before.getTime();

interval = Math.floor(interval / (1000 \* 60 \* 60 \* 24));

alert('Interval: ' + interval + '일');

</script>

- 아래와 같이 Date 객체의 프로토타입에 메서드를 넣으면 쉽고 편리하게 날짜 간격을 구할 수도 있다.

<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();

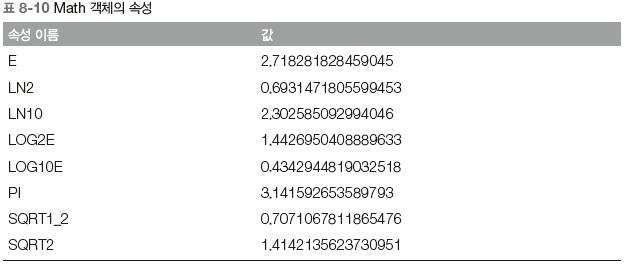
var before = new Date(2018,2,27);

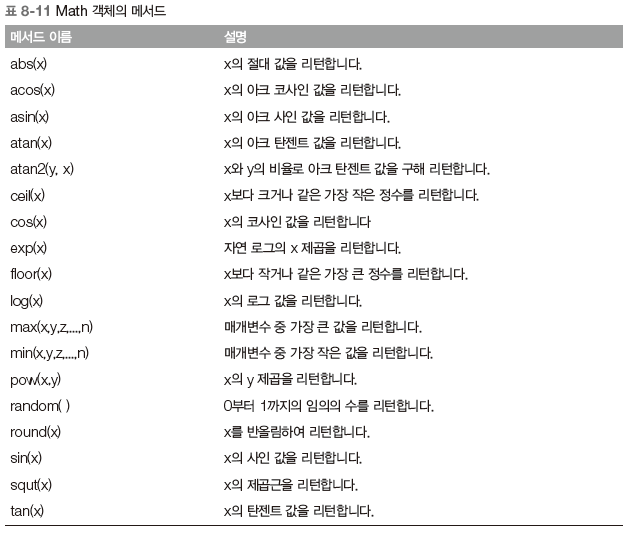
// 출력합니다.

alert('Interval: ' + now.getInterval(before) + '일');

</script>

8.7 Math 객체





<script>

var max = Math.max;

alert(max(52, 273, 103, 57, 32));

</script>