

1. 함수 선언과 호출

● 함수 선언과 호출 형식

```
function 함수명(매개 변수1, 매개 변수2, ...) { // 함수 선언
  실행 문장;
  return 반환값;
}
함수명(인자1, 인자2, ...); // 함수 호출
```

- 함수명 : 함수 이름
- 인자 : 함수를 호출할 때 전달하는 입력값
- 매개 변수 : 함수 호출문에서 전달한 인자를 받기 위해 선언된 변수
- function : 함수를 선언할 때 사용하는 키워드
- return : 함수에서 수행한 결과값을 반환할 때 사용하는 키워드

1. 함수 선언과 호출

● 함수 선언 – 일반적인 방법(기본 함수)

```
function 함수명(매개 변수1, 매개 변수2, ...) { // 함수 선언
    실행 문장;
}
함수명(인자1, 인자2, ...); // 함수 호출
```

예제 10-1 기본 함수 호출하기

ch10/01_func.html

```
<script>
var text1="함수 선언 전 호출";
var text2="함수 선언 후 호출";
printMsg(text1); // 함수 선언 전 호출
function printMsg(msg) { // 함수 선언
    document.write("함수 호출 메시지 : " + msg + "<br>");
}
printMsg(text2); // 함수 선언 후 호출
</script>
```

```
함수 호출 메시지 : 함수 선언 전 호출
함수 호출 메시지 : 함수 선언 후 호출
```

1. 함수 선언과 호출

예제 10-2 onclick 속성값으로 함수 호출하기

ch10/02_func.html

```
<script>
  function printMsg(name, age) {    // 함수 선언
    document.write("학생 이름 : <b>" + name + "</b><br>");
    document.write("학생 나이 : <b>" + age + "</b><br>");
  }
</script>
<button type="button" onclick="printMsg('홍길동', 21)">학생 정보</button>
```

학생 정보

학생 이름 : 홍길동
학생 나이 : 21

1. 함수 선언과 호출

● 함수 선언 – 함수 표현식으로 작성하는 방법(무명 함수)

```
var 변수명=function(매개 변수1, 매개 변수2, ...) { // 함수 선언
    실행 문장;
}
변수명(인자1, 인자2, ...); // 함수 호출
```

예제 10-3 무명 함수 호출하기

ch10/03_func.html

```
<script>
var text1="함수 선언 전 호출 에러";
var text2="함수 선언 후 호출만 가능";
// printMsg(text1); // 함수 선언 전 호출 에러
var printMsg=function(msg) { // 함수 객체 선언
    document.write("함수 호출 메시지 : " + msg + "<br>");
}
printMsg(text2); // 함수 선언 후 호출 가능
</script>
```

함수 호출 메시지 : 함수 선언 후 호출만 가능

1. 함수 선언과 호출

예제 10-4 기본 함수와 무명 함수 호출 우선순위 살펴보기

ch10/04_func.html

```
<script>
  var printMsg=function(msg) {    // 무명 함수 선언
    document.write("무명 함수 : " + msg + "<br>");
  }
  function printMsg(msg){         // 기본 함수 선언
    document.write("기본 함수 : " + msg + "<br>");
  }
  printMsg("호출되었습니다.");    // 함수 호출
</script>
```

무명 함수 : 호출되었습니다.

2. 반환값 출력

```
function 함수명(매개 변수1, 매개 변수2, 매개 변수3) { // 함수 선언
    실행 문장;
    return 반환값;
}
result = 함수명(인자1, 인자2, 인자3); // 함수 호출
```




그림 10-1 함수 선언문과 호출문

예제 10-5 변수를 이용하여 반환값 출력하기

ch10/05_func.html

```
<script>
var result;
function add(name, n) {
    document.write(name + " 학생이 1부터 " + n + "까지 덧셈 수행<br>");
    var sum=0;
    for(var i=1; i<=n; i++) {
        sum+=i;
    }
    return sum;
}
result=add('홍길동', 10);
document.write("결과 : " + result + "<p/>");
result=add('이영희', 100);
document.write("결과 : " + result + "<p/>");
</script>
```

홍길동 학생이 1부터 10까지 덧셈 수행
결과 : 55

이영희 학생이 1부터 100까지 덧셈 수행
결과 : 5050

2. 반환값 출력

예제 10-6 변수 없이 반환값 출력하기

ch10/06_func.html

```
<script>
function add(name, n) {
    document.write(name + " 학생이 1부터 " + n + "까지 덧셈 수행<br>");
    var sum=0;
    for(var i=1; i<=n; i++) {
        sum+=i;
    }
    return sum;
}
document.write("결과 : " + add('홍길동', 10) + "<p/>");
document.write("결과 : " + add('이영희', 100) + "<p/>");
</script>
```

홍길동 학생이 1부터 10까지 덧셈 수행
결과 : 55

이영희 학생이 1부터 100까지 덧셈 수행
결과 : 5050

2. 반환값 출력

예제 10-7 서로 다른 변수로 같은 함수의 반환값 출력하기

ch10/07_func.html

```
<script>
  function add(x, y) {
    return x+y;
  }
  var calSum=add;    // 함수를 변수에 할당
  var addUp=add;     // 함수를 변수에 할당
  document.write("결과 값 : " + calSum(5, 10) + "<br>");
  document.write("결과 값 : " + addUp(3, 20) + "<br>");
</script>
```

결과 값 : 15
결과 값 : 23

3. 인자와 매개 변수

```
function 함수명(매개 변수1, 매개 변수2, 매개 변수3) { // 함수 선언
    실행 문장;
    return 반환값;
}
result=함수명(인자1, 인자2, 인자3); // 함수 호출
```



그림 10-2 인자와 매개 변수

3. 인자와 매개 변수

예제 10-8 서로 다른 변수로 같은 함수의 반환값 출력하기

ch10/07_func.html

```
<script>
function add() {
    var sum=1;
    return sum;
}
function add(x) {
    var sum=x+1;
    return sum;
}
function add(x, y) {
    var sum=x+y;
    return sum;
}
function add(x, y, z) {
    var sum=x+y+z;
    return sum;
}
var r0=add();
var r1=add(1);
var r2=add(2, 3);
var r3=add(4, 5 ,6);
var r4=add(7, 8, 9, 10);
document.write("함수 호출 인자 없음 : " + r0 + "<p/>");
document.write("함수 호출 인자 부족 : " + r1 + "<p/>");
document.write("함수 호출 인자 부족 : " + r2 + "<p/>");
document.write("정상적인 함수 호출 : " + r3 + "<p/>");
document.write("7, 8, 9만 인자값으로 적용 : " + r4 + "<p/>");
</script>
```

함수 호출 인자 없음 : NaN

함수 호출 인자 부족 : NaN

함수 호출 인자 부족 : NaN

정상적인 함수 호출 : 15

7, 8, 9만 인자값으로 적용 : 24

3. 인자와 매개 변수

예제 10-9 인자의 개수가 적을 때 처리 방법 살펴보기

ch10/09_func.html

```
<script>
function add(x, y, z) {
    var sum;
    if((y===undefined) && (z===undefined)) {
        sum=x;
    }
    else if(z===undefined) {
        sum=x+y;
    }
    else {
        sum=x+y+z;
    }
    return sum;
}
var r1=add(2);
var r2=add(2, 3);
var r3=add(4, 5, 6);
document.write("함수 호출 인자 부족 : " + r1 + "<p/>");
document.write("함수 호출 인자 부족 : " + r2 + "<p/>");
document.write("정상적인 함수 호출 : " + r3 + "<p/>");
</script>
```

함수 호출 인자 부족 : 2

함수 호출 인자 부족 : 5

정상적인 함수 호출 : 15

3. 인자와 매개 변수

예제 10-10 인자를 arguments 객체로 처리하기

ch10/10_func.html

```
<script>
function add() {
    var i, sum=0;
    for(i=0; i<arguments.length; i++) {
        sum=sum+arguments[i];
    }
    document.write("수행 결과 : " + sum + "<p/>");
}
add(2, 3);
add(2, 3, 4);
add(4, 5, 6, 7, 8);
add(1, 2, 3, 4, 5, 6, 7, 8, 9, 10);
</script>
```

수행 결과 : 5

수행 결과 : 9

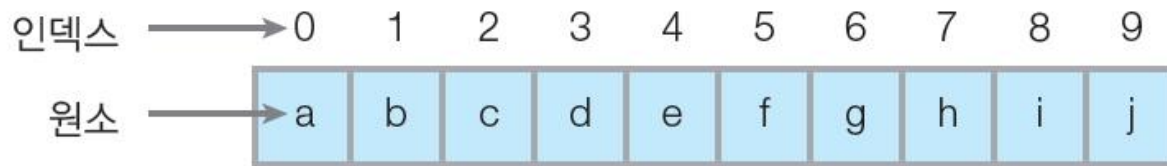
수행 결과 : 30

수행 결과 : 55

1. 배열의 개념

● 배열

- 여러 데이터 값을 저장하는 공간
- 원소: 배열에 저장된 하나 하나의 데이터
- 인덱스: 원소를 구분하는 번호, 0부터 매김



- 배열 크기 : 10
- 인덱스 : 0~9
- 인덱스 8의 데이터 값 : i

그림 10-3 배열의 구조

2. 배열 생성

● 배열 리터럴로 생성하기

```
var 배열명=[원소1, 원소2, 원소3, ... ];
```

예제 10-11 배열 변수에 초깃값을 할당하여 배열 만들기

ch10/11_arr.html

```
<script>
var city=["Seoul","Busan","Incheon"]; // 배열 리터럴
function printArr() {
    var i;
    for(i=0; i<city.length; i++) {
        document.write("배열 데이터["+ i + "] = " + city[i] + "<br>");
    }
}
printArr();
</script>
```

```
배열 데이터[0] = Seoul
배열 데이터[1] = Busan
배열 데이터[2] = Incheon
```

2. 배열 생성

예제 10-12 배열 변수 먼저 선언하고 원소 값을 따로 할당하기

ch10/12_arr.html

```
<script>
var city=[];    // 배열 변수 선언
city[0]="Seoul";
city[1]="Busan";
city[2]="Incheon";
city[3]="Mokpo";
city[4]="Sejeong";
function printArr(){
    var i;
    for(i=0; i<city.length; i++) {
        document.write("배열 데이터 [" + i + "] = " + city[i] + "<br>");
    }
}
printArr();
</script>
```

배열 데이터 [0] = Seoul
배열 데이터 [1] = Busan
배열 데이터 [2] = Incheon
배열 데이터 [3] = Mokpo
배열 데이터 [4] = Sejeong

2. 배열 생성

예제 10-13 배열에 공백 데이터 포함하기

ch10/13_arr.html

```
<script>
  var city=["Seoul", , "Busan", , "Incheon"];    // 공백 리터럴 포함
  function printArr() {
    var i;
    for(i=0; i<city.length; i++) {
      document.write("배열 데이터 [" + i + "] = " + city[i] + "<br>");
    }
  }
  printArr();
</script>
```

배열 데이터 [0] = Seoul
배열 데이터 [1] = undefined
배열 데이터 [2] = Busan
배열 데이터 [3] = undefined
배열 데이터 [4] = Incheon

2. 배열 생성

예제 10-14 공백 데이터를 포함한 배열 연산하기

ch10/14_arr.html

```
<script>
var com=[95, 88, ,72 ,68, ,99 ,82 ,78, 85]; // 10명 중 8명의 점수만 입력
var getAvg;
function printAvg() {
    var i, sum=0;
    var n=com.length;
    document.write(n + "명의 점수 입력<p/>");
    for(i=0; i<n; i++) {
        sum+=com[i];
    }
    return (sum/n);
}
getAvg=printAvg(); // 함수 호출
document.write("평균 : <b>" + getAvg + "</b><p/>");
</script>
```

10명의 점수 입력

평균 : NaN

2. 배열 생성

예제 10-15 공백 데이터 제외하고 연산하기

ch10/15_arr.html

```
<script>
var com=[95, 88, ,72 ,68, ,99 ,82 ,78, 87];    // 10명 중 8명의 점수만 입력
var getAvg ;
function printArr() {
    var i;
    var sum=0;
    var count=0;    // 입력 점수 카운트 변수
    var n=com.length;
    document.write( n + "명의 점수 입력<p/>");
    for(i=0; i<n; i++) {
        if(com[i]==undefined) {    // 점수가 입력되지 않은 학생은 연산하지 않음
            continue;
        }
        else {
            sum+=com[i];
            count++
        }
    }
    document.write("점수를 입력한 학생 : " + count + "명<p/>");
    document.write("총합 : " + sum + "<p/>");
    return (sum/count);
}
getAvg=printArr();
document.write("평균 : " + getAvg + "<p/>");
</script>
```

10명의 점수 입력

점수를 입력한 학생 : 8명

총합 : 669

평균 : 83.625

2. 배열 생성

예제 10-16 배열에 다양한 데이터 타입을 가진 원소 저장하기

ch10/16_arr.html

```
<script>
  var x=5;
  var arr=[100, "Seoul", true, x];    // 다양한 데이터 타입 저장
  function printArr() {
    var i;
    for(i=0; i<arr.length; i++) {
      document.write("배열 데이터 [" + i + "] = " + arr[i] + "<br>");
    }
  }
  printArr();    // 함수 호출
</script>
```

배열 데이터 [0] = 100
배열 데이터 [1] = Seoul
배열 데이터 [2] = true
배열 데이터 [3] = 5

2. 배열 생성

예제 10-17 같은 데이터 타입을 가진 배열 연산하기

ch10/17_arr.h

```
<script>
var arr=[10, 20, 30, 40, 50];    // 같은 데이터 타입 요소
function printArr() {
    var i, sum=0;
    for(i=0; i<arr.length; i++) {
        sum+=arr[i];
    }
    return sum;
}
var result=printArr();    // 함수호출
document.write("배열 원소 합 : " + result + "<br>");
</script>
```

배열 원소 합 : 150

2. 배열 생성

예제 10-18 다른 데이터 타입을 가진 배열 연산하기

ch10/18_arr.htm

```
<script>
  var arr=[10, 20, 30, 40, '50'];    // 다른 데이터 타입 요소
  function printArr() {
    var i, sum=0;
    for(i=0; i<arr.length; i++) {
      sum+=arr[i];
    }
    return sum;
  }
  var result=printArr();
  document.write("배열 원소 합 : " + result + "<br>");
</script>
```

배열 원소 합 : 10050

2. 배열 생성

● 배열 객체로 생성하기

```
var 배열명=new Array(원소1, 원소2, 원소3, ... );
```

예제 10-19 배열 객체 생성하기

ch10/19_arr.html

```
<script>
var city=new Array("Seoul","Busan","Incheon");
function printArr() {
    var i;
    for(i=0; i<city.length; i++) {
        document.write("배열 데이터 ["+ i + "] = " + city[i] + "<br>");
    }
}
printArr();
</script>
```

```
배열 데이터 [0] = Seoul
배열 데이터 [1] = Busan
배열 데이터 [2] = Incheon
```

2. 배열 생성

● 배열 객체 생성 확인 방법

방법	사용 예	결과
타입 확인 연산자인 typeof 사용	typeof city	object
배열 객체의 메소드인 isArray() 사용	Array.isArray(city)	true
Array 생성자의 연산자인 instanceof 사용	city instanceof Array	true

2. 배열 생성

예제 10-20 배열 객체 생성 확인하기

ch10/20_arr.html

```
<script>
var city=new Array("Seoul","Busan","Incheon");
function printArr() {
    if(city instanceof Array) {
        document.write("배열 객체가 생성되었습니다.<p/>");
        var i;
        for(i=0; i<city.length; i++) {
            document.write("배열 데이터 [" + i + "] = " + city[i] + "<br>");
        }
    }
    else {
        document.write("배열 객체가 아닙니다.<br>");
        document.write("데이터 : " + city + "<br>");
    }
}
printArr();
document.write("<p/> city 변수 타입 : " + typeof city + "<br>");
document.write("배열 객체 확인 결과 : " + Array.isArray(city) + "<br>");
</script>
```

배열 객체가 생성되었습니다.

배열 데이터 [0] = Seoul

배열 데이터 [1] = Busan

배열 데이터 [2] = Incheon

city 변수 타입 : object

배열 객체 확인 결과 : true

3. 배열 데이터 접근 및 조작

예제 10-21 배열에 1부터 100까지 저장한 후 모두 더하기

ch10/21_arr.html

```
<script>
var arrdata=[];
function insertArr() {    // 배열 데이터 입력 함수
    var i=0;
    for(i=0; i<=99; i++) {
        arrdata[i]=i+1;    // 1~100까지 저장
    }
    selectArr();
}
function selectArr() {    // 배열 데이터 조회 함수
    var i;
    for(i=0; i<arrdata.length; i++) {
        document.write(arrdata[i] + " ");    // 데이터 조회
    }
    addArr();
}
function addArr() {        // 배열 데이터 덧셈 함수
    var i;
    var sum=0;
    for(i=0; i<arrdata.length; i++) {
        sum+=arrdata[i];    // 덧셈 연산
    }
    document.write("<p/> 배열 데이터 덧셈 연산 결과 : " + sum + "<p/>");
    document.write("<a href='21_arr.html'>돌아가기</a>");
}
</script>
<button type="button" onclick="insertArr()">배열 생성/조회/연산</button>
```

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21
 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38
 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55
 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72
 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89
 90 91 92 93 94 95 96 97 98 99 100

배열 데이터 덧셈 연산 결과 : 5050

[돌아가기](#)

3. 배열 데이터 접근 및 조작

예제 10-22 홀수 번째 저장된 데이터만 0으로 초기화하기

ch10/22_arr.html

```
<script>
var arrdata=[];
function insertArr() {
    var i=0;
    for(i=0; i<=99; i++) {
        arrdata[i]=i+1;
        document.write(arrdata[i] + " ");
    }
}
function delArr() {
    var i;
    for(i=0; i<arrdata.length; i++) {
        if(i%2==0) {
            arrdata[i]=0;
        }
        continue;
    }
    selectArr();
}
function selectArr() {
    var i;
    for(i=0; i<arrdata.length; i++) {
        document.write(arrdata[i] + " ");
    }
    document.write("<p>홀수 번째 데이터 초기화 완료!" + "</p>");
    document.write("<a href='22_arr.html'>돌아가기</a>");
}
insertArr();
</script>
<p/>
<button type="button" onclick="delArr()">배열의 홀수 번째 데이터 초기화</button>
```

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21
 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38
 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55
 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72
 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89
 90 91 92 93 94 95 96 97 98 99 100

배열의 홀수 번째 데이터 초기화

0 2 0 4 0 6 0 8 0 10 0 12 0 14 0 16 0 18 0 20 0 22 0
 24 0 26 0 28 0 30 0 32 0 34 0 36 0 38 0 40 0 42 0 44
 0 46 0 48 0 50 0 52 0 54 0 56 0 58 0 60 0 62 0 64 0
 66 0 68 0 70 0 72 0 74 0 76 0 78 0 80 0 82 0 84 0 86
 0 88 0 90 0 92 0 94 0 96 0 98 0 100

홀수 번째 데이터 초기화 완료!

[돌아가기](#)

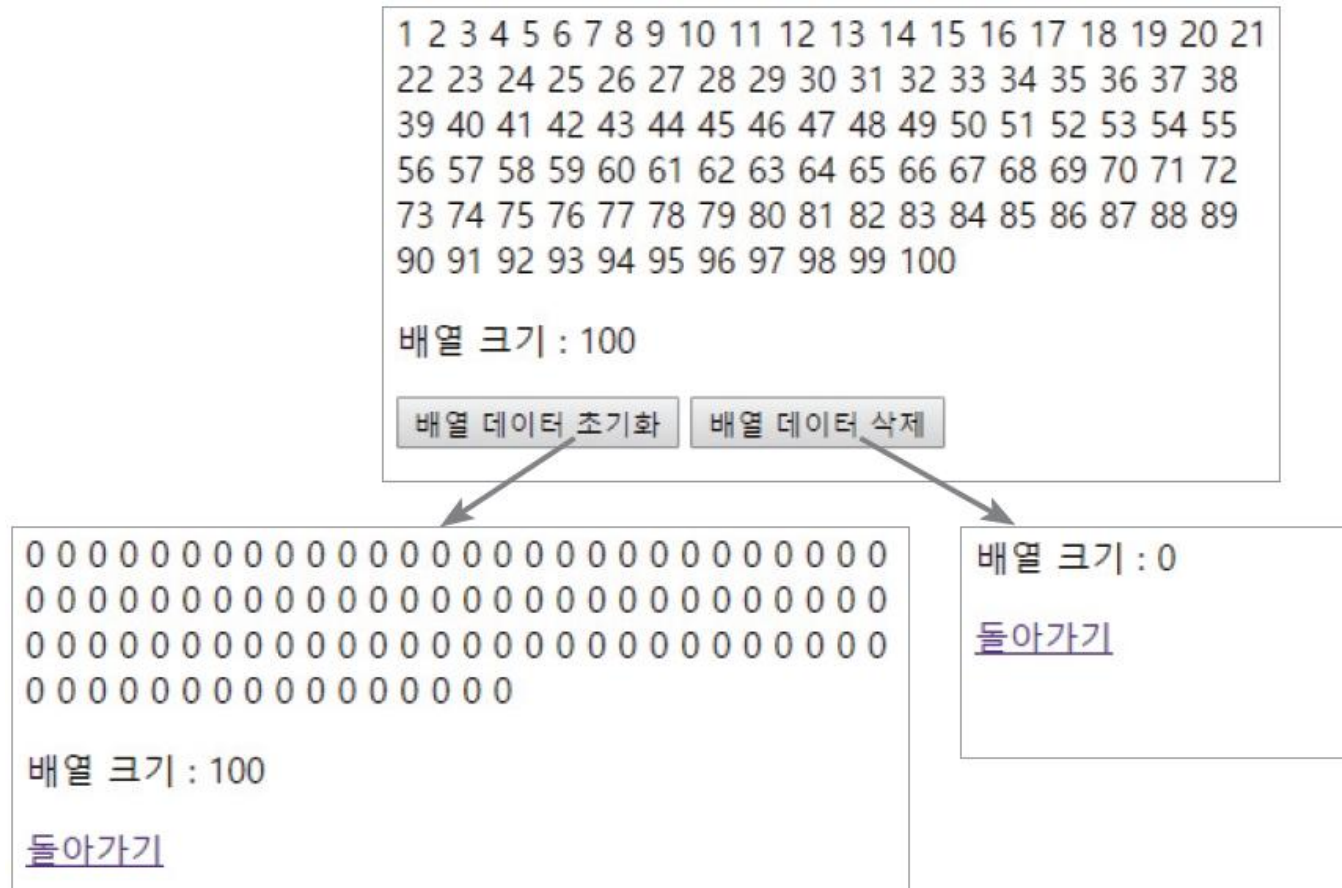
3. 배열 데이터 접근 및 조작

예제 10-23 배열에 저장된 데이터 삭제하기

ch10/23_arr.html

```
<script>
var arrdata=[];
function insertArr() {
    var i=0;
    for(i=0; i<=99 ; i++) {
        arrdata[i]=i+1;      // 1~100 저장
        document.write(arrdata[i] + " ");  // 데이터 출력
    }
    document.write("<p>배열 크기 : " + arrdata.length + "</p>");
}
function delDataArr() {
    var i;
    for(i=0; i<arrdata.length; i++) {
        arrdata[i]=0;        // 배열 데이터를 0으로 초기화
    }
    selectArr();
}
function allDelArr() {
    arrdata.length=0;    // 배열 초기화
    selectArr();
}
function selectArr() {
    var i;
    for(i=0; i <arrdata.length; i++) {
        document.write(arrdata[i] + " ");    // 데이터 조회
    }
    document.write("<p> 배열 크기 : " + arrdata.length + "</p>");
    document.write("<a href='23_arr.html'>돌아가기</a>");
}
insertArr();    // 배열 데이터 생성 함수 호출
</script>
<p/>
<button type="button" onclick="delDataArr()">배열 데이터 초기화</button>
<button type="button" onclick="allDelArr()">배열 데이터 삭제</button>
```

3. 배열 데이터 접근 및 조작



1. join 메소드

● join

- 배열에 저장된 모든 원소를 문자열로 변환한 후 연결하여 출력

예제 10-24 join 메소드 활용하기

ch10/24_arr.html

```
<script>
var city=["서울", "부산", "대전"];
var joindata1=city.join();
var joindata2=city.join('-');
var joindata3=city.join(' 그리고 ');
document.write("조인 결과1 : " + joindata1 + "<p/>");
document.write("조인 결과2 : " + joindata2 + "<p/>");
document.write("조인 결과3 : " + joindata3 + "<p/>");
</script>
```

조인 결과1 : 서울,부산,대전

조인 결과2 : 서울-부산-대전

조인 결과3 : 서울 그리고 부산 그리고 대전

2. concat 메소드

● concat

- 지정된 배열에 두 개 이상의 데이터를 결합하거나 다른 배열 객체를 결합

예제 10-25 concat 메소드 활용하기

ch10/25_arr.html

```
<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>
```

결과1 : 서울,부산,대전,수원,오산

결과2 : 서울,부산,대전,대구,광주,인천

결과3 : 서울,부산,대전,전주,부여,세종,대구,광주,인천

3. reverse 메소드

- reverse

- 배열 원소의 순서를 반대로 정렬

예제 10-26 reverse 메소드 활용하기

ch10/26_arr.html

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

배열 : 9,8,7,6,5,4,3,2,1

결과 : 1,2,3,4,5,6,7,8,9

4. sort 메소드

● sort

- 배열 원소를 정렬

예제 10-27 sort 메소드 활용하기

ch10/27_arr.html

```
<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

5. slice 메소드

● slice

- 배열의 특정 범위에 속하는 원소만 선택하여 배열 생성

예제 10-28 slice 메소드 활용하기

ch10/28_arr.html

```
<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>
```

부분 배열1 : 서울,부산,구포,대구

부분 배열2 : 구포,대구,인천,대전

부분 배열3 : 대구,인천

6. splice 메소드

● splice

- 배열의 원소 추가 또는 제거

예제 10-29 splice 메소드 활용하기

ch10/29_arr.html

```
<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

7. pop & push 메소드

● 스택

- 모든 데이터의 삽입과 삭제가 배열의 한쪽 끝에서만 수행되는 자료 구조
 - **push 메소드**: 배열의 마지막 위치에 데이터를 추가하고 배열의 길이를 반환
 - **pop 메소드**: 배열의 마지막 위치에 있는 데이터를 삭제하고 삭제한 데이터를 반환

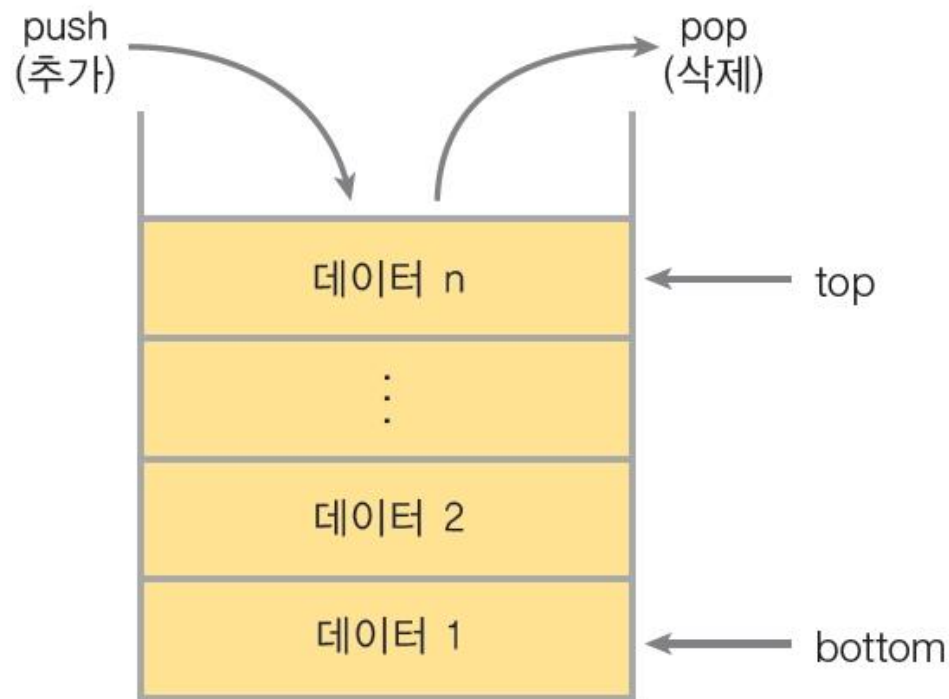


그림 10-4 스택의 구조

7. pop & push 메소드

예제 10-30 pop & push 메소드 활용하기

ch10/30_arr.html

```
<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

8. shift & unshift 메소드

● shift & unshift

- shift 메소드: 배열의 맨 처음 위치에 데이터를 삭제하고 배열의 길이 반환
- unshift 메소드: 배열의 맨 처음 위치에 데이터를 삽입하고 배열의 길이 반환

예제 10-31 shift & unshift 메소드 활용하기

ch10/31_arr.html

```
<script>
var kdata=[ '서울', '부산' ];
var p1=kdata.unshift('청주', '세종');
document.write("데이터 : " + p1 + "<br>");
document.write("배열 데이터 : " + kdata + "<p/>");
var p2=kdata.shift();
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

9. forEach 메소드

● forEach

- 배열을 반복하며 저장된 데이터를 조회

예제 10-32 forEach 메소드 활용하기 1

ch10/32_arr.html

```
<script>
  var kdata=[ '서울', '부산', '청주', '대구' ];
  function printArr(item, index) {
    document.write("배열 데이터 [" + index + "] : " + item + "<br>");
  }
  kdata.forEach(printArr);
</script>
```

```
배열 데이터 [0] : 서울
배열 데이터 [1] : 부산
배열 데이터 [2] : 청주
배열 데이터 [3] : 대구
```

예제 10-33 forEach 메소드 활용하기 2

ch10/33_arr.htm

```
<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>
```

```
배열 데이터 합 : 55
```

10. map 메소드

● map

- 배열의 데이터를 함수의 인자로 전달하고 함수의 수행 결과를 반환 받아 새로운 배열 생성

예제 10-34 map 메소드 활용하기

ch10/34_arr.html

```
<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>
```

원래 배열 :1,2,3,4,5,6,7,8,9,10

map 메소드 적용 배열 :1,4,9,16,25,36,49,64,81,100

11. filter 메소드

● filter

- 배열의 데이터 중에 조건이 참인 데이터만 반환하여 새로운 배열 생성

예제 10-35 filter 메소드 활용하기

ch10/35_arr.html

```
<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>
```

필터 전 배열 : 21,42,33,14,25,12,37,28,16,11

필터 후 배열 : 21,42,33,25,37,28

12. indexOf & lastIndexOf 메소드

● indexOf & lastIndexOf

- 배열의 데이터를 검색하여 인덱스 위치를 반환
 - **indexOf 메소드**: 검색 시작 위치를 지정할 수 있음
 - **lastIndexOf 메소드**: 배열의 맨 마지막 원소부터 검색 시작

예제 10-36 indexOf & lastIndexOf 메소드 활용하기

ch10/36_arr.html

```
<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>
```

```
배열 데이터 : [10,20,30,40,30,60,70,30,90,100]

처음부터 검색한 30의 인덱스 : 2

마지막에서 검색한 30의 인덱스 : 7

세 번째부터 검색한 30의 인덱스 : 4

처음부터 검색한 300의 인덱스 : -1
```

1. 연관 배열

● 연관 배열 생성 방법

```
arr={key_1:value1, key_2:value2, ..... , key_n:value_n};
```

예제 10-37 연관 배열로 저장된 데이터 조회하기

ch10/37_arr.html

```
<script>
var data={'f0':100, 'f1':200, 'f2':300};
data['f3']=400;    // 배열 데이터 저장
data.f4=500;      // 배열 데이터 저장
document.write(data.f0 + "<br>");    // 'f0'키 데이터 조회
document.write(data.f1 + "<br>");    // 'f1'키 데이터 조회
document.write(data['f2'] + "<br>");
document.write(data['f3'] + "<br>");
document.write(data['f4'] + "<br>");
</script>
```

```
100
200
300
400
500
```

2. 2차원 배열

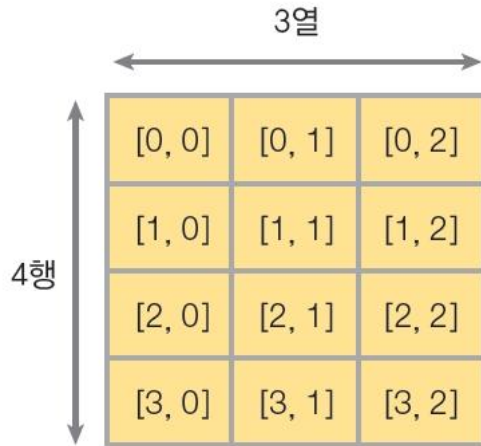


그림 10-5 2차원 배열의 구조

예제 10-38 2차원 배열 생성하고 조회하기

ch10/38_arr.html

```
<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>
```

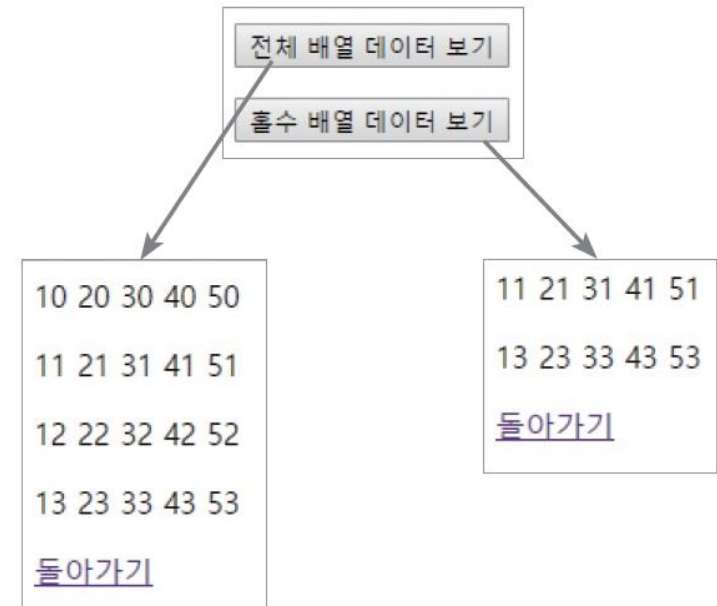
```
2차원 배열 첫 번째 데이터 : 10
2차원 배열 마지막 데이터 : 100
2차원 배열 행 길이 : 2
2차원 배열 열 길이 : 5
```

2. 2차원 배열

예제 10-39 1차원 배열로 2차원 배열 생성하고 조회하기

ch10/39_arr.html

```
<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>
```



2. 2차원 배열

예제 10-40 반복문을 이용하여 2차원 배열 만들기

Ch10/40_arr.html

```
<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>
```

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](#)