Chapter 05-1

객체-1 (내장 객체)-2

Math 내장 객체

Math 내장 객체

■ Math 객체

- 각종 산술 연산에 활용될 수 있는 속성과 메소드를 포함하고 있는 객체
  - 삼각함수, 지수함수, 로그함수, 각종 상수, 제곱근 등
- Math 객체는 정적 객체임
  - 인스턴스를 생성하지 않고 객체 자신이 직접 인스턴스로 사용되는 객체
  - new 연산자로 생성 불가능
    - [예] m = new Math(); (불가능)
  - 객체 자신이 직접 메소드와 속성을 호출해 사용
    - [예] Math.sin();
    - [예] Math.Pi;
  - 상수와 메서드를 제공

Math 내장 객체 (메서드)

■ Math 객체의 메서드와 상수

메서드와 상수	설명
abs(값)	인자로 가지는 값의 절대값을 반환
max(값 리스트)	인자로 가지는 값 리스트 중 가장 큰 값을 반환
min(값 리스트)	인자로 가지는 값 리스트 중 가장 작은 값을 반환
power(x, y)	인자로 가지는 x값의 y승 값을 반환
random()	0에서 1까지의 난수 값을 반환
round(값)	인자로 가지는 값을 소수점 첫째 자리에서 반올림하여 반환
ceil(값)	인자로 가지는 값보다 같거나 작은 수 중에서 가장 큰 정수 값을 반환
floor(값)	인자로 가지는 값보다 같거나 큰 수 중에서 작은 큰 정수 값을 반환
sqrt(값)	인자로 가지는 값의 제곱근을 반환
PI	원주율 상수를 반환

Math 내장 객체 (메서드)

5-math-1.htm (p.95)

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

	var  num = 2.1234;
	var  maxNum = Math.max(10, 5, 8, 30);
	var  minNum = Math.min(10, 5, 8, 30);

	document.write(maxNum, "<br>");
	document.write(minNum, "<br>");
	document.write(Math.round(num), "<br>");
	document.write(Math.floor(num), "<br>");
	document.write(Math.ceil(num), "<br>");
	document.write(Math.random(), "<br>");
	document.write(Math.PI, "<br>");

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

```
30
5
2
2
3
0.664225533262192
3.141592653589793
```

Math 내장 객체 (메서드)

■ Math.random() 메서드를 이용한 난수 발생

```
Math.random( ) * 10
```

- 0부터 10 사이의 실수로 난수를 발생

```
Math.floor( Math.random( ) * 11 )
```

- 정수만으로 난수를 반환하려면 floor( ) 메서드를 사용
  - floor() 메서드는 값을 인자보다 작은 정수를 반환하기 때문에 10이 아닌 11을 곱함

- 120~150 사이의 정수 값 난수를 발생하는 경우

```
Math.floor( Math.random( ) * 31 ) + 120
```

  - Math.floor(Math.random( ) * 31) -> 0부터 30까지의 난수를 발생

- 난수를 발생하는 공식

```
Math.floor( Math.random( ) * (최대값-최소값+1)) + 최소값
```

Math 내장 객체 (메서드)

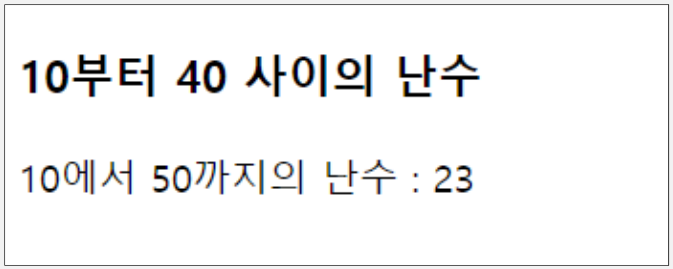
5-math-2.htm

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

		document.write("<h3>10부터 40 사이의 난수</h3>")

		var num =  Math.floor(Math.random()*(50-10+1)+10);
		document.write("10에서 50까지의 난수 : ", num ,"<br>");

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

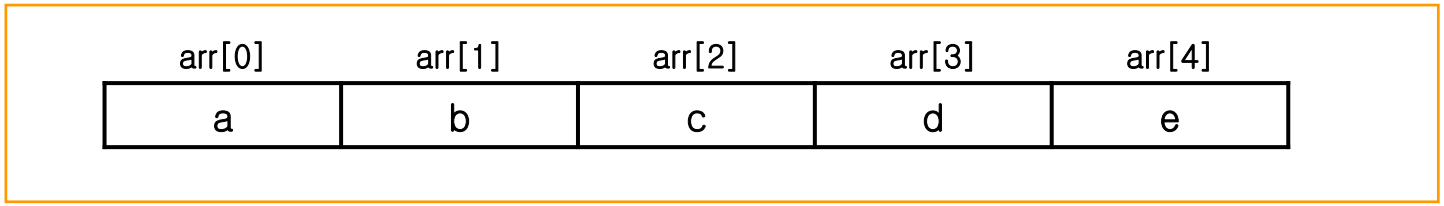


Array 내장 객체

Array 내장 객체

■ 배열(Array)

- 일반적으로 배열은 같은 데이터 타입, 같은 길이의 연속된 공간을 의미
  - 배열을 구성하는 각 기억 공간을 배열 요소라고 함
  - 배열 요소에 대한 접근은 배열의 이름과 0부터 시작하는 숫자로 구성된 첨자를 사용
- new 연산자를 사용해 인스턴스를 생성
- 배열의 예



  - 배열의 이름 : arr
  - 배열의 크기 : 5
  - 배열의 세 번째 요소 c의 참조 : arr[2]

Array 내장 객체

■ 배열의 생성과 초기화

• 인스턴스 생성 후 초기화

```
var 배열변수 = new Array (n);
배열변수[0] = 값1;
배열변수[1] = 값2;
    . . . . . .
배열변수[n-1] = 값n-1;
```

```
var arr = new Array();
arr[0] = 'a';
arr[1] = 'b';
arr[2] = 'c';
```

```
var arr new Array(3)
arr[0] = 'a';
arr[1] = 'b';
arr[2] = 'c';
```

- 배열 변수(arr) : 배열 요소를 참조할 때 사용하는 배열의 이름을 의미
- 배열 크기(n) : 생략 가능 (초기화되는 요소 수에 따라 크기가 결정됨)

• 인스턴스의 생성과 동시에 초기화

```
var 배열변수 = new Array(값1, 값2, ...., 값n-1);
```

```
var arr = new Array( 'a', 'b', 'c' );
```

• 대괄호를 사용해 생성과 동시에 초기화

```
var 배열변수 = [ 값1, 값2, ...., 값n-1 ];
```

```
var arr = [ 'a', 'b', 'c' ];
```

Array 내장 객체

■ 자바스크립트에서 배열의 특성

- 데이터 타입에 대한 제약이 없으므로 하나의 배열에 서로 다른 자료 형이 존재할 수 있음

```
var  arr = new Array( ‘a’, ‘b’, 100, ‘korea’, ‘d’ );
```

  - 하나의 배열 내에 문자 데이터와 숫자 데이터가 함께 저장될 수 있음

- 배열의 크기를 미리 선언하더라도 배열 요소의 수를 증가시키면 배열의 크기는 자동으로 증가됨

```
var  arr = new Array(3)
arr[0] = 10;
arr[1] = 20;
arr[2] = 30;
arr[3] = 40;
```

  - 크기가 3인 배열로 선언되었지만, 네 번째 요소가 입력됨에 따라 배열의 크기는 자동으로 4로 변경됨

Array 내장 객체

■ 다차원 배열

- **자바스크립트는 다차원 배열을 제공하지 않음**
  - 다차원 배열을 사용하기 위해서는 일차원 배열의 일차원 배열 개념을 사용
  - 일차원 배열을 선언하고, 각 요소에 다시 일차원 배열을 선언함

- **자바스크립트에서 2차원 배열 사용의 예**

```
Data = new Array(2);
Data[0] = new Array('kor', 'eng', 'mat');
Data[1] = new Array(80, 90, 100);
```

  - **물리적 구조**
    - 배열의 크기가 6인 1차원 배열
  - **논리적 구조**
    - 2 X 3인 테이블 구조의 배열

Array 내장 객체 (속성)

■ Array 객체의 속성

속성	설명
length	배열의 길이를 반환

- 자바스크립트 Array 객체가 가지는 유일한 속성
- 일반적으로 배열 요소를 모두 출력할 때 순환문의 조건으로 사용

- 배열의 모든 요소 출력
  - for 순환문을 사용한 출력
    - length 속성을 사용해 배열의 크기를 추출한 다음 배열의 모든 요소를 출력
  - for - in 순환문을 사용한 출력

Array 내장 객체 (속성)

- 배열의 요소 출력의 예

5-array-length.htm

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

		var array = ['포도', '사과', '바나나', '망고'];

		// for 순환문을 사용한 출력
		document.write("<h3>for 순환문을 이용한 출력</h3>");
		for( var i=0; i < array.length; i++ ) {
			document.write("array["+i+"] : ", array[i], "<br>");
		}

		// for-in 순환문을 사용한 출력
		document.write("<h3>for-in 순환문을 이용한 출력</h3>");
		for (var i in array) {
			document.write("array["+i+"] : ", array[i], "<br>");
		}

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

for 순환문을 이용한 출력

array[0] : 포도
array[1] : 사과
array[2] : 바나나
array[3] : 망고

for-in 순환문을 이용한 출력

array[0] : 포도
array[1] : 사과
array[2] : 바나나
array[3] : 망고

Array 내장 객체 (메서드)

■ Array 내장 객체의 메서드

메서드	설명
join(연결문자)	배열의 요소들을 연결문자를 사용해 하나의 문자열로 결합하여 반환
reverse()	배열의 각 요소를 역순으로 출력
sort()	배열의 각 요소를 오름차순으로 정렬하여 반환
slice()	배열의 일부 요소를 추출하여 새로운 배열로 반환
splice()	배열의 특정 위치의 요소들을 제거하고 새로운 배열 요소를 지정
concat()	2개의 배열을 결합하여 하나의 배열로 반환
pop()	배열의 마지막 요소를 제거하고 제거된 문자열을 반환
push()	배열의 마지막 부분에 하나 이상의 새로운 배열 요소를 삽입
shift()	배열의 첫 번째 요소를 제거하고 제거된 문자열을 반환
unshift()	배열의 앞 부분에 하나 이상의 새로운 배열 요소 추가

Array 내장 객체 (메서드)

■ join() 메서드

```
var 변수  =  배열.join(결합자)
```

- **배열의 요소를 인자로 지정한 [결합자]로 결합하여 새로운 문자열 생성**
  - 반환 값은 배열이 아니라 문자열 임에 유의

Array 내장 객체 (메서드)

■ reverse() 메서드

```
배열.reverse()
```

- 배열 요소의 순서를 역순으로 반환
  - 정렬이 아님
- 원래의 배열을 역순으로 저장
  - 원래의 배열 내용이 변화됨

```
배열1.reverse()
```

- 배열1의 내용을 역순으로 바꾼 후 다시 배열1에 저장

```
배열2 = 배열1.reverse()
```

- 배열1의 내용을 역순으로 바꾼 후 배열1과 배열2에 저장
- 배열1의 내용 = 배열2의 내용

Array 내장 객체 (메서드)

5-array-join-reverse.htm

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

		var array = ['포도', '사과', '바나나', '망고'];

		var str = array.join('-');
		document.write(str, "<br><br>");

		array.reverse();

		for (var i in array) {
			document.write("array["+i+"] : ", array[i], "<br>");
		}

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

```
포도-사과-바나나-망고

array[0] : 망고
array[1] : 바나나
array[2] : 사과
array[3] : 포도
```

Array 내장 객체 (메서드)

■ sort() 메서드

- 배열의 각 요소를 오름차순으로 정렬하여 반환
  - 숫자 정렬이 아닌 문자 정렬을 수행
- 원래의 배열을 정렬하여 저장
  - 원래의 배열 내용이 변화됨

```
배열1.sort()
```

- 배열1의 내용을 오름차순 정렬한 후 다시 배열1에 저장

```
배열2 = 배열1.sort()
```

- 배열1의 내용을 오름차순으로 정렬하여 배열1과 배열2에 저장(배열1도 정렬됨)

  - 내림차순 정렬을 수행하는 메서드는 존재하지 않음
    - sort()와 reverse() 함께 사용

Array 내장 객체 (메서드)

- 숫자 정렬의 경우

오름차순 정렬	내림차순 정렬

array.sort(function (left, right)) {
  return left - right;
});
	array.sort(function (left, right)) {
	  return right - left;
	});

  - 내부적으로 quick sort나 merge sort 같은 정렬이 수행됨

Array 내장 객체 (메서드)

5-array-sort.htm

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

		var array1 = ['포도', '사과', '바나나', '망고'];
		var array2 = [12, 45, 23, 7];

		array1.sort();
		for (var i in array1) {
			document.write("array1["+i+"] : ", array1[i], "<br>");
		}

		document.write("<br>");

		array2.sort(function(left,right) {
			return left-right;
		});
		for (var i in array2) {
			document.write("array2["+i+"] : ", array2[i], "<br>");
		}

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

```
array1[0] : 망고
array1[1] : 바나나
array1[2] : 사과
array1[3] : 포도

array2[0] : 7
array2[1] : 12
array2[2] : 23
array2[3] : 45
```

Array 내장 객체 (메서드)

■ slice() 메서드

```
배열변수 = 배열.slice( m, n )
```

- 배열의 m 위치부터 n위치까지의 요소를 추출하여 새로운 배열 생성
  - 메소드의 반환값은 배열임
- 배열1의 m위치부터 n위치까지 추출하여 배열2를 생성
  - m 위치 : 0부터 시작
  - n 위치 : 1부터 시작

Array 내장 객체 (메서드)

5-array-slice.htm

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

		var array1 = new Array('when', 'a', 'child', 'is', 'born');
		for (var i in array1) {
			document.write("array1["+i+"] : ", array1[i], "<br>");
		}

		document.write("<hr>");

		var array2 = array1.slice(2,4);

		for( var i in array2 ) {
			document.write("arr2["+i+"] : " + array2[i] + "<br>");
		}

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

```
array1[0] : when
array1[1] : a
array1[2] : child
array1[3] : is
array1[4] : born

arr2[0] : child
arr2[1] : is
```

Array 내장 객체 (메서드)

■ splice() 메서드

```
문자열 변수 = 배열.splice( s, m, data )
```

- 배열의 s번째 위치에서부터 m개의 요소를 제거한 다음 요소 data를 삽입
  - 제거된 요소를 반환

- 사용 예

```
문자열변수 = 배열.splice( s, m, data )
```

- 배열에서 옵셋이 s인 위치(0부터 시작)로부터 m개의 요소를 제거한 후 해당 위치에 data 요소를 삽입
  - 기존 배열에는 새로운 요소가 추가되고, 문자열 변수에는 제거된 요소들이 콤마()로 구분되어 저장

```
문자열변수 = 배열.splice( s, 0, data )
```

- 배열에서 옵셋이 s인 위치(0부터 시작)에 data 요소를 삽입
  - 기존 요소를 제거하지 않음
  - 기존 배열에는 새로운 요소가 추가되고, 문자열 변수에는 null이 저장됨

Array 내장 객체 (메서드)

5-array-splice.htm

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

		var array1 = new Array('서울', '부산', '대구', '광주');
		for (var i in array1) {
			document.write("array1["+i+"] : ", array1[i], "<br>");
		}

		document.write("<h3>배열1의 부산과 대구를 제거, 청주 대전 울산을 추가</h3>");

		str = array1.splice(1, 2, '청주', '대전','울산');

		for (var i in array1) {
			document.write("array1["+i+"] : ", array1[i], "<br>");
		}

		document.write("<BR>제거된 배열 요소 : " + str);

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

array1[0] : 서울
array1[1] : 부산
array1[2] : 대구
array1[3] : 광주

배열1의 부산과 대구를, 청주 대전 울산을 추가

array1[0] : 서울
array1[1] : 청주
array1[2] : 대전
array1[3] : 울산
array1[4] : 광주

제거된 배열 요소 : 부산,대구

Array 내장 객체 (메서드)

■ concat() 메서드

```
배열3 = 배열1.concat( 배열2 )
```

- 배열1의 뒷부분에 배열2를 추가하여 새로운 배열3을 생성

Array 내장 객체 (메서드)

5-array-concat.htm

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

		var array1 = new Array('a', 'b', 'c');
		var array2 = new Array(10, 20, 30);

		var array3 = array1.concat(array2)

		for (var i in array3) {
			document.write("array3["+i+"] : ", array3[i], "<br>");
		}

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

```
array3[0] : a
array3[1] : b
array3[2] : c
array3[3] : 10
array3[4] : 20
array3[5] : 30
```

Array 내장 객체 (메서드)

■ pop() 메서드와 push() 메서드

- pop() 메서드

```
문자열변수 = 배열.pop( )
```

  - 배열의 마지막 요소를 제거하고 제거된 문자열을 반환

- push() 메서드

```
배열.push( 배열요소 리스트 )
```

  - 배열의 마지막에 인자로 가지는 배열 요소들을 추가
    - 메소드의 반환 값은 없음

Array 내장 객체 (메서드)

5-array-pop-push.htm

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

		var array1 = new Array('서울', '부산', '대구', '광주');
		document.write(array1,"<br>")

		document.write("<h3>마지막에 청주 대전 추가</h3>");
		array1.push('청주', '대전');
		document.write(array1,"<br>");

		document.write("<h3>마지막 요소 제거</h3>");
		var str1 = array1.pop();
		document.write(array1,"<br>");
		document.write("제거된 요소 : ", str1,"<br>");

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

서울,부산,대구,광주

마지막에 청주 대전 추가

서울,부산,대구,광주,청주,대전

마지막 요소 제거

서울,부산,대구,광주,청주
제거된 요소 : 대전

Array 내장 객체 (메서드)

■ shift() 메서드와 unshift() 메서드

- shift() 메서드

```
문자열변수 = 배열.pop( )
```

  - 배열의 첫 번째 요소를 제거하고 제거된 문자열을 반환

- unshift() 메서드

```
배열.push( 배열요소 리스트 )
```

  - 배열의 앞부분에 인자로 가지는 배열 요소들을 추가
    - 메소드의 반환값은 없음

Array 내장 객체 (메서드)

5-array-shift-unshift.htm

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

		var array1 = new Array('서울', '부산', '대구', '광주');
		document.write(array1,"<br>")

		document.write("<h3>배열의 앞에 청주 대전 추가</h3>");
		array1.unshift('청주', '대전');
		document.write(array1,"<br>");

		document.write("<h3>마지막 첫 번째 요소 제거</h3>");
		var str1 = array1.shift();
		document.write(array1,"<br>");
		document.write("제거된 요소 : ", str1,"<br>");

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

서울,부산,대구,광주

배열의 앞에 청주 대전 추가

청주,대전,서울,부산,대구,광주

마지막 첫 번째 요소 제거

대전,서울,부산,대구,광주
제거된 요소 : 청주

Array 내장 객체 (메서드)

■ ECMA6에 추가된 Array 내장 객체 관련 메서드

메서드	설명
isArray()	인자로 가지는 값이 배열인지 조사하는 메서드 (배열일 경우 true를 반환)
indexOf()	배열의 처음 요소부터 조사해 인자로 지정된 요소가 존재하면 해당 index를 반환
lastIndexOf()	배열의 마지막 요소부터 조사해 인자로 지정된 요소가 존재하면 해당 index를 반환
forEach()	반복문을 사용해 배열의 각 요소에 대해 특정한 함수를 모두 적용
filter()	기존 배열 요소 중에서 특정 조건을 만족하는 요소만을 추출해 새로운 배열을 생성
every()	기존 배열의 모든 요소가 특정 조건을 만족하는지 조사
some()	기존 배열의 모든 요소 중 적어도 하나 이상이 특정 조건을 만족하는지 조사

Array내장 객체 (메서드)

■ isArray() 메서드

```
var 변수 = 배열.isArray( [인자] )
```

- 인자로 가지는 값이 배열인지 조사하는 메서드 (배열일 경우 true를 반환)

5-Array-isarray.htm

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

		var myArray = [1, 2, 3];

		if(Array.isArray(myArray)) {
			document.write(myArray);
		}

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

1,2,3

Array 내장 객체 (메서드)

■ indexOf() 메서드

```
var 변수 = 배열.indexOf( 인자 )
```

- 배열의 처음(왼쪽) 요소부터 조사하여 인자로 지정된 요소가 존재하면 해당 index를 반환
  - index의 기준은 맨 좌측 요소를 사용(index=0)
    - 존재하지 않으면 -1을 반환

■ lastIndexOf() 메서드

```
var 변수 = 배열.lastIndexOf( 인자 )
```

- 배열의 마지막(오른쪽) 요소부터 조사하여 인자로 지정된 요소가 존재하면 해당 index를 반환
  - index의 기준은 맨 좌측 요소를 사용(index=0)
    - 존재하지 않으면 -1을 반환

Array 내장 객체 (메서드)

5-Array-indexof-lastindexof.htm

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

		var array = [1, 2, 3, 4, 5, 5, 4, 3, 2, 1];

		var output1 = array.indexOf(4);
		var output2 = array.lastIndexOf(4);

		var output3 = array.indexOf(8);

		document.write("output1 : ", output1, "<br>");
		document.write("output2 : ", output2, "<br>");
		document.write("output3 : ", output3, "<br>");

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

```
output1 : 3
output2 : 6
output3 : -1
```

Array 내장 객체 (메서드)

■ forEach() 메서드

- 반복문을 사용해 배열의 각 요소에 대해 특정한 함수를 모두 적용

```
forEach ( function( element, index ) {

        적용할 코드

});
```

- element
  - 현재 반복문에서 수행되는 배열 요소의 값
- index
  - 현재 반복문에서 수행되는 배열 요소의 index

Array 내장 객체 (메서드)

5-Array-forEach.htm

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

		var array = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10];

		var sum = 0;
		var output = '';

		array.forEach(function (element, index) {
			sum = element * 10;
			output += index + ": " + sum + "<br>";
		});

		document.write(output);

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

```
0: 10
1: 20
2: 30
3: 40
4: 50
5: 60
6: 70
7: 80
8: 90
9: 100
```

Array 내장 객체 (메서드)

■ filter() 메서드

- 기존 배열 요소 중에서 특정 조건을 만족하는 요소만을 추출해 새로운 배열을 생성

```
var  변수  = 배열.filter( function() { } );
```

5-Array-filter.htm

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

		var myArray = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10];

		function my_filter(element, index, myArray) {
			return element <= 5;
		}
		newArray = myArray.filter(my_filter);
		document.write(newArray);

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

1,2,3,4,5

Array 내장 객체 (메서드)

■ every() 메서드

- 배열의 모든 요소가 특정 조건을 만족하는지 조사 (모두 만족할 경우 true를 반환)

```
var  변수  = 배열.every( function() { } );
```

5-Array-every.htm

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

		var myArray = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10];
		function lessThenFive(element, index, myArray) {
			return element < 5;
		}

		var result = myArray.every(lessThenFive);
		document.write(result);

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

false

Array 내장 객체 (메서드)

■ some() 메서드

- 배열의 모든 요소 중 하나 이상이 특정 조건을 만족하는지 조사 (하나 이상 만족할 경우 true를 반환)

```
var  변수  = 배열.some( function() { } );
```

5-Array-some.htm

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

		var myArray = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10];
		function lessThenFive(element, index, myArray) {
			return element < 5;
		}

		var result = myArray.some(lessThenFive);
		document.write(result);

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

true

수고하셨습니다