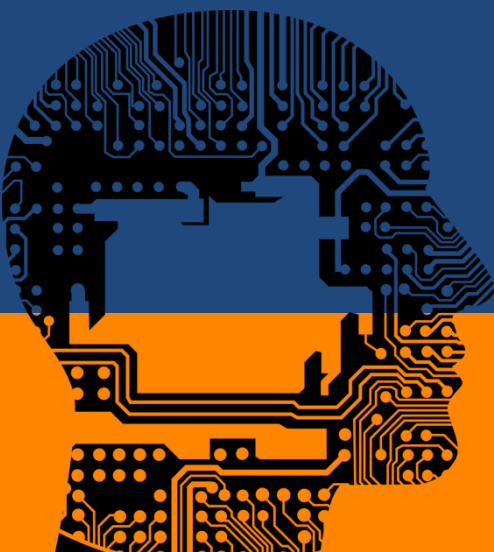


JAVA 웹 개발자 양성과정 Javascript

4강 - 제어문

By SoonGu Hong



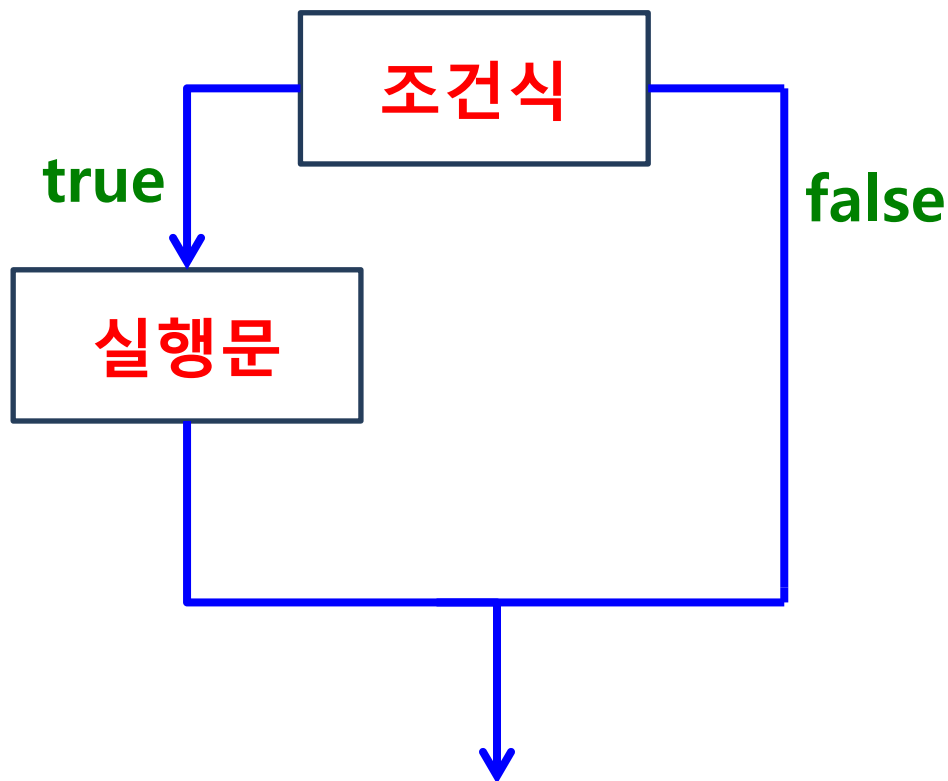
JAVA 웹 개발자 양성과정

Javascript

1. 조건문



* 조건문 if



- 조건문은 프로그램에서 조건식의 참, 거짓에 따라 코드를 다르게 실행하게 하는 **분기점을 만드는 제어문**입니다.

- if는 조건식의 논리결과가 **참일 경우** 블록 내의 코드를 **실행**하며 거짓일 경우 코드를 실행하지 않습니다.

- if블록 내부의 코드가 **단 한 줄일 경우** 블록(중괄호)을 생략할 수 있습니다.

* if문 사용법

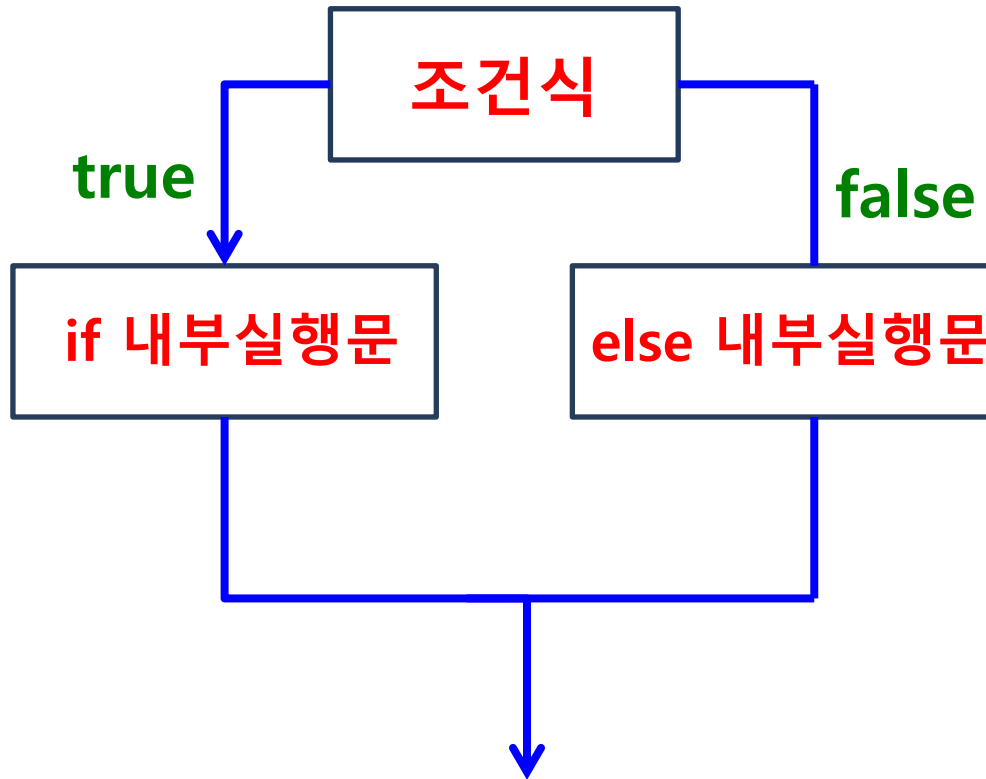
```
var point = 70;

console.log(`점수: ${point}`);

if (point >= 60) {
    console.log('60점 이상입니다. ');
    console.log('합격했어요! ');
}
```

1. 키워드 if를 사용하여 ()안에 논리형 값이 도출되는 조건식이나 함수를 넣어줍니다.
2. 블록을 만들어 준 뒤 블록 내부에 조건이 참일 경우 실행할 코드를 적습니다. (단 한문장일 경우 블록 생략 가능)

* 조건문 if ~ else



- else 키워드는 if문과 반드시 **함께 사용**해야 하며 단독으로 사용할 수 없습니다.

- if가 가지고 있는 **조건식**의 결과가 **거짓**일 경우 자동으로 **else**가 가진 코드가 **실행**되는 구조입니다.

- else 블록에도 실행문이 단 한 문장이라면 블록을 생략할 수 있습니다.

* if ~ else문 사용법

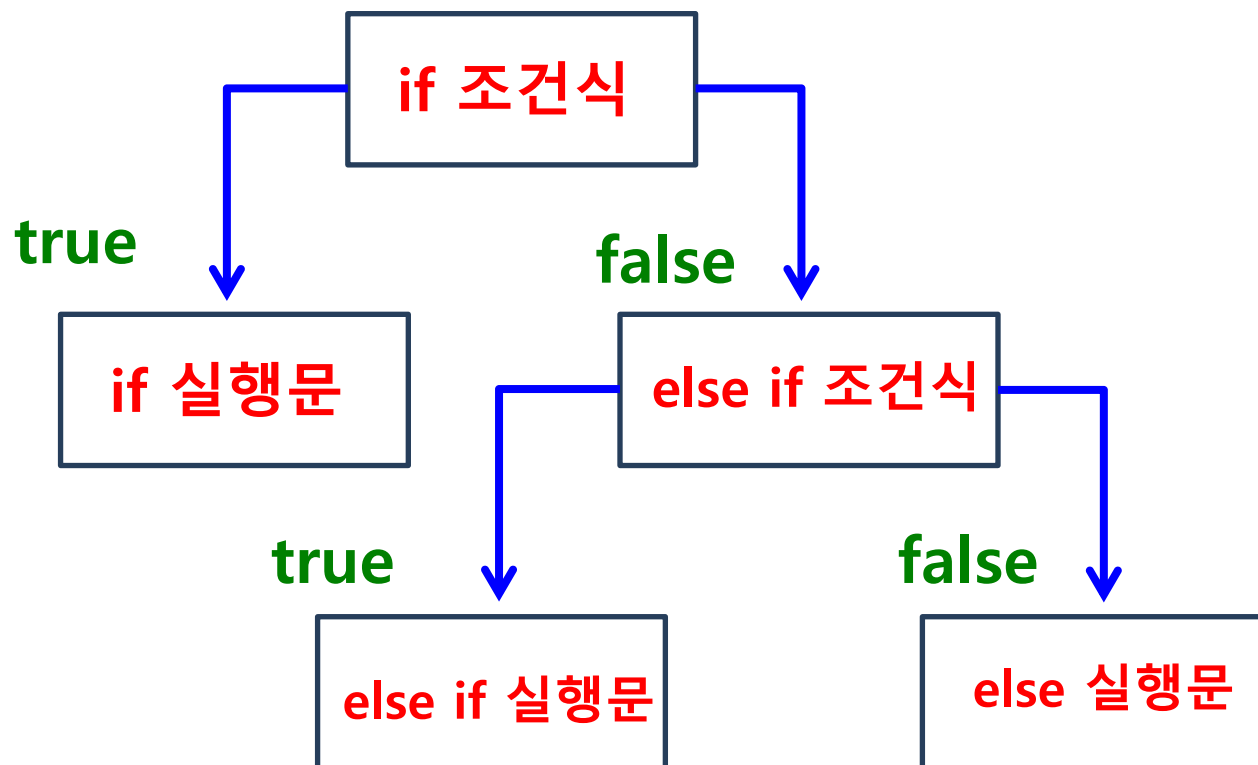
```
var point = 70;

console.log(`점수: ${point}`);

if (point >= 60) {
  console.log('60점 이상입니다. ');
  console.log('합격했어요! ');
} else {
  console.log('60점 미만입니다. ');
  console.log('불합격했어 ㅌㅌ ');
}
```

1. 기본적인 if문을 만들고 추가로 else키워드를 사용하여 새로운 블록을 만듭니다.
2. else블록 내부에 조건식이 거짓일 경우 실행할 코드를 적습니다.

* 다중분기 조건문 if ~ else if



- if ~ else if문은 좀 더 많은 분기를 만들고 싶을 때 사용하는 조건문입니다.
- 다만 위에서부터 조건을 검색하면서 내려오기 때문에 범위조건일 경우 상위 조건이 하위조건을 포괄적으로 포함하지 않도록 주의해야 합니다.

* if ~ else if문 사용법

```
var age = 14;

if (age >= 20) {
    console.log('성인입니다.');
```



```
} else if (age >= 17) {
    console.log('고딩입니다.');
```



```
} else if (age >= 14) {
    console.log('중딩입니다.');
```



```
} else if (age >= 8) {
    console.log('초딩입니다.');
```



```
} else {
    console.log('아동입니다.');
```



```
}
```

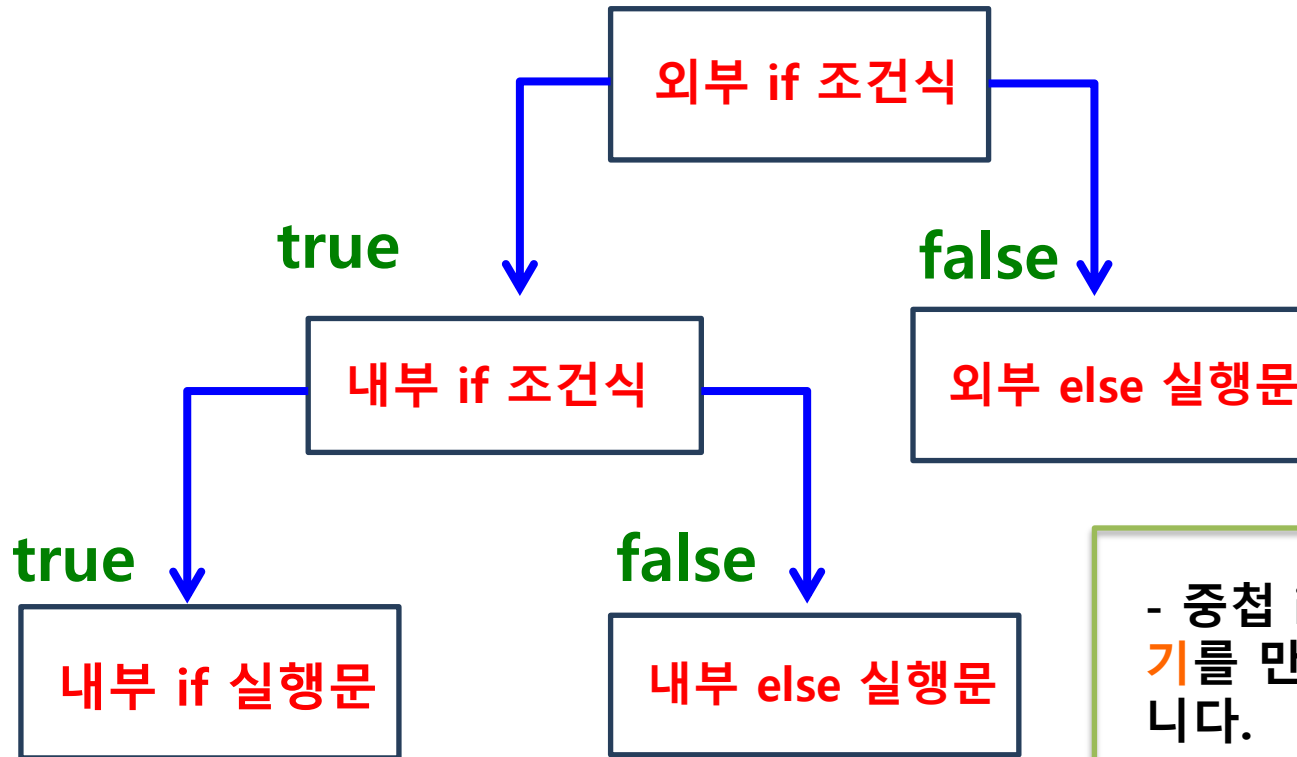
1. 기본적인 if문을 만들고 추가로 else if키워드를 사용하여 새로운 블록을 만듭니다.

2. else if블록 내부에 추가 조건식을 설정합니다.

3. else if블록을 여러 개 만들 수 있습니다.

4. 모든 조건이 거짓일 경우 실행할 else문은 선택적으로 만듭니다.

* 중첩 if문



- 중첩 if문은 단계적 조건 분기를 만들고 싶을 때 사용합니다.

- 외부 1차 if의 조건식을 만족하면 2차적으로 내부 if의 조건을 판단하고 참, 거짓에 따라 실행 코드를 결정합니다.

* 중첩 if문 사용법

```
var height = +prompt('당신의 키는??');  
  
if (height >= 140) {  
    var age = +prompt('당신의 나이는??');  
    if (age >= 8) {  
        alert('놀이기구 타러가세요~');  
    } else {  
        alert('나이 때문에 못타요~');  
    }  
} else {  
    alert('키 때문에 못타요~');  
}  
alert('오늘 하루 즐거운 시간되세요!');
```

1. 기본적인 if문을 만들고 해당 if블록 내부에 새로운 if블록을 생성합니다.

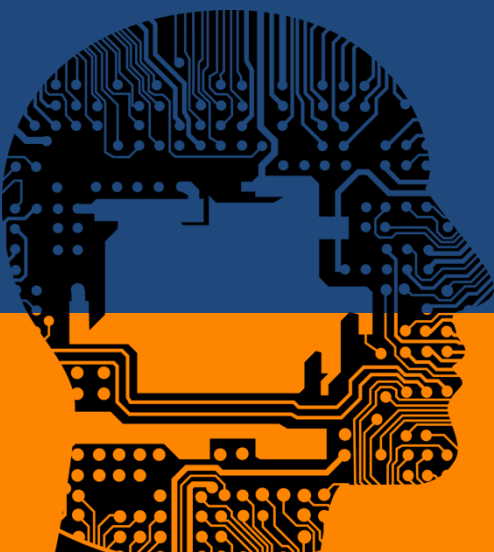
2. 전 단계에서 배운 다중 분기 if ~ else if문도 사용가능합니다.

3. 중첩의 단계가 많아질수록 블록에 주의해서 프로그래밍해야 합니다.

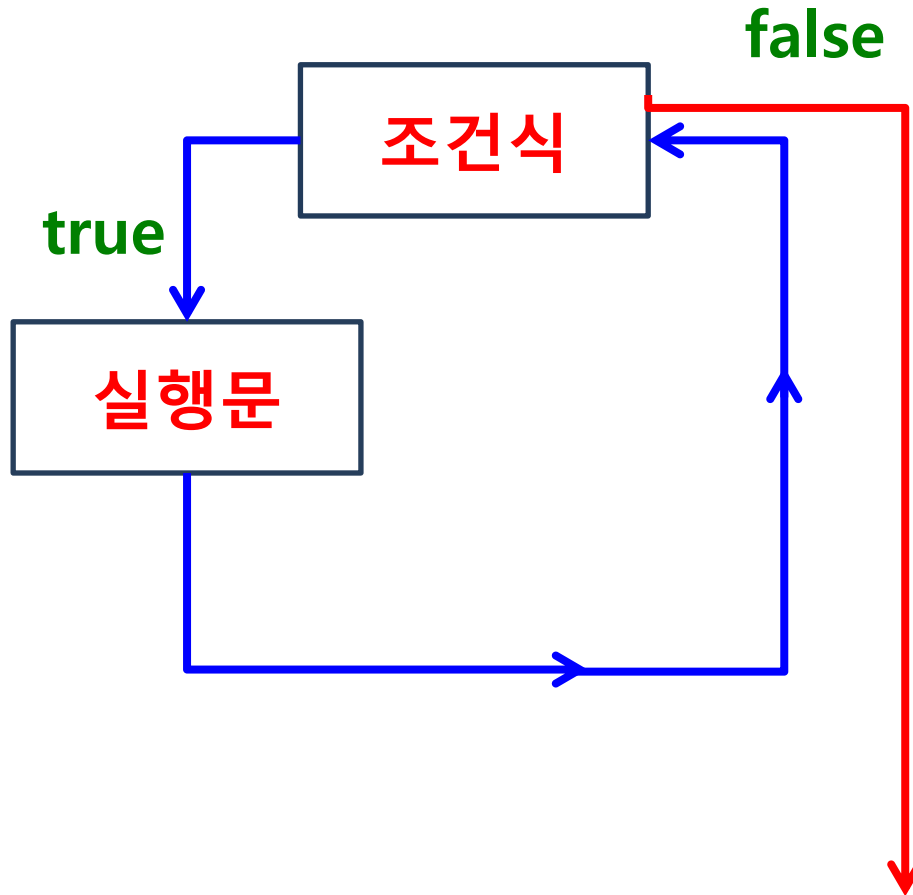
JAVA 웹 개발자 양성과정

Javascript

2. 반복문



* 반복문 while



- while문은 조건식을 검사하여 참일 경우 블록 내부의 코드를 실행하며 실행이 끝날 때마다 반복적으로 조건식을 검사하여 false가 나올 때까지 반복합니다.

- while문도 if와 마찬가지로 논리값이 도출되는 조건식이나 함수를 사용합니다.

- while문은 반복 횟수를 제어하기 위한 증감식이나 탈출문이 필요합니다.

* while문 사용법

```
var total = 0; //총합을 저장할 변수
var n = 1; //1. 제어변수의 선언(begin)

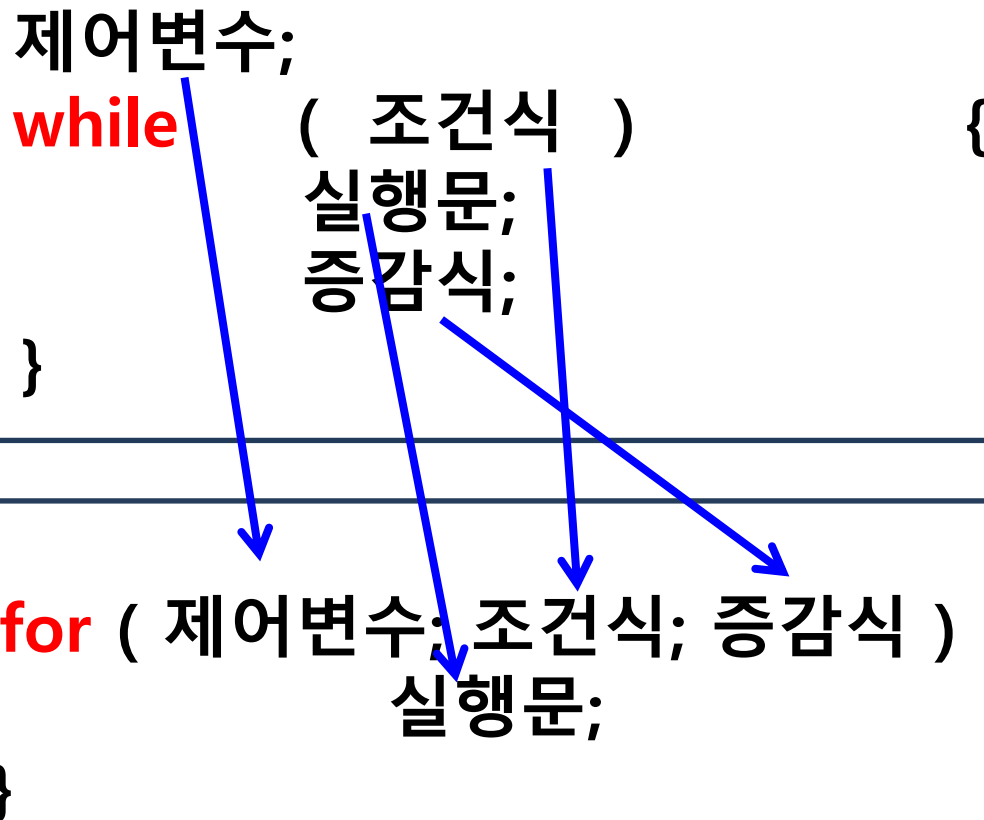
while (n <= 10) { //2. 논리형 조건식: 반복문의 종료조건(end)
    total += n; //반복실행할 코드
    n++; //3. 제어변수의 증감식(step)
}
```

1. 반복문의 **시작점**이 되는 값을 저장하는 **제어변수**를 선언합니다.
2. while문의 조건식 자리에 반복문이 **끝나는 시점**을 **조건식**이나 **함수**로 표현합니다.
3. 반복실행할 코드를 적고 1번에서 만든 **제어변수의 증감식**을 적어 반복문이 언젠가 false가 되어 종료될 수 있게 합니다.

< while문과 for문 비교 >

```
제어변수;  
while ( 조건식 )  
{  
    실행문;  
    증감식;  
}
```

```
for ( 제어변수; 조건식; 증감식 ) {  
    실행문;  
}
```



- for문은 **반복제어조건을 한꺼번에 지정**한다는 점이 다른 반복문과는 다릅니다.

- 따라서 **정확한 반복 횟수를 알고 있을 때는** for문이 while문보다 효율적입니다.

* for문 사용법

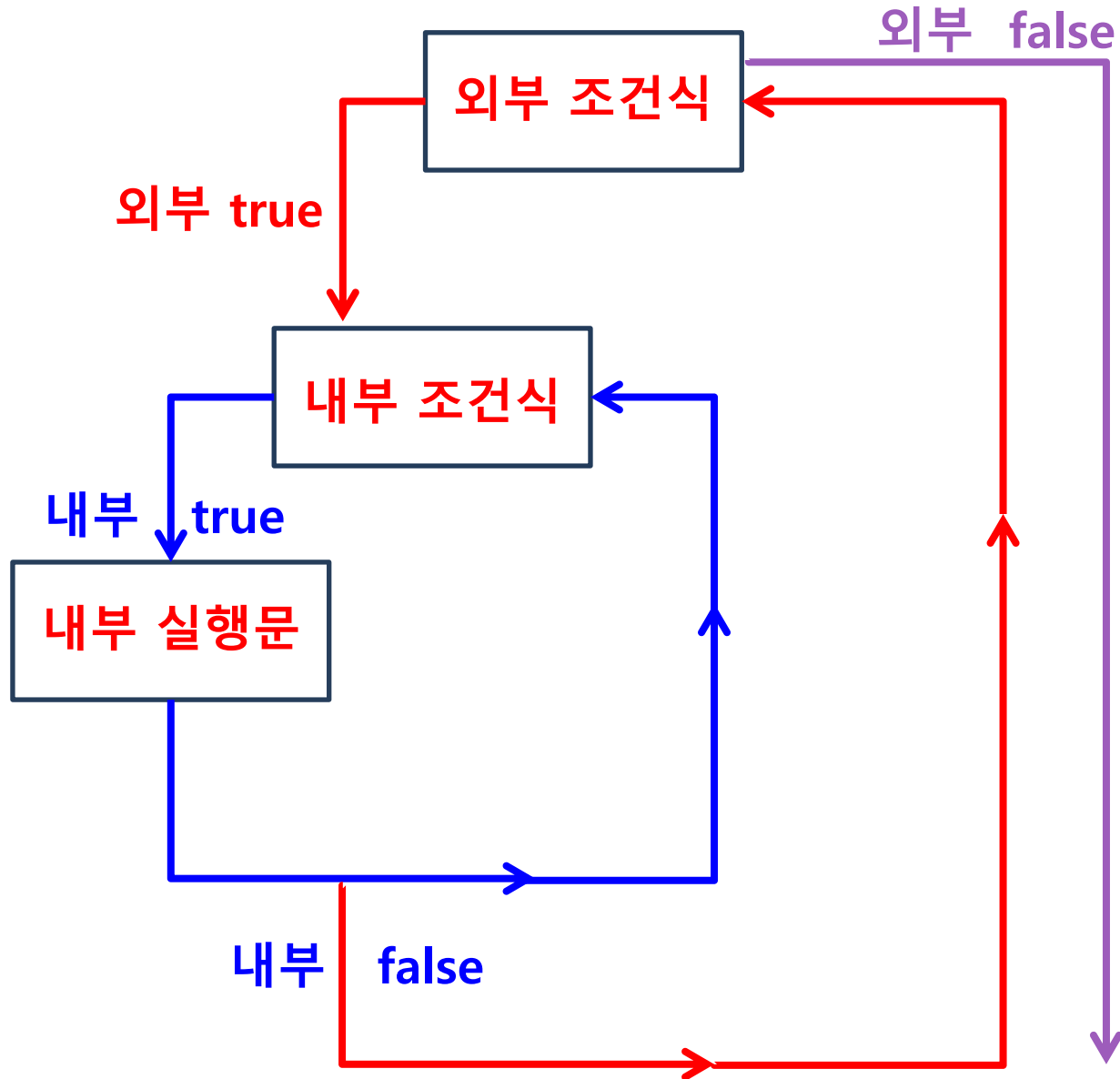
```
var total = 0;  
  
for (var ①n = 1; n ②<= 10; n③++) {  
    total += n; ④  
}
```

1. for 키워드와 함께 시작 값을 저장하는 제어변수 선언, 끝 지점을 체크할 조건식, 제어변수를 조작할 증감문을 소괄호 안에 **순서대로 세미콜론과 함께 배치**합니다.

2. 블록을 열고 반복 실행할 문장들을 적습니다.

3. 실행순서는 ①제어변수 선언 -> ②조건식 판단 -> ④실행문 -> ③증감식 -> ② 조건식 판단 -> ④실행문 -> ③증감식 순서로 진행되므로 **순서에 주의**하세요!

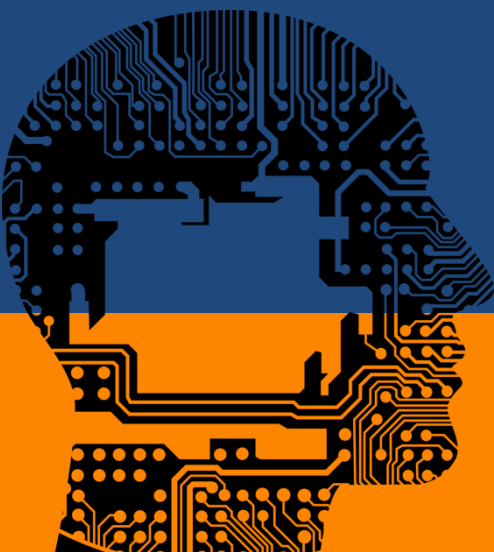
* 중첩 반복문



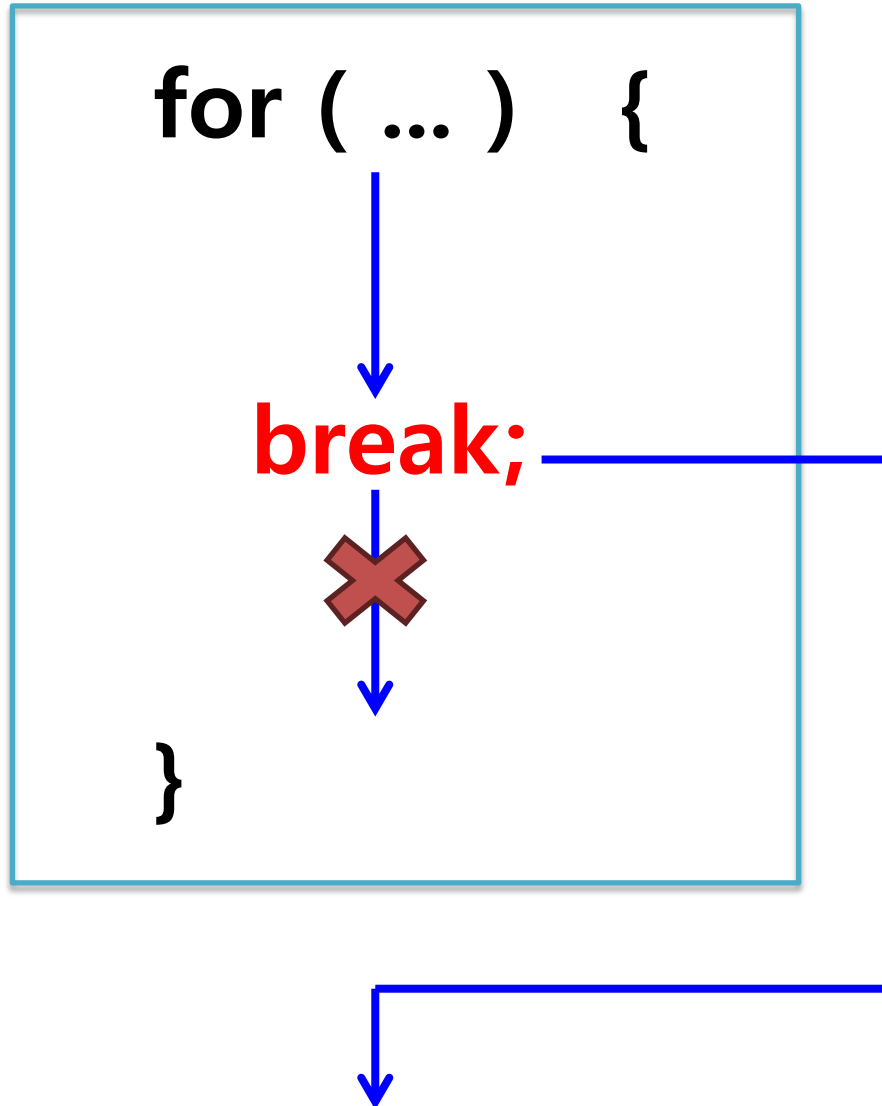
JAVA 웹 개발자 양성과정

Javascript

3. 탈출문



* 탈출문 break



- break문은 for문이나 while문에서 사용되며 반복문이 실행 중 break를 만나는 순간 **반복문을 해당 시점에서 종료**합니다.

- break를 만나는 순간 이하의 **반복문의 남은 코드는 모두 실행되지 않으며** 블록을 탈출합니다.

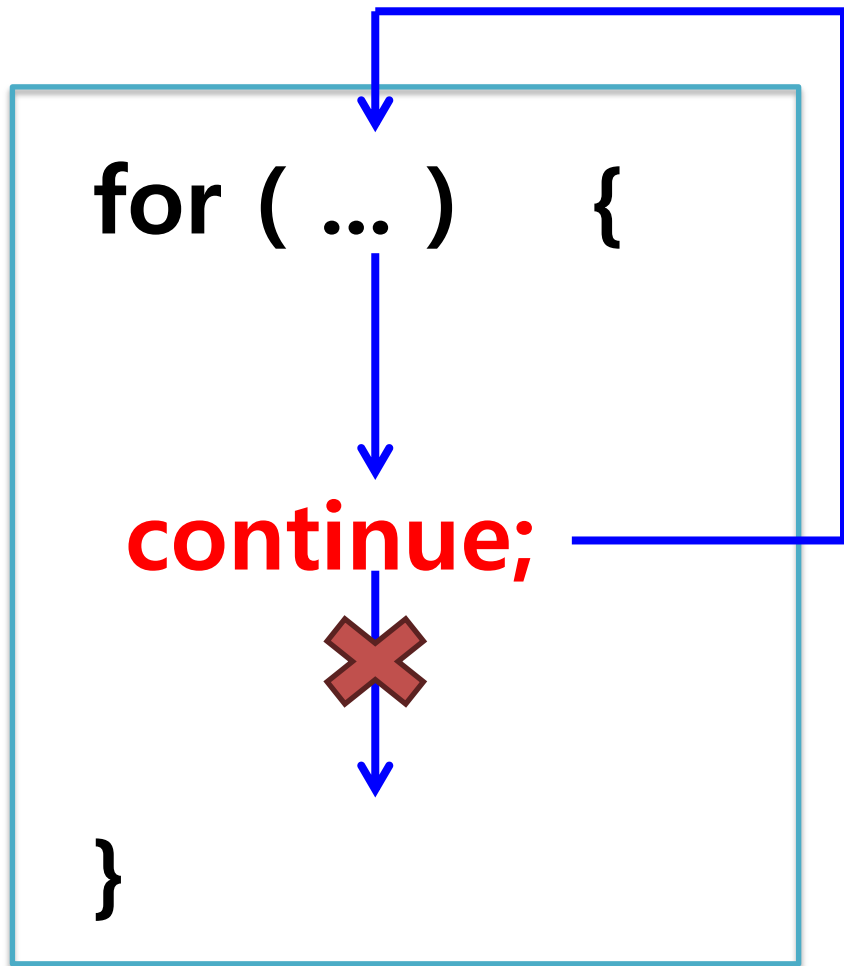
- 대개 if문과 함께 사용되며 조건에 따라 반복문을 종료할 때 사용합니다.

* 무한 루프

```
while ( true ) {  
  
    if( ... ) {  
  
        break;  
  
    }  
  
}
```

- 무한 루프는 반복문의 **반복 횟수**를 개발자가 **사전에** 정확하게 **인지하지 못하는 상황**에서 사용하며 특정 조건 하에서 반복문을 강제로 종료하는 형태로 구성합니다.
- 프로그램이 **종단되지 않게 유지**할 때도 무한루프를 사용합니다.
- 무한 루프는 일반적으로 while문을 사용하며 while의 조건식 자리에 논리값 true를 적으면 무한 루프를 구성할 수 있습니다.

* 탈출문 continue



- **continue**문은 `for`문이나 `while`문에서 사용되며 반복문이 실행 중 `continue`를 만나는 순간 **for문의 경우 증감식, while문의 경우 조건식**으로 이동합니다.

- `continue`는 `break`와 달리 반복문을 종료하지 않고 계속 수행합니다.

- 조건부로 특정 반복회차를 건너뛸 때 사용합니다.

감사합니다
THANK YOU