

### 3. 다중 if~else문

- 다중 if~else문

- 여러 조건을 체크해야 할 때 사용

```
if(조건식-1) {  
    조건식-1의 결과가 참일 때 실행할 문장;  
}  
else if(조건식-2) {  
    조건식-2의 결과가 참일 때 실행할 문장;  
}  
else {  
    조건식-1, 조건식-2 모두 거짓일 때 실행할 문장;  
}
```

### 3. 다중 if~else문

예제 9-21 학점 환산 프로그램 만들기

ch09/21\_js.html

```
<script>
  var point=93;  // 과목 점수
  var grade=" ";
  if(point>100) {
    document.write("0~100점 사이 값을 입력해야 합니다." + "<p/>");
  }
  else if(point>=90) {
    grade="A";
    document.write("아주 잘했어요." + "<p/>");
  }
  else if(point>=80) {
    grade="B";
    document.write("잘했어요." + "<p/>");
  }
  else if(point>=70) {
    grade="C";
    document.write("조금만 노력하면 잘할 수 있어요." + "<p/>");
  }
  else if(point>=60) {
    grade="D";
    document.write("좀 더 노력하세요." + "<p/>");
  }
  else{
    grade="F";
    document.write("많이 노력하시기 바랍니다." + "<p/>");
  }
  document.write("학생의 학점은 <b>" + grade + "</b>입니다.<p/>");
</script>
```

아주 잘했어요.

학생의 학점은 A입니다.

## 4. switch~case문

### ● switch~case문

- 조건문을 체크하여 다음에 처리할 문장의 위치를 파악한 후 해당 문장으로 가서 바로 처리

```
switch(상수값) {  
    case n:  
        실행 문장;  
        break;  
    case n:  
        실행 문장;  
        break;  
    default:  
        기본 실행 문장;  
}
```

## 4. switch~case문

예제 9-22 요일을 알려주는 프로그램 만들기

ch09/22\_js.html

```
<script>
var day;
var week = new Date().getDay(); // 0(일요일)~6(토요일)
switch(week) {
    case 0:
        day = "일요일";    break;
    case 1:
        day = "월요일";    break;
    case 2:
        day = "화요일";    break;
    case 3:
        day = "수요일";    break;
    case 4:
        day = "목요일";    break;
    case 5:
        day = "금요일";    break;
    case 6:
        day = "토요일";    break;
    default:
        day = "없는 요일";
}
document.write("오늘은 <b>" + day + "</b>입니다. <p/>");
</script>
```

오늘은 월요일입니다.

## 4. switch~case문

예제 9-23 요일별 일정을 알려주는 프로그램 만들기

ch09/23\_js.html

```
<script>
var text;
var week=new Date().getDay(); // 0(일요일)~6(토요일)
switch(week) {
    case 1: // 월요일
    case 2: // 화요일
        text="HTML5";
        break;
    case 3: // 수요일
    case 4: // 목요일
        text="자바스크립트";
        break;
    case 5: // 금요일
    case 6: // 토요일
        text="영어";
        break;
    case 0: //일요일
    default:
        text="수영";
}
document.write("오늘은 <b>" + text + "</b> 학습하는 날입니다. <p/>");
</script>
```

오늘은 HTML5 학습하는 날입니다.

## 5. for문

### ● for문 형식

```
for(초기식; 조건식; 증감식) {  
    실행 문장;  
}
```

- 초기식 : 반복 변수값을 초기화하며, for문이 처음 시작할 때 단 한 번만 실행됨
- 조건식 : 블록 내 문장을 얼마나 반복할지 결정하며, 조건식이 참인 동안 반복함
- 증감식 : 초기식에서 초기화한 변수의 값을 증가 또는 감소시킴

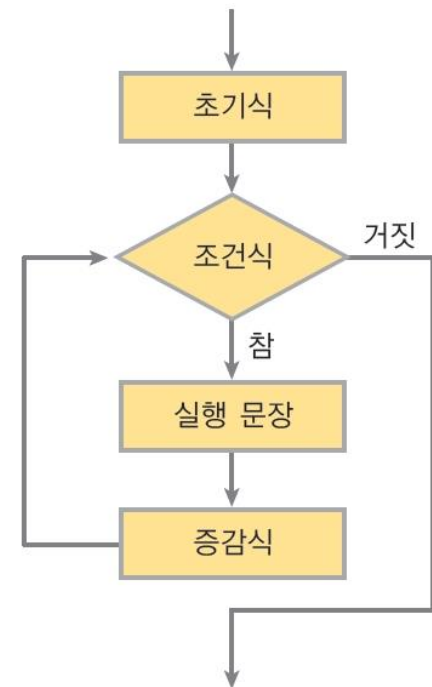


그림 9-7 for문의 순서도

## 5. for문

### ● for문의 변칙적 사용

- ① 초기식을 for문 이전에 먼저 선언을 했다면 for문에서는 생략 가능

```
<script>
  var num =1;    // 초깃값 선언
  for(   ; num<=5; num++) {
    document.write("for문 수행 : <b>" + num + "</b> <p/>");
  }
</script>
```

- ② 초기식은 여러 개 선언할 수 있음

```
<script>
  var num;
  var st="ABCDEF";    // 문자열 길이 6
  var ct;
  for(num=1, ct=st.length; num<ct; num++) {
    document.write("for문 수행 : <b>" + num + "</b> <p/>");
  }
</script>
```

## 5. for문

### ● for문의 변칙적 사용

- ③ for문의 블록 내에 증감식 문장을 포함한다면 for문 자체에서 증감식을 생략해도 됨

```
<script>
  var num =1;
  var st="ABCDEF";
  var ct=st.length;
  for(   ; num<ct;   ) {
    document.write("for문 수행 : <b>" + num + "</b> <p/>");
    num++;
  }
</script>
```



## 5. for문

### ● for문의 변칙적 사용

- ④ for( ; ; )와 같이 초기식, 조건식, 증감식을 모두 작성하지 않으면 블록 내 문장을 무한 반복하게 됨

```
<script>
  var num=1;
  var st="ABCDEF";
  var ct=st.length;
  for( ; ; ) {
    if(num<ct){
      document.write(" for문 수행 : <b>" + num + "</b> <p/>");
      num++;
      continue;
    }
    break;
  }
</script>
```

## 5. for문

예제 9-24 구구단 프로그램 만들기

ch09/24\_js.html

```
<script>
  var x, y;
  for(x=2; x<=5; x++) {
    document.write("<b> ---[" + x + "단]--- </b>" + "<br>");
    for(y=1; y <= 9; y++) {
      document.write(x + "*" + y + "=" + (x * y) + "<br>");
    }
  }
</script>
```

---[2단]---

2\*1=2

2\*2=4

2\*3=6

2\*4=8

2\*5=10

2\*6=12

2\*7=14

## 6. while문

- while문의 형식

```
while(조건식) {  
    실행 문장;  
}
```

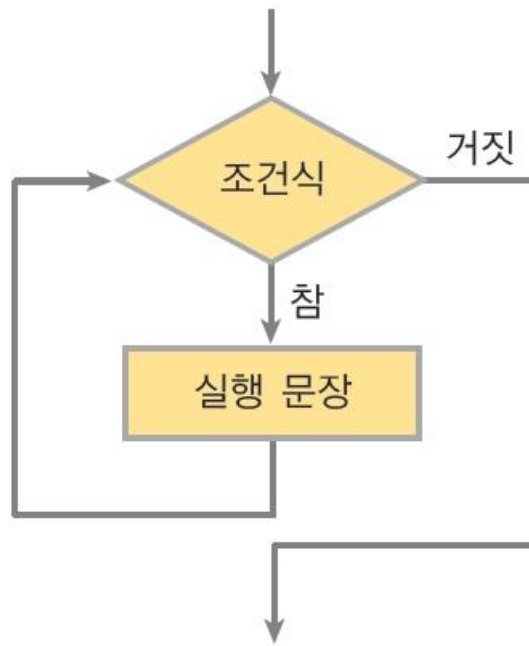


그림 9-8 while문의 순서도

## 6. while문

예제 9-25 1부터 100까지 합 구하기

ch09/25\_js.html

```
<script>
  var x=1;
  var sum=0;
  while(x<=100) {
    sum+=x;
    x++;
  }
  document.write("1~100까지 합 : <b>" + sum + "</b>");
</script>
```

1~100까지 합 : 5050

예제 9-26 1부터 10000까지 합 구하기

ch09/26\_js.html

```
<script>
  var x=1;
  var sum=0;
  while(1) { // 무한 반복
    sum += x;
    x++;
    if(x==10001)
      break;
  }
  document.write("1~10000까지 합 : <b>" + sum + "</b>");
</script>
```

1~10000까지 합 : 50005000

## 7. do~while문

### ● do~while문의 형식

```
do {  
    실행 문장;  
}  
while(조건식);
```

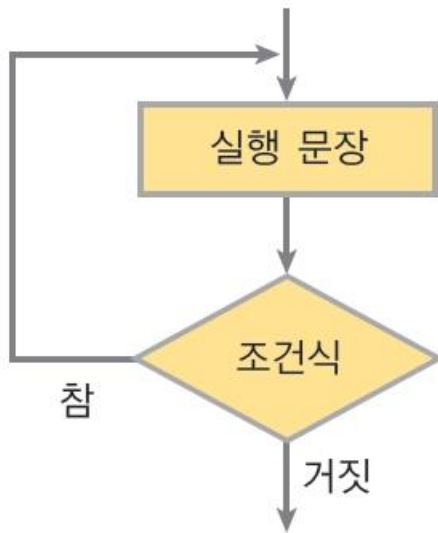


그림 9-9 do~while문의 순서도

예제 9-27 do~while문으로 1부터 100까지 합 구하기

ch09/27\_js.html

```
<script>  
    var x=1;  
    var sum=0;  
    do {  
        sum+=x;  
        x++;  
    } while(x<=100);  
    document.write("1~100까지 합 : <b>" + sum + "</b>");  
</script>
```

1~100까지 합 : 5050

## 8. break문

### ● break문

- for문, while문, do~while문과 같은 반복문이나 switch~case문 내에서 해당 블록을 강제로 벗어나 다음 문장을 처리하도록 할 때 사용

예제 9-28 break문으로 1부터 100까지 수 중 3의 배수 합 구하기

ch09/28\_js.html

```
<script>
var x=0;
var sum=0;
while(1) {
    x += 3; // 3의 배수
    if(x>100)
        break;
    sum += x;
    document.write(x + " ");
}
document.write("<p/>");
document.write("1~100까지 수 중 3의 배수 합 : <b>" + sum + "</b>");
</script>
```

3 6 9 12 15 18 21 24 27 30 33 36 39 42 45 48 51 54 57 60 63 66 69  
72 75 78 81 84 87 90 93 96 99

1~100까지 수 중 3의 배수 합 : 1683

## 9. continue문

### ● continue문

- if문의 조건식이 참이면 continue문 이후의 문장을 처리하지 않고 제어를 반복문의 시작 위치로 옮김

예제 9-29 continue문으로 1부터 100까지 수 중 3의 배수 합 구하기

ch09/29\_js.html

```
<script>
var x=0;
var sum=0;
for(x=1; x<=100; x++) {
    if(x%3 != 0) //3으로 나누었을 때 0이면 계속 아래 코드를 진행해라
        continue;
    sum += x;
    document.write(x + " ");
}
document.write("<p/>");
document.write("1~100까지 수 중 3의 배수 합 : <b>" + sum + "</b>");
</script>
```

3 6 9 12 15 18 21 24 27 30 33 36 39 42 45 48 51 54 57 60 63 66 69  
72 75 78 81 84 87 90 93 96 99

1~100까지 수 중 3의 배수 합 : 1683

## Chapter 10

# 자바스크립트 함수와 배열



# C.ontents

---

**01** 자바스크립트 함수

**02** 자바스크립트 배열

**03** 배열 관련 메소드

**04** 연관 배열과 2차원 배열

# 학습목표

---

- 함수 선언 및 호출 방법을 알고 함수를 이용한 프로그램을 작성할 수 있다.
- 배열 생성 방법을 알고 배열을 이용한 프로그램을 작성할 수 있다.
- 배열 관련 메소드의 종류와 역할을 알고 메소드를 이용한 프로그램을 작성할 수 있다.
- 연관 배열과 2차원 배열의 개념을 알고 이를 활용하여 프로그램을 작성할 수 있다.

# 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 입력 값
- 매개 변수 : 함수 호출문에서 전달한 인자를 받기 위해 선언된 변수
- function : 함수를 선언할 때 사용하는 키워드
- return : 함수에서 수행한 결과 값을 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
```



The diagram illustrates the relationship between function parameters and arguments. Three red arrows point from the arguments '인자1', '인자2', and '인자3' in the function call to the corresponding parameters '매개 변수1', '매개 변수2', and '매개 변수3' in the function definition. The parameter names are highlighted in yellow in the original image.

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