

JS : 함수와 배열

JS 함수와 배열

- ◆ 함수
- ◆ 배열
- ◆ **배열 메소드**
- ◆ 2차원 배열

join 메소드

예제 10-24 join 메소드 활용하기

```
<script>
```

```
var city=["서울", "부산", "대전"];
```

```
var joindata1=city.join();
```

```
var joindata2=city.join('-');
```

```
var joindata3=city.join(' 그리고 ');
```

배열에 저장된 모든 원소를 문자열로
변환한 후 연결하여 출력

```
document.write("조인 결과1 : " + joindata1 + "<p/>");
```

```
document.write("조인 결과2 : " + joindata2 + "<p/>");
```

```
document.write("조인 결과3 : " + joindata3 + "<p/>");
```

```
</script>
```

concat 메소드

예제 10-25 concat 메소드 활용하기

```
<script>
  var city01=[ "서울", "부산", "대전" ];
  var city02=[ "대구", "광주", "인천" ];
  var city03=[ "전주", "부여", "세종" ];

  var data1=city01.concat( "수원", "오산" );
  var data2=city01.concat(city02);
  var data3=city01.concat(city03, city02);

  document.write( "결과1 : " + data1 + "<p/>" );
  document.write( "결과2 : " + data2 + "<p/>" );
  document.write( "결과3 : " + data3 + "<p/>" );
</script>
```

reverse 메소드

예제 10-26 reverse 메소드 활용하기

```
<script>
  var data=[9, 8, 7, 6, 5, 4, 3, 2, 1];
  document.write("배열 : " + data.join() + "<p/>");

  var rdata=data.reverse();

  document.write("결과 : " + rdata + "<p/>");
</script>
```

sort 메소드

예제 10-27 sort 메소드 활용하기

```
<script>
  var ndata1=[19, 38, 67, 26, 55, 24, 53, 12, 31];
  var ndata2=[132, 2, 41, 123, 45, 1234, 6, 29, 4567];
  var edata=[ 'Apple', 'Html', 'Game', 'Computer', 'Java' ];
  var kdata=[ '서울', '부산', '구포', '대구', '인천' ];

  document.write("수치 정렬1 : " + ndata1.sort() + "<p/>");
  document.write("수치 정렬2 : " + ndata2.sort() + "<p/>");

  document.write("수치 정렬3 : " + ndata2.sort( function(a, b) {return a - b;} ) + "<p/>");

  document.write("영문 정렬 : " + edata.sort() + "<p/>");
  document.write("한글 정렬 : " + kdata.sort() + "<p/>");
</script>
```

수치 정렬1 : 12,19,24,26,31,38,53,55,67

수치 정렬2 : 123,1234,132,2,29,41,45,4567,6

수치 정렬3 : 2,6,29,41,45,123,132,1234,4567

영문 정렬 : Apple,Computer,Game,Html,Java

한글 정렬 : 구포,대구,부산,서울,인천

slice 메소드

예제 10-28 slice 메소드 활용하기

```
<script>
  var kdata=[ '서울', '부산', '구포', '대구', '인천', '대전', '세종' ];
  var str1=kdata.slice(0, 4);
  var str2=kdata.slice(2, -1);
  var str3=kdata.slice(-4, -2);
  document.write("부분 배열1 : " + str1 + "<p/>");
  document.write("부분 배열2 : " + str2 + "<p/>");
  document.write("부분 배열3 : " + str3 + "<p/>");
</script>
```

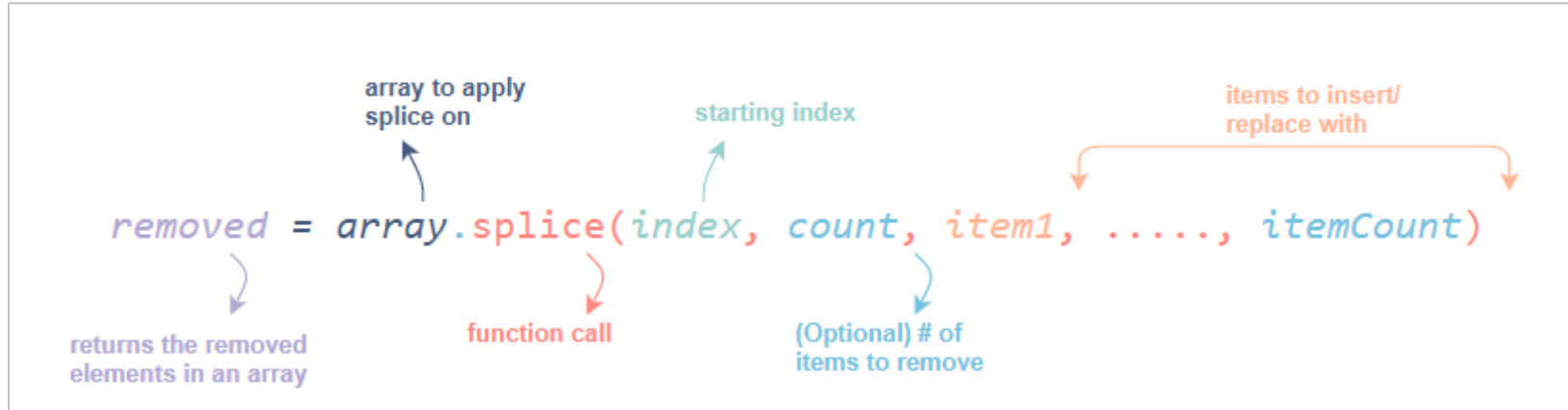
배열의 특정 범위에 속하는 원소만
선택하여 배열 생성

부분 배열1 : 서울,부산,구포,대구

부분 배열2 : 구포,대구,인천,대전

부분 배열3 : 대구,인천

splice 메소드



예제 10-29 splice 메소드 활용 – 배열에 데이터 추가/삭제, 제거된 원소 반환

```
<script>
  var kdata=[ '서울', '부산', '구포', '대구', '대전' ];
  var str1=kdata.splice(1, 2);

  document.write("삭제 데이터 : " + str1 + "<br>");
  document.write("남은 배열 : " + kdata + "<p/>");

  var str2=kdata.splice(1, 1, '강릉', '세종');
  document.write("삭제 데이터 : " + str2 + "<br>");
  document.write("남은 배열 : " + kdata + "<p/>");

  var str3=kdata.splice(2, Number.MAX_VALUE);
  document.write("삭제 데이터 : " + str3 + "<br>");
  document.write("남은 배열 : " + kdata + "<p/>");
</script>
```

삭제 데이터 : 부산,구포
남은 배열 : 서울,대구,대전

삭제 데이터 : 대구
남은 배열 : 서울,강릉,세종,대전

삭제 데이터 : 세종,대전
남은 배열 : 서울,강릉

https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array/splice

pop & push 메소드

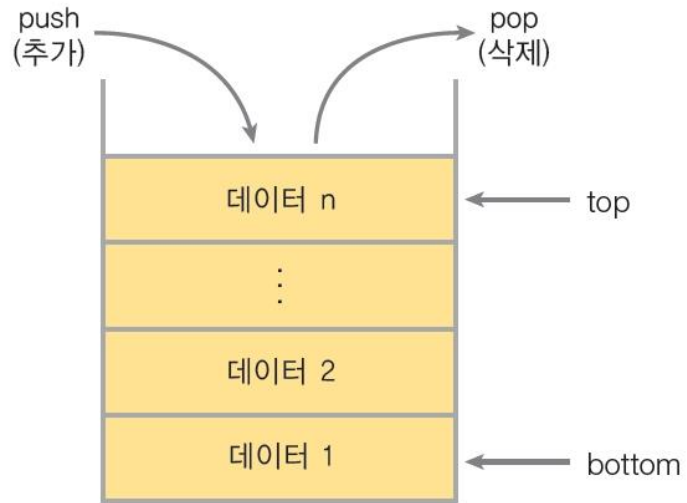


그림 10-4 스택의 구조

예제 10-30 pop & push 메소드 활용하기

```
<script>
  var kdata = ['서울', '부산', '구포', '대구', '대전'];
  var p1 = kdata.push('청주', '세종');

  document.write("데이터 : " + p1 + "<br>");
  document.write("배열 데이터 : " + kdata + "<p/>");

  var p2 = kdata.pop();

  document.write("데이터 : " + p2 + "<br>");
  document.write("배열 데이터 : " + kdata + "<p/>");
</script>
```

데이터 : 7
배열 데이터 : 서울,부산,구포,대구,대전,청주,세종

데이터 : 세종
배열 데이터 : 서울,부산,구포,대구,대전,청주

shift & unshift 메소드

예제 10-31 shift & unshift 메소드 활용하기

```
<script>
  var kdata = ['서울', '부산'];
  var p1 = kdata.unshift('청주', '세종');  배열의 첫 번째에 데이터를 삽입

  document.write("데이터 : " + p1 + "<br>");
  document.write("배열 데이터 : " + kdata + "<p/>");

  var p2 = kdata.shift();  배열의 첫 번째 원소를 삭제
                           해당 원소 반환

  document.write("데이터 : " + p2 + "<br>");
  document.write("배열 데이터 : " + kdata + "<p/>");
</script>
```

```
데이터 : 4
배열 데이터 : 청주,세종,서울,부산

데이터 : 청주
배열 데이터 : 세종,서울,부산
```

The `unshift()` method adds one or more elements to the beginning of an array and returns the new length of the array.


The `shift()` method removes the **first** element from an array and returns that removed element. This method changes the length of the array.

forEach 메소드

The `forEach()` method executes a provided function once for each array element.

```
var txt = "";
var numbers = [45, 4, 9, 16, 25];
numbers.forEach(myFunction);

function myFunction(value, index, array) {
    txt = txt + value + "<br>";
}
```



```
// Arrow function
forEach((element) => { /* ... */ })
forEach((element, index) => { /* ... */ })
forEach((element, index, array) => { /* ... */ })

// Callback function
forEach(callbackFn)
forEach(callbackFn, thisArg)

// Inline callback function
forEach(function(element) { /* ... */ })
forEach(function(element, index) { /* ... */ })
forEach(function(element, index, array){ /* ... */ })
forEach(function(element, index, array) { /* ... */ }, thisArg)
```

예제 10-32 forEach 메소드 활용하기 1

```
<script>
  var kdata=[ '서울', '부산', '청주', '대구' ];

  function printArr(item, index) {
    document.write("배열 데이터 [" + index + "] : " + item + "<br>");
  }

  kdata.forEach(printArr);
</script>
```


예제 10-33 forEach 메소드 활용하기 2

```
<script>
  var data=[1, 2, 3, 4, 5, 6, 7, 8, 9, 10];
  var sum=0;
  function addArr(value) {
    sum+=value;
  }
  data.forEach(addArr);
  document.write("배열 데이터 합 : " + sum + "<p/>");
</script>
```

map 메소드

The `map()` method **creates a new array** populated with the results of calling a provided function on every element in the calling array.

```
1 const array1 = [1, 4, 9, 16];
2
3 // pass a function to map
4 const map1 = array1.map(x => x * 2);
5
6 console.log(map1);
7 // expected output: Array [2, 8, 18, 32]
```



```
// Arrow function
map((element) => { /* ... */ })
map((element, index) => { /* ... */ })
map((element, index, array) => { /* ... */ })

// Callback function
map(callbackFn)
map(callbackFn, thisArg)

// Inline callback function
map(function(element) { /* ... */ })
map(function(element, index) { /* ... */ })
map(function(element, index, array){ /* ... */
})
map(function(element, index, array) { /* ... */
}, thisArg)
```


예제 10-34 map 메소드 활용하기

```
<script>
  var data=[1, 2, 3, 4, 5, 6, 7, 8, 9, 10];
  function mapArr(value) {
    return value*value;
  }

  var mapdata = data.map(mapArr);

  document.write("원래 배열 :" + data + "<p/>");
  document.write("map 메소드 적용 배열 :" + mapdata + "<p/>");
</script>
```

filter 메소드

예제 10-35 filter 메소드 활용하기

```
<script>
  var data=[21, 42, 33, 14, 25, 12, 37, 28, 16, 11];
  function filterArr(value) {
    return value>=18;
  }

  var fdata = data.filter(filterArr);

  document.write("필터 전 배열 : " + data + "<p/>");
  document.write("필터 후 배열 : " + fdata + "<p/>");
</script>
```

The `filter()` method **creates a new array** with all elements that pass the test implemented by the provided function.

indexOf & lastIndexOf 메소드

예제 10-36 indexOf & lastIndexOf 메소드 활용하기

```
<script>
  var data=[10, 20, 30, 40, 30, 60, 70, 30, 90,100];

  document.write("배열 데이터 : [" + data + "]<p/>");

  document.write("처음부터 검색한 30의 인덱스 : " + data.indexOf(30) + "<p/>");

  document.write("마지막에서 검색한 30의 인덱스 : " + data.lastIndexOf(30) + "<p/>");

  document.write("세 번째부터 검색한 30의 인덱스 : " + data.indexOf(30, 3) + "<p/>");

  document.write("처음부터 검색한 300의 인덱스 : " + data.indexOf(300) + "<p/>");
</script>
```

JS 함수와 배열

- ◆ 함수
- ◆ 배열
- ◆ 배열 메소드
- ◆ **2차원 배열**

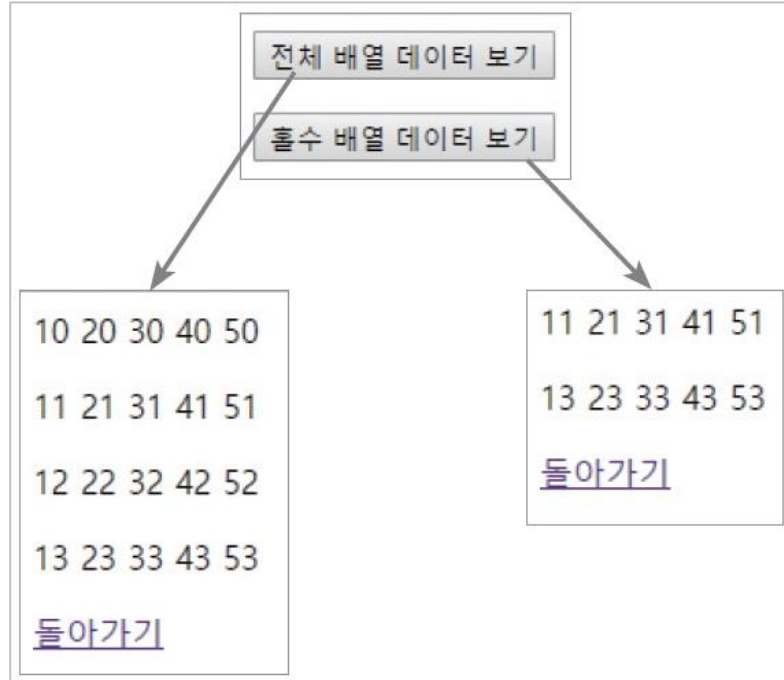
2차원 배열

예제 10-38 2차원 배열 생성하고 조회하기

```
<script>
var d2data=[[10, 20, 30, 40, 0], [60, 70, 80, 90, 0]];
d2data[0][4]=50;
d2data[1][4]=100;

document.write("2차원 배열 첫 번째 데이터 : " + d2data[0][0] + "<br>");
document.write("2차원 배열 마지막 데이터 : " + d2data[1][4] + "<br>");

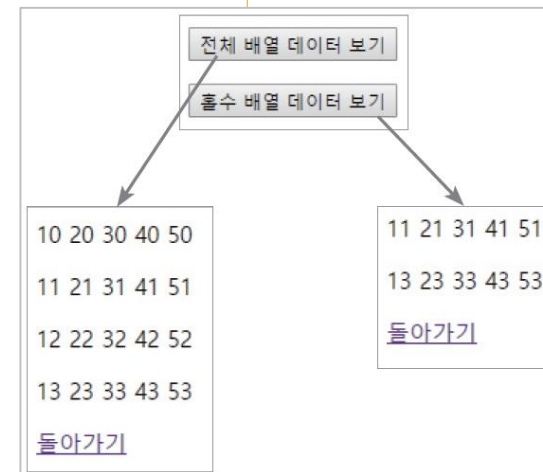
document.write("2차원 배열 행 길이 : " + d2data.length + "<br>");
document.write("2차원 배열 열 길이 : " + d2data[0].length + "<br>");
</script>
```

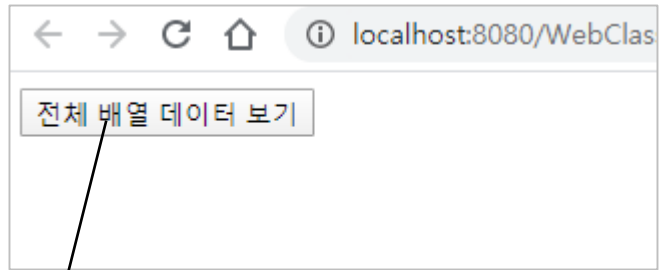


예제 10-39 1차원 배열로 2차원 배열 생성하고 조회하기

```
<script>
var arr0=[10, 20, 30, 40, 50];
var arr1=[11, 21, 31, 41, 51];
var arr2=[12, 22, 32, 42, 52];
var arr3=[13, 23, 33, 43, 53];
var allArr=[arr0, arr1, arr2, arr3];    // 2차원 배열 생성
var partArr=[arr1, arr3];              // 2차원 배열 생성
function printAll() {
    for(var x=0; x<allArr.length; x++) {
        for(var y=0; y<allArr[x].length; y++) {
            document.write(allArr[x][y] + " ");
        }
        document.write("<p/>");
    }
    document.write("<a href='39_arr.html'>돌아가기</a>");
}
function printPart() {
    for(var x=0; x<partArr.length; x++) {
        for(var y=0; y<partArr[x].length; y++) {
            document.write(partArr[x][y] + " ");
        }
        document.write("<p/>");
    }
    document.write("<a href='39_arr.html'>돌아가기</a>");
}
</script>
```

```
<button type="button" onclick="printAll()">전체 배열 데이터 보기</button></p>
<button type="button" onclick="printPart()">홀수 배열 데이터 보기</button>
```



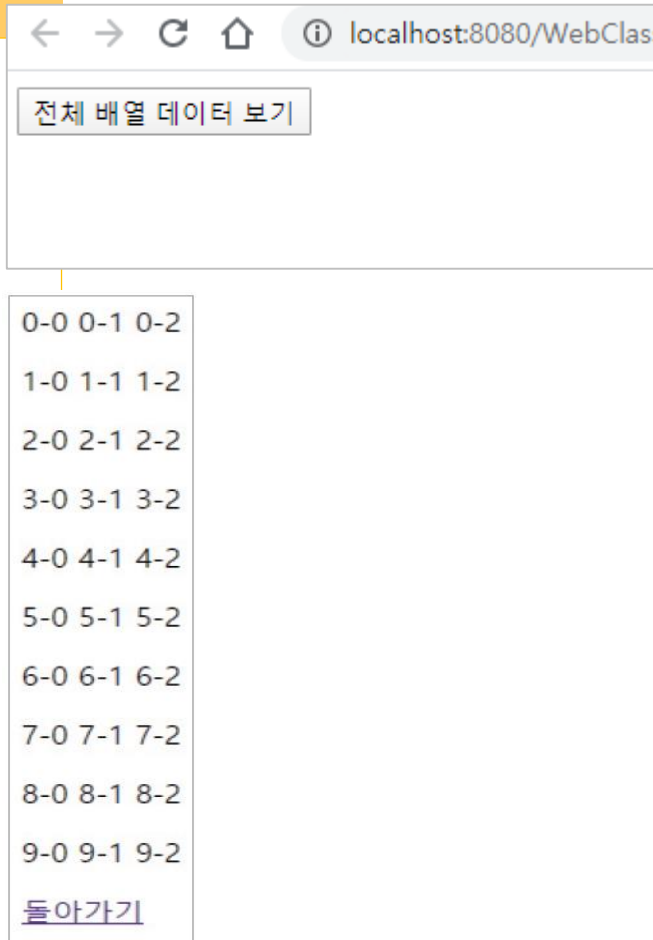


0-0	0-1	0-2
1-0	1-1	1-2
2-0	2-1	2-2
3-0	3-1	3-2
4-0	4-1	4-2
5-0	5-1	5-2
6-0	6-1	6-2
7-0	7-1	7-2
8-0	8-1	8-2
9-0	9-1	9-2
돌아가기		

예제 10-40 반복문을 이용하여 2차원 배열 만들기

```
<script>
  var data=[];
  for(var i=0; i<10; ++i) {
    data[i]=[String(i+"-"+0), String(i+"-"+1), String(i+"-"+2)];
  }

  function printData() {
    for(var x=0; x<data.length; x++) {
      for(var y=0; y<data[x].length; y++) {
        document.write(data[x][y] + " ");
      }
      document.write("<p/>");
    }
    document.write("<a href='40_arr.html'>돌아가기</a>");
  }
</script>
<button type="button" onclick="printData()">전체 배열 데이터 보기</button>
```



JS 함수와 배열

◆ 자바스크립트 함수

- 함수선언, 호출, 반환값, 인자와 매개변수

◆ 자바스크립트 배열

- 배열 생성, 데이터 접근/조작

◆ 배열 관련 메소드

- join, concat, reverse, sort, slice, splice
- pop/push, shift/unshift
- forEach, map, filter
- indexOf/lastIndexOf

◆ 2차원 배열