

연산자와 조건문

연산자

연산자

아래와 같은 식이 있을 때

$$\text{age} = \text{currentYear} - \text{birthYear} + 1$$

age, currentYear, birthYear, 1은 연산 대상이 되기 때문에 '피연산자'라고 부름

위 식에서 '='와 '-', '+'은 연산자

산술 연산자

수학 계산을 할 때 사용하는 연산자

종류	설명	예시
+	두 피연산자의 값을 더합니다.	$c = a + b$
-	첫 번째 피연산자 값에서 두 번째 피연산자 값을 뺍니다.	$c = a - b$
*	두 피연산자의 값을 곱합니다.	$c = a * b$
/	첫 번째 피연산자 값을 두 번째 피연산자 값으로 나눕니다.	$c = a / b$
%	첫 번째 피연산자 값을 두 번째 피연산자 값으로 나눈 나머지를 구합니다.	$c = a \% b$
++	피연산자를 1 증가시킵니다.	$a++$
--	피연산자를 1 감소시킵니다.	$b--$

- 나누기 연산자(/) : 나눈 값 자체
- 나머지 연산자(%) : 나눈 후에 남은 나머지 값

산술 연산자 - 증감 연산자

- 증가(++) 연산자 : 변수값을 1 증가시킴
 - 감소(--) 연산자 : 변수값을 1 감소시킴
- 합쳐서 증감 연산자라고 부름
- 증감 연산자는 단항 연산자

```
a = 10
sum = a + 5    // 15
a              // 10
```

```
sum = a++ + 5  // 15
a              // 11
```

증감 연산자는 연산자가 어느 위치에 붙느냐에 따라
처리 방법이 달라짐

```
a = 10
a++      // 10
a        // 11
```

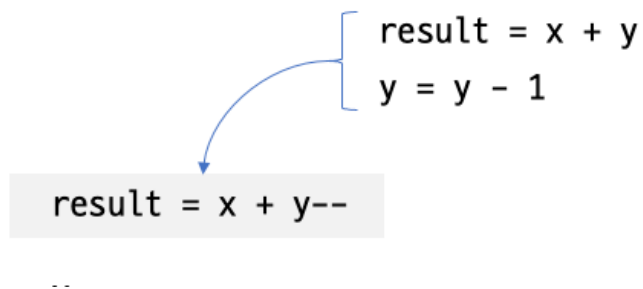
```
a = 10
++a      // 9
a        // 9
```

산술 연산자 - 증감 연산자

증감 연산자는 연산자가 어느 위치에 붙느냐에 따라 처리 방법이 달라짐

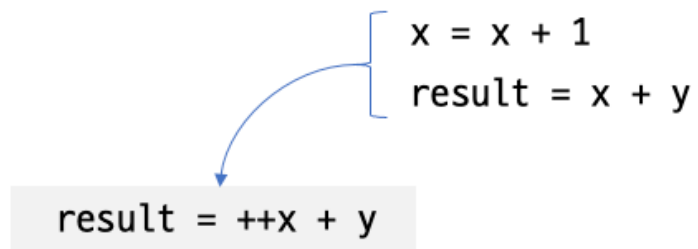
```
x = 10, y = 4, result
```

```
result = x + y--    // 14  
y                  // 3
```



```
result = x + y--  
..
```

```
result = ++x - y    // 8  
x                  // 3
```



```
result = ++x + y
```

할당 연산자 (대입 연산자)

연산자 오른쪽의 실행 결과를 왼쪽 변수에 할당하는 연산자
산술 연산자와 할당 연산자를 묶어서 표현할 수 있다

종류	설명	예시
=	연산자 오른쪽의 값을 왼쪽 변수에 할당합니다.	<code>y = x + 3</code>
+=	<code>y = y + x</code> 를 의미합니다.	<code>y += x</code>
-=	<code>y = y - x</code> 를 의미합니다.	<code>y -= x</code>
*=	<code>y = y * x</code> 를 의미합니다.	<code>y *= x</code>
/=	<code>y = y / x</code> 를 의미합니다.	<code>y /= x</code>
%=	<code>y = y % x</code> 를 의미합니다.	<code>y %= x</code>

```
x = 5
y = 5
y += x
y -= x
y *= x
y /= x
y %= x
```

```
> x = 5
< 5
> y = 5
< 5
> y += x
< 10
> y -= x
< 5
> y *= x
< 25
> y /= x
< 5
> y %= x
< 0
>
```

연결 연산자

- 산술 연산자의 더하기(+)를 연결 연산자로 사용함
- 문자열과 문자열을 연결하는 연산자

```
user = prompt("이름을 입력하세요.")  
alert(user + "님, 안녕하세요?")
```

```
alert(currentYear + "년 현재,\n" + birthYear + "년에 태어난 사람의 나이는 " + age + "세입니다.");
```

<문제점> 산술 연산자로 사용했는데, 연결 연산자로 인식해서 예상하지 못한 결과가 나옴.

```
result = 10;  
userNumber = prompt("10보다 작은 수 입력")  
result = result + userNumber    // result += userNumber
```


비교 연산자

- 피연산자 2개의 값을 비교해서 true나 false로 결과값 반환
- 비교 연산자는 조건을 확인할 때 자주 사용하는 연산자
- 나중에 공부할 AND, OR, || 연산자와 함께 사용해서 복잡한 조건도 체크할 수 있음

== , !=	두 개의 값이 같은지, 같지 않은지 확인
<, <=	왼쪽 값이 오른쪽 값보다 작은지 혹은 작거나 같은지 확인
>, >=	왼쪽 값이 오른쪽 값보다 큰지 혹은 크거나 같은지 확인
=== , !==	두 개의 값이 자료형까지 완벽하게 같은지, 같지 않은지 확인

3 < 4

3 <= 4

3 > 4

3 >= 4

비교 연산자

== 연산자 와 != 연산자

피연산자의 자료형을 자동으로 변환해서 비교

```
3 == "3" // true  
3 != "3" // false
```

=== 연산자 와 !== 연산자

피연산자의 자료형까지 정확하게 맞는지 비교

```
3 === "3" // false  
3 !== "3" // true
```

비교 연산자

컴퓨터에서 문자를 숫자에 일대일로 대응한 값을 가리킨다.

아스키값을 정리한 표를 '아스키 코드 테이블'이라고 하며 인터넷에서 검색할 수 있다

문자열의 비교

피연산자가 문자열이라면 문자열에 있는 문자들의 **아스키_{ASCII}값**을 비교해서 결정한다.

대략적인 아스키값 순서 : 제어 문자 < 특수 기호 < 숫자 < 영대문자 < 영소문자

```
"A" > "B"           // false ("B"의 아스키값이 더 크므로)
"A" < "a"           // true (영소문자의 아스키값이 숫자보다 크므로)
```

글자가 여러 개인 문자열을 비교할 경우 맨 앞의 문자부터 하나씩 비교한다.

```
"Javascript" < "JavaScript" // false (소문자 아스키값 > 대문자 아스키값)
```

논리 연산자

불리언^{boolean} 연산자라고도 하며 true, false를 처리하는 연산자
프로그램에서 조건을 처리할 때 사용하는 연산자
조건문을 공부할 때 자세히 다룰 예정

종류	기호	설명
OR 연산자		피연산자 중 하나만 true여도 true가 됩니다.
AND 연산자	&&	피연산자가 모두 true일 경우에만 true가 됩니다.
NOT 연산자	!	피연산자의 반댓값을 지정합니다.

연산자 우선 순위

단항 연산자 → 산술 연산자 → 비교 연산자 → 논리 연산자 → 할당 연산자

	1st	2nd	3rd	4th	5th	6th	7th
단항 연산자	!	++	--				
산술 연산자	*	/	%	+	-		
비교 연산자	<	<=	>	>=	==	!=	===
논리 연산자	&&						
할당 연산자	=	+=	-=	*=	/=	%=	

조건문

if문

괄호 안의 조건이 true이면 { } 사이의 명령을 처리하고,
false 이면 { } 안의 명령 무시하고 다음 명령 처리

```
if(조건) {  
    조건이 true일 때 실행할 명령  
}
```

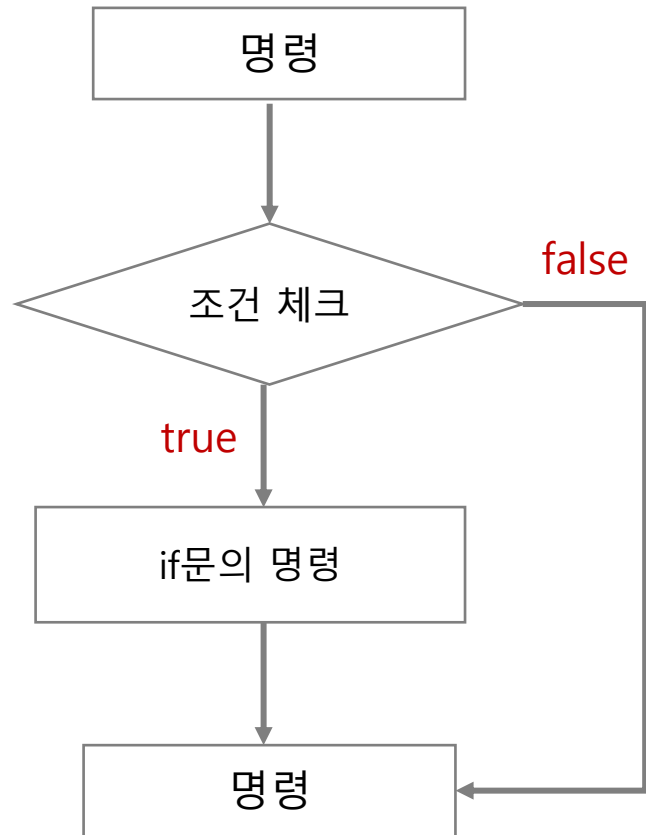
if ... else 문

if 문은 결괏값이 true일 때만 실행하므로 true가 아닐 때 명령을 따로 수행할 수 없다.

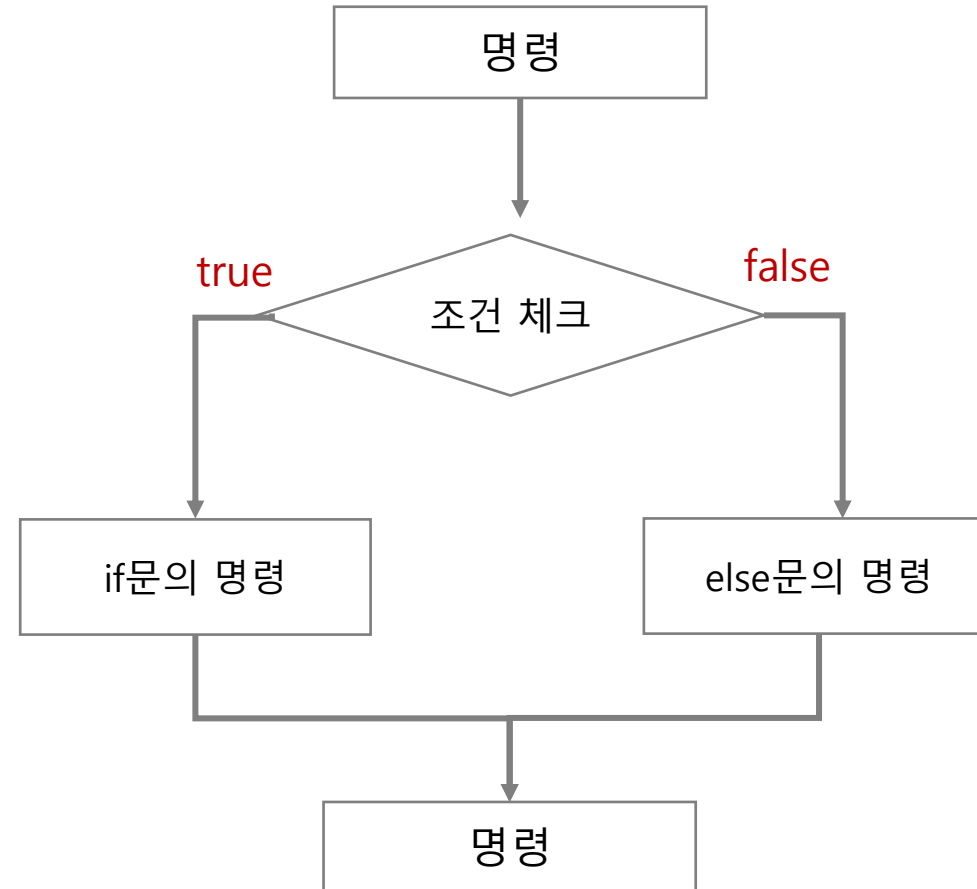
if~else 문은 if 조건의 결괏값이 true가 아닐 때 실행할 명령을 else 문 다음에 추가한다.

```
if(조건) {  
    조건 결괏값이 true일 때 실행할 명령  
} else {  
    조건 결괏값이 false일 때 실행할 명령  
}
```


if문은 조건을 체크한 후 조건에 맞을 경우
에만 if문 안에 있는 명령을 실행하고 바로
다음 명령으로 넘어감



if...else문은 조건을 체크한 후
if문의 명령이나 else문의 명령 중 하나를
실행한 후에야 다음 명령으로 넘어감



if ... else 문

```
if(조건) {  
    ...  
} else {  
    ...  
}
```

둘다 가능

```
if(조건) {  
    ...  
}  
else {  
    ...  
}
```

if ... else 문을 계속 연결해서 사용할 수도 있다

```
if(조건1) {  
    ...  
} else if(조건2) {  
    ...  
} else {  
    ...  
}
```

조건 연산자

조건이 하나이고 실행할 명령도 하나일 때 조건문을 간단하게 처리하는 연산자

(조건)? 명령1: 명령2

```
if (num1 < num2 ) {  
    small = num1;  
} else {  
    small = num2;  
}
```



```
small = (num1 < num2) ? num1 : num2;
```

true라면

small = (num1 < num2) ? num1 : num2;

false라면

switch문

처리할 명령이 많을 경우 switch 문이 편리하다.

- switch 키워드 오른쪽에 조건을 확인할 변수 지정
- 조건값은 case문 다음에 지정
- 조건값에 맞을 때 실행할 명령은 콜론(:) 다음에 나열
- 둘 이상의 명령이라면 { } 사용
- 조건에 맞는 명령을 실행한 후에는 break문을 써서 switch문을 완전히 빠져나옴
- case의 값과 일치하는게 없을 경우 default 문 실행
- default 문에는 break 문이 없음

```
switch (변수)
{
    case 값1 : 문장
        break;
    case 값2 : 문장
        break;

    .....

    default:  문장
}
```

두 가지 이상의 조건 체크하기

두 개 이상의 조건을 체크해야 할 경우에는 논리 연산자를 사용해 조건식을 만들어야 한다.

- **OR 연산자(||)** : 두 개의 피연산자 중 하나라도 true가 있으면 결과값은 true가 된다.
- **AND 연산자(&&)** : 두 개의 피연산자 중 false가 하나라도 있으면 결과값은 false가 된다.
- **NOT 연산자(!)** : 피연산자의 값과 정반대의 값

피연산자1 피연산자2

op1	op2	op1 op2	op1 && op2	!op1
false	false	false	false	true
false	true	true	false	true
true	false	true	false	false
true	true	true	true	false

OR 연산자 (||)

피연산자 중 true가 하나라도 있으면 결과값 true

op 1	op 2	op 1 op 2
false	false	false
false	true	true
true	false	true
true	true	true

```
a = 10
b = 20
a > 10 || b > 20
a <= 10 || b > 20
a < 10 || b <= 20
a <= 10 || b <= 20
```

AND 연산자 (&&)

피연산자 중 false가 하나라도 있으면 결과값 false

op 1	op 2	op 1 && op 2
false	false	false
false	true	false
true	false	false
true	true	true

```
a = 10
b = 20
a > 10 && b > 20
a <= 10 && b > 20
a < 10 && b <= 20
a <= 10 && b <= 20
```