


# 혼자 공부하는 자바스크립트

Chapter 03 조건문  
Chapter 04 반복문





# Contents

- CHAPTER 03: 조건문

- SECTION 3-1 if 조건문

- SECTION 3-2 switch 조건문과 짧은 조건문

- CHAPTER 04: 반복문

- SECTION 4-1 배열

- SECTION 4-2 반복문



# CHAPTER 03 조건문

프로그램의 흐름을 변화시키는 요소. 조건문의 종류를 알아보고 사용 방법을 이해

## SECTION 3-1 if 조건문(1)

- if 조건문

- 불 표현식의 값이 true면 중괄호 안의 문장을 실행하고 false면 문장을 무시

---

```
if(불 값이 나오는 표현식){  
    불 값이 참일 때 실행할 문장  
}
```

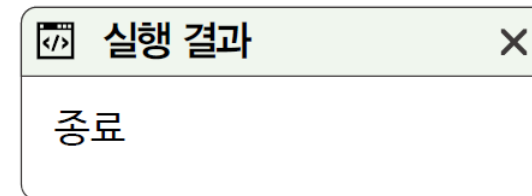
---

- if 조건문 사용하기

---

```
01 <script>  
02 // if 조건문  
03 if (273 < 100) {  
04     // 표현식 273 < 100이 참일 때 실행합니다.  
05     alert('273 < 100 => true')  
06 }  
07  
08 // 프로그램 종료  
09 alert('종료')  
10 </script>
```

---



## SECTION 3-1 if 조건문(2)

- if 조건문
  - 현재 시각에 따라 오전과 오후를 구분하는 프로그램
    - 현재 시각 구하기 (Chapter 7에서 학습)

---

```
> const date = new Date()  
undefined  
> date.getFullYear()  
2020  
> date.getMonth() + 1  
6  
> date.getDate()  
4  
> date.getHours()  
15  
> date.getMinutes()  
5  
> date.getSeconds()  
7
```

---

## SECTION 3-1 if 조건문(3)

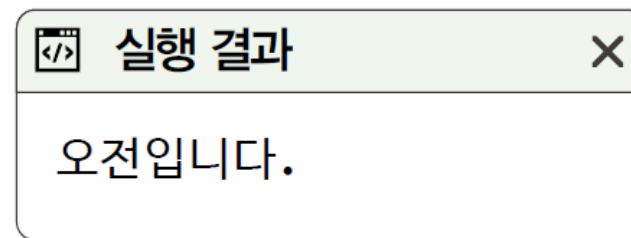
### ◦ if 조건문

#### ▪ 현재 시각에 따라 오전과 오후를 구분하는 프로그램

##### • 오전과 오후 구분하기 (소스 코드 3-1-2.html)

```
01 <script>
02 // 변수를 선언
03 const date = new Date()
04 const hour = date.getHours()
05
06 // if 조건문
07 if (hour < 12) {
08     // 표현식 hour < 12가 참일 때 실행
09     alert('오전입니다.');
10 }
11
12 if (hour >= 12) {
13     // 표현식 hour >= 12가 참일 때 실행
14     alert('오후입니다.')
15 }
16 </script>
```

현재 날짜와 시간을 갖는 객체 생성  
현재시간을 0~23 사이의 값으로 출력하는 메소드



## SECTION 3-1 if 조건문(4)

- if else 조건문

- 서로 반대되는 상황을 표현하는 구문

---

```
if(불 값이 나오는 표현식){  
    불 값이 참일 때 실행할 문장  
} else {  
    불 값이 거짓일 때 실행할 문장  
}
```

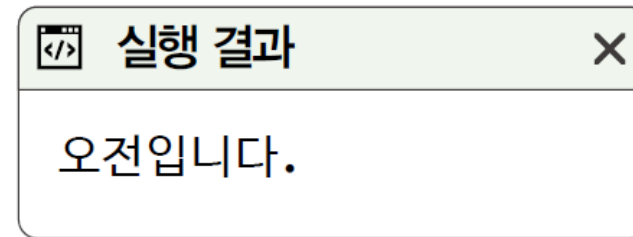
---

- if else 조건문을 사용해 현재 시간 구하기

## SECTION 3-1 if 조건문(5)

- if else 조건문
  - if else 조건문을 사용해 현재 시간 구하기

```
01 <script>
02  // 변수를 선언
03  const date = new Date()
04  const hour = date.getHours()
05
06  // if 조건문
07  if (hour < 12) {
08    // 표현식 hour < 12가 참일 때 실행
09    alert('오전입니다.')
10  } else {
11    // 표현식 hour < 12가 거짓일 때 실행
12    alert('오후입니다.')
13  }
14 </script>
```





## SECTION 3-1 if 조건문(6)

- 중첩 조건문

- 조건문 안에 조건문을 중첩해 사용

---

```
if (불 값이 나오는 표현식 1) {  
    if (불 값이 나오는 표현식 2) {  
        표현식 2가 참일 때 실행할 문장  
    } else {  
        표현식 2가 거짓일 때 실행할 문장  
    }  
} else {  
    if (불 값이 나오는 표현식 3) {  
        표현식 3이 참일 때 실행할 문장  
    } else {  
        표현식 3이 거짓일 때 실행할 문장  
    }  
}  
}
```

---

표현식 1이 참이면 실행

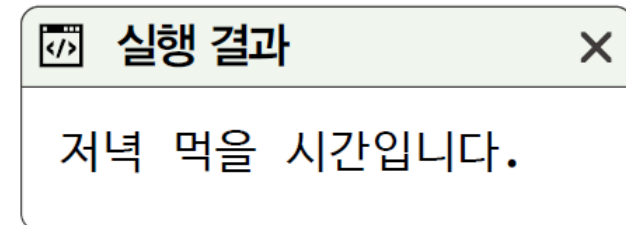
표현식 1이 거짓이면 실행

## SECTION 3-1 if 조건문(7)

- 중첩 조건문

- 중첩 조건문으로 시간 파악하기

```
01 <script>
02  // 변수를 선언
03  const date = new Date()
04  const hour = date.getHours()
05
06  // 중첩 조건문
07  if (hour < 11) {
08    // 표현식 hour < 11이 참일 때 실행
09    alert( ' 아침 먹을 시간입니다. ' )
10  } else {
11    // 표현식 hour < 11이 거짓일 때 실행
12    if (hour < 15) {
13      // 표현식 hour < 15가 참일 때 실행
14      alert('점심 먹을 시간입니다.')
15    } else {
16      // 표현식 hour < 15가 거짓일 때 실행
17      alert('저녁 먹을 시간입니다.')
18    }
19  }
20 </script>
```



## SECTION 3-1 if 조건문(8)

- if else if 조건문

- 중첩 조건문에서 중괄호를 생략한 형태

---

```
if (불 표현식) {  
    문장  
} else if (불 표현식) {  
    문장  
} else if (불 표현식) {  
    문장  
} else {  
    문장  
}  
}
```

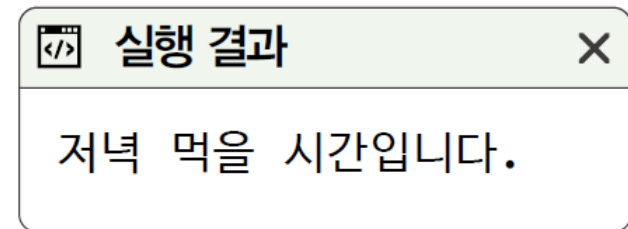
---

- 중첩 조건문으로 만들었던 예제를 if else if 조건문의 형태로 바꾸는 것은 매우 간단하여, 한 쌍의 중괄호를 지우면 됨

## SECTION 3-1 if 조건문(9)

- if else if 조건문
  - if else if 조건문으로 시간 파악하기

```
01 <script>
02  // 변수를 선언
03  const date = new Date()
04  const hour = date.getHours()
05
06  // if else if 조건문
07  if (hour < 11) {
08    // 표현식 hour < 11이 참일 때 실행
09    alert('아침 먹을 시간입니다.')
10  } else if (hour < 15) {
11    // 표현식 hour < 11이 거짓이고 표현식 hour < 15가 참일 때 실행
12    alert('점심 먹을 시간입니다.')
13  } else {
14    // 표현식 hour < 15가 거짓일 때 실행
15    alert('저녁 먹을 시간입니다.')
16  }
17 </script>
```



## [요점 정리]

- 4가지 키워드로 정리하는 핵심 포인트
  - if 조건문은 조건에 따라 코드를 실행하거나 실행하지 않도록 하기 위해 사용하는 구문
  - else 구문은 if 조건문 뒤에 사용하며, if 조건문이 거짓일 때 사용
  - 중첩 조건문은 조건문을 중첩해서 사용하는 경우를 의미
  - if ~ else if 조건문은 중첩 조건문에서 중괄호를 생략한 형태로, 겹치지 않는 3가지 이상의 조건으로 나눌 때 사용

## SECTION 3-2 switch 조건문과 짧은 조건문(1)

- switch 조건문
  - switch 조건문의 기본 형태. default 키워드는 생략 가능

---

```
switch (자료) {  
    case 조건 A:  
        break  
    case 조건 B:  
        break  
    default: → 생략할 수 있음  
        break  
}
```

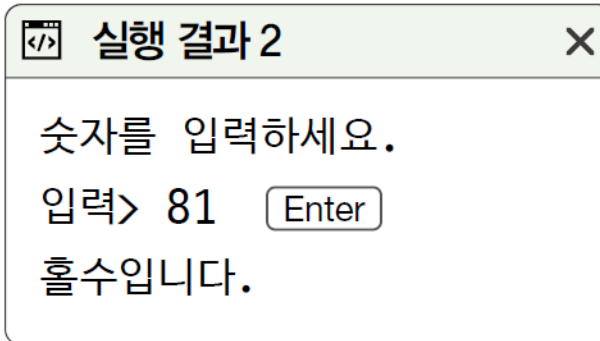
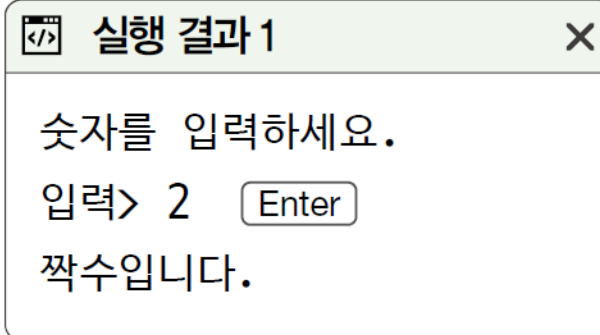
---

## SECTION 3-2 switch 조건문과 짧은 조건문(2)

- switch 조건문

- switch 조건문 사용하기

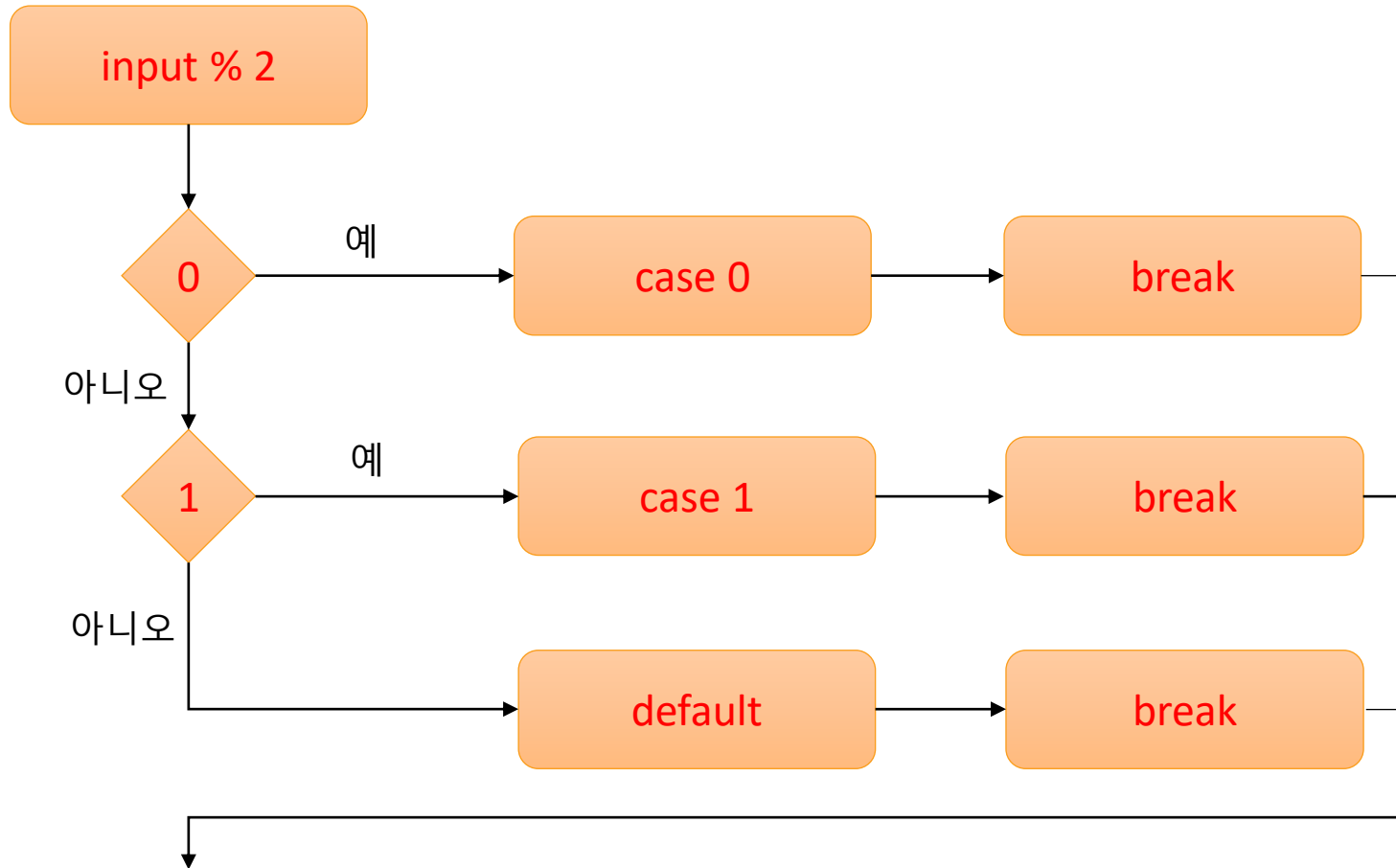
```
01 <script>
02 // 변수를 선언
03 const input = Number(prompt('숫자를 입력하세요.', '숫자'))
04
05 // 조건문
06 switch (input % 2) { → 나머지 연산자를 사용하여 홀수와 짝수를 구분
07   case 0:
08     alert('짝수입니다.')
09     break
10   case 1:
11     alert('홀수입니다.')
12     break
13   default:
14     alert('숫자가 아닙니다.')
15     break
16 }
17 </script>
```



## SECTION 3-2 switch 조건문과 짧은 조건문(3)

- switch 조건문

- break: switch 조건문이나 반복문을 빠져나가기 위해 사용하는 키워드
- switch 조건문의 괄호 안에는 비교할 값을 입력

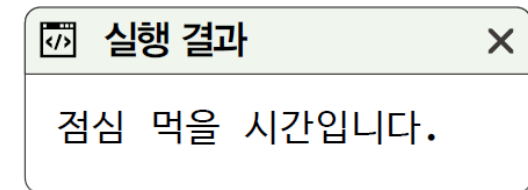




## SECTION 3-2 switch 조건문과 짧은 조건문(4)

- switch 조건문
  - If 조건문을 switch 조건문으로 변환하기

```
01 <script>
02  // 변수를 선언
03  const date = new Date()
04  const hour = date.getHours()
05
06  // 조건문
07  switch (true) {
08    case hour < 11:
09      // 표현식 hour < 11이 참일 때 실행
10      alert( ' 아침 먹을 시간입니다. ' )
11      break
12    case hour < 15:
13      // 표현식 hour < 11이 거짓이고 표현식 hour < 15가 참일 때 실행
14      alert('점심 먹을 시간입니다.')
15      break
16    default:
17      // 위의 모든 것이 거짓일 때 실행
18      alert('저녁 먹을 시간입니다.')
19      break
20  }
21 </script>
```



## SECTION 3-2 switch 조건문과 짧은 조건문(5)

### ◦ 조건부 연산자

#### ▪ 기본 형태

불 표현식 ? 참일 때의 결과 : 거짓일 때의 결과

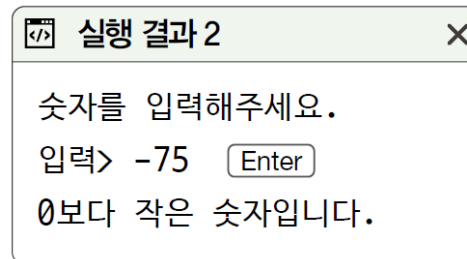
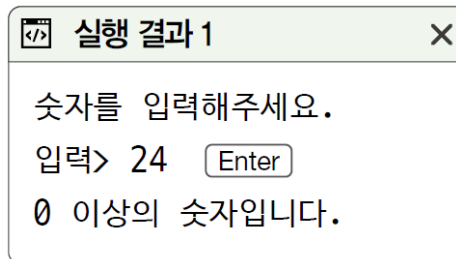
- 자바스크립트에서 항을 3개 갖는 연산자는 조건부 연산자가 유일해서 **삼항 연산자**라고 부르기도 함

#### ▪ 조건부 연산자 사용하기

```
01 <script>
02 // 변수를 선언
03 const input = prompt('숫자를 입력해주세요.', '')
04 const number = Number(input)
05
06 // 조건문
07 const result = (number >= 0) ? 0 이상의 숫자입니다. : 0보다 작은 숫자입니다.
08 alert(result)
09 </script>
```

(number >= 0)이 true면 이 값이 할당

(number >= 0)이 false면 이 값이 할당



## SECTION 3-2 switch 조건문과 짧은 조건문(6)

### ◦ 짧은 조건문

- 짧은 조건문은 논리 연산자의 특성을 조건문으로 사용

- 논리합 연산자를 사용한 짧은 조건문

---

불 표현식 || 불 표현식이 거짓일 때 실행할 문장

---

- 논리곱 연산자를 사용한 짧은 조건문

---

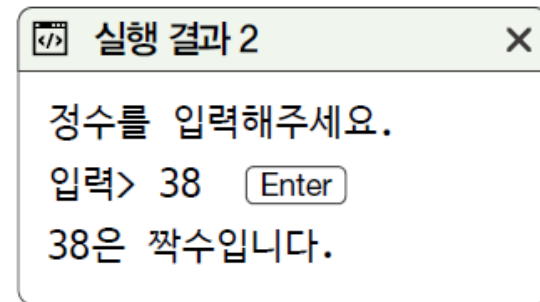
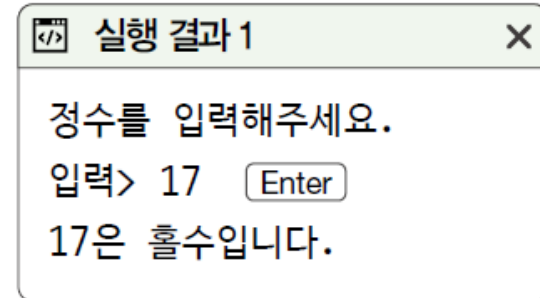
결과가 거짓인 불 표현식 && 불 표현식이 참일 때 실행할 문장

---

## SECTION 3-2 switch 조건문과 짧은 조건문(7)

- 짝수와 홀수 구분하기(누적 예제)
  - if else 조건문으로 짝수와 홀수 구분하기(1)

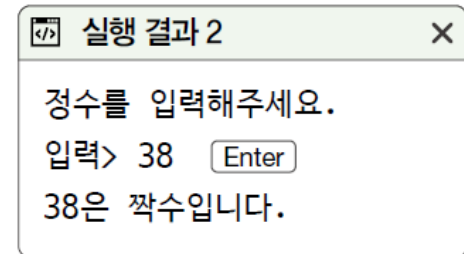
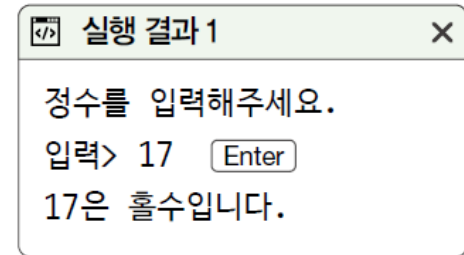
```
01 <script>
02 // 입력이 문자열이므로 다음과 같은 코드를 사용할 수 있음
03 const input = prompt('정수를 입력해주세요.');"
04 const last = input[input.length - 1]
05
06 // 끝자리를 비교
07 if (last === '0' ||
08     last === '2' ||
09     last === '4' ||
10     last === '6' ||
11     last === '8') {
12   alert(`${input}은 짝수입니다.`)
13 } else {
14   alert(`${input}은 홀수입니다.`)
15 }
16 </script>
```



## SECTION 3-2 switch 조건문과 짧은 조건문(8)

- 짝수와 홀수 구분하기(누적 예제)
  - if else 조건문으로 짝수와 홀수 구분하기(2)

```
01 <script>
02  const input = prompt('정수를 입력해주세요,')
03  const number = Number(input)
04
05  if (number % 2 === 0) {
06    alert(`${input}은 짝수입니다.`)
07  } else {
08    alert(`${input}은 홀수입니다.`)
09  }
10 </script>
```



## SECTION 3-2 switch 조건문과 짧은 조건문(9)

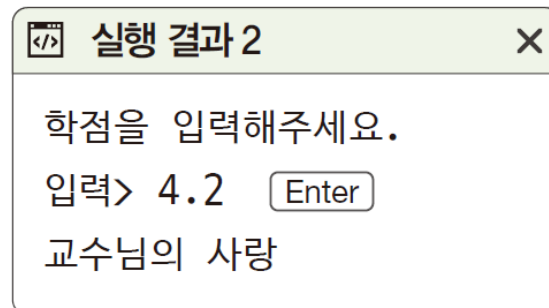
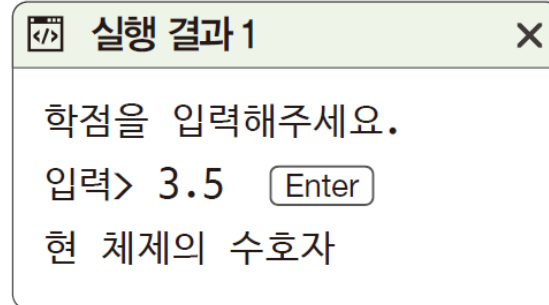
- 학점을 기반으로 별명 붙여주기(누적 예제)
  - 인터넷에서 학점을 학생들이 재미있게 표현한 유머를 이를 조건문으로 구현하고 출력

조건	설명(학생 평가)
4.5	신
4.2~4.5	교수님의 사랑
3.5~4.2	현 체제의 수호자
2.8~3.5	일반인
2.3~2.8	일탈을 꿈꾸는 소시민
1.75~2.3	오락문화의 선구자
1.0~1.75	불가촉천민
0.5~1.0	자벌레
0~0.5	플랑크톤
0	시대를 앞서가는 혁명의 씨앗

## SECTION 3-2 switch 조건문과 짧은 조건문(10)

- 학점을 기반으로 별명 붙여주기(누적 예제)
  - 중첩 조건문 사용하기(1)

```
01 <script>
02  const score = Number(prompt('학점을 입력해주세요.', '학점'))
03  if (score === 4.5) {
04    alert('신')
05  } else if (4.2 <= score && score < 4.5) {
06    alert('교수님의 사랑')
07  } else if (3.5 <= score && score < 4.2) {
08    alert('현 체제의 수호자')
09  } else if (2.8 <= score && score < 3.5) {
10    alert('일반인')
11  } else if (2.3 <= score && score < 2.8) {
12    alert('일탈을 꿈꾸는 소시민')
13  } else if (1.75 <= score && score < 2.3) {
14    alert('오락문화의 선구자')
15  } else if (1.0 <= score && score < 1.75) {
16    alert('불가촉천민')
17  } else if (0.5 <= score && score < 1.0) {
18    alert('자벌레')
19  } else if (0 < score && score < 0.5) {
20    alert('플랑크톤')
21  } else {
22    alert('시대를 앞서가는 혁명의 씨앗')
23  }
24 </script>
```



## SECTION 3-2 switch 조건문과 짧은 조건문(11)

- 학점을 기반으로 별명 붙여주기(누적 예제)
  - 중첩 조건문 사용하기(2)

```
01 <script>
02  const score = Number(prompt('학점을 입력해주세요.', '학점'))
03  if (score === 4.5) {
04    alert('신')
05  } else if (4.2 <= score) {
06    alert('교수님의 사랑')
07  } else if (3.5 <= score) {
08    alert('현 체제의 수호자')
09  } else if (2.8 <= score) {
10    alert('일반인')
11  } else if (2.3 <= score) {
12    alert('일탈을 꿈꾸는 소시민')
13  } else if (1.75 <= score) {
14    alert('오락문화의 선구자')
15  } else if (1.0 <= score) {
16    alert('불가촉천민')
17  } else if (0.5 <= score) {
18    alert('자벌레')
19  } else if (0 < score) {
20    alert('플랑크톤')
21  } else {
22    alert('시대를 앞서가는 혁명의 씨앗')
23  }
24 </script>
```

- if 조건문은 위에서 아래로 흐르고 else 구문은 이전의 조건이 맞지 않을 때 넘어오는 부분. 따라서 앞에서 이미 제외된 조건을 한 번 더 검사할 필요는 없음
- 3행에서 score가 4.5인지는 검사했으므로 이를 생략. 이렇게 조건식을 바꾸면 조건 비교를 절반만 하게 되고 코드도 훨씬 쉽게 읽을 수 있음

---

```
else if (4.2 <= score && score < 4.5)
```

---



---

```
else if (4.2 <= score)
```

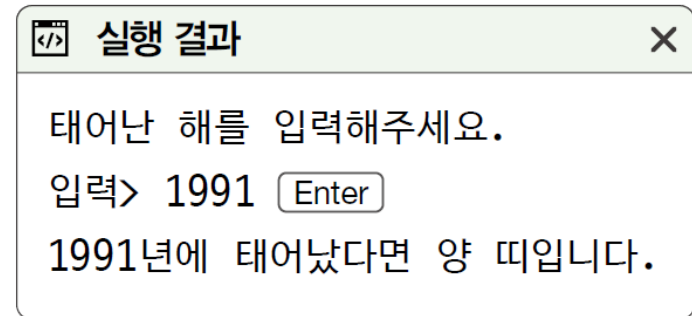
---



## SECTION 3-2 switch 조건문과 짧은 조건문(12)

- 태어난 연도를 입력받아 띠 출력하기(누적 예제)
  - if else if 조건문 사용해보기

```
01 <script>
02  const rawInput = prompt('태어난 해를 입력해주세요.', '')
03  const year = Number(rawInput)
04  const e = year % 12
05
06  let result
07  if (e === 0) { result = '원숭이' }
08  else if (e === 1) { result = '닭' }
09  else if (e === 2) { result = '개' }
10  else if (e === 3) { result = '돼지' }
11  else if (e === 4) { result = '쥐' }
12  else if (e === 5) { result = '소' }
13  else if (e === 6) { result = '호랑이' }
14  else if (e === 7) { result = '토끼' }
15  else if (e === 8) { result = '용' }
16  else if (e === 9) { result = '뱀' }
17  else if (e === 10) { result = '말' }
18  else if (e === 11) { result = '양' }
19  alert(`${year}년에 태어났다면 ${result} 띠입니다.`)
20 </script>
```



## SECTION 3-2 switch 조건문과 짧은 조건문(13)

- 태어난 연도를 입력받아 띠 출력하기(누적 예제)
  - split로 문자열을 잘라 사용하기

---

```
01 <script>
02  const rawInput = prompt('태어난 해를 입력해주세요.');"
03  const year = Number(rawInput)
04  const tti = '원숭이,닭,개,돼지,쥐,소,호랑이,토끼,용,뱀,말,양'.split(',')
05
06  alert(`${year}년에 태어났다면 ${tti[year % 12]} 띠입니다.`)
07 </script>
```

---

[노트] ' 문자열A '.split( ' 문자열B ' ) 메소드는 문자열A를 문자열B로 잘라서 배열을 만들어내는 메소드. 배열과 관련된 내용은 04장에서 학습

위의 코드 '원숭이,닭,개,돼지,쥐,소,호랑이,토끼,용,뱀,말,양'.split(',')에서는 원숭이,닭,개,돼지,쥐,소,호랑이,토끼,용,뱀,말,양을 ' , ' 로 잘랐으므로, ['원숭이', '닭', '개', '돼지', '쥐', '소', '호랑이', '토끼', '용', '뱀', '말', '양']라는 배열이 만들어짐

## [요점 정리]

- 3가지 키워드로 정리하는 핵심 포인트
  - switch 조건문은 값에 따라서 조건 분기를 걸어주는 조건문
  - 조건부 연산자(삼항 연산자)는  $A ? B : C$ 와 같은 형태로 피연산자 3개를 갖는 연산자. 조건 분기에 사용할 수 있음
  - 짧은 조건문은 논리 연산자의 특이한 성질을 사용해서 조건 분기에 활용하는 코드



# CHAPTER 04 반복문

배열의 개념과 문법을 익혀 while 반복문과 for 반복문 학습

## SECTION 4-1 배열(1)

- 배열(array): 여러 자료를 묶어서 활용할 수 있는 특수한 자료

---

```
> const str = '안녕하세요'
```

```
> str[2]
```

①

```
> str[str.length - 1]
```

②

---

str

안	녕	하	세	요
[0]	[1]	[2]	[3]	[4]

↑  
str[2]

↑  
str[str.length-1]

↓  
5

## SECTION 4-1 배열(2)

### 배열 만들기

---

[요소, 요소, 요소, ... ,요소]

---

---

```
> const array = [273, 'String', true, function () { }, {}, [273, 103]]
```

```
undefined
```

```
> array Enter
```

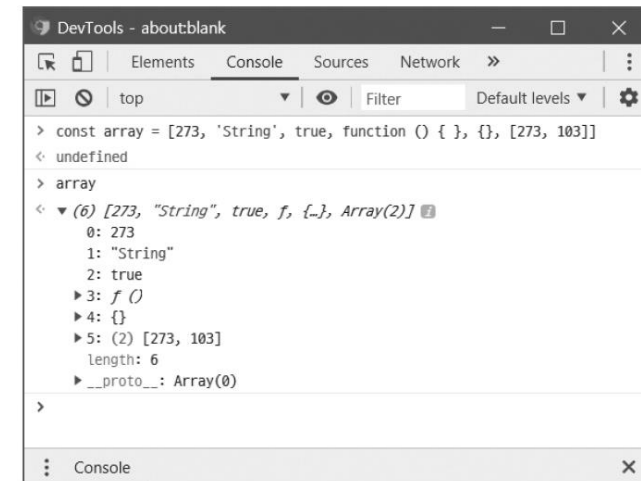
```
(6) [273, 'String', true, f, {...}, Array(2)]
```

---

요소 개수

요소

구글 크롬 개발자 도구의 Console에서 코드를 실행할 때 출력된 배열 결과 왼쪽에 드롭 다운 버튼 ▶ 클릭하면 0번째에 273, 1번째에 'String', 2번째에 true 등의 값을 확인 할 수 있음



## SECTION 4-1 배열(3)

- 배열 요소에 접근하기

---

배열[인덱스]

---

---

```
> const numbers = [273, 52, 103, 32]
```

```
undefined
```

```
> numbers[0]
```

```
273
```

```
> numbers[1]
```

```
52
```

```
> numbers[1 + 1]
```

```
103
```

```
> numbers[1 * 3]
```

```
32
```

---

① 배열은 여러 개의 요소를 갖기 때문에 일반적으로 배열 이름을 복수형으로(number → numbers)

② numbers[1 + 1], numbers[1 \* 3]처럼 대괄호 안에 계산식을 넣을 수도 있음

## SECTION 4-1 배열(4)

- 배열 요소 개수 확인하기

---

배열.length

---

---

> const fruits = ['배', '사과', '키위', '바나나']  
undefined

→ 배열 이름을 복수형으로 작성

> fruits.length  
4

→ 4 배열 fruits에 4개의 요소가 들어있으므로 4를 출력

> fruits[fruits.length - 1]  
'바나나'

→ fruits[4-1], 배열의 3번째 요소인 "바나나"를 출력

---



## SECTION 4-1 배열(5)

### 배열 뒷부분에 요소 추가하기

- push() 메소드를 사용해 배열 뒷부분에 요소 추가하기

배열.push(요소)

---

```
> const todos = ['우유 구매', '업무 메일 확인하기', '필라테스 수업']
```

```
undefined
```

```
> todos
```

```
(3) ['우유 구매', '업무 메일 확인하기', '필라테스 수업']
```

```
> todos.push('저녁 식사 준비하기')
```

```
4
```

```
> todos.push('피아노 연습하기')
```

```
5
```

```
> todos
```

```
(5) ['우유 구매', '업무 메일 확인하기', '필라테스 수업', '저녁 식사 준비하기', '피아노 연습하기']
```

---

push() 메소드로 요소가 추가되어 기존 요소  
개수에서 추가된 요소 개수가 출력

↓  
뒷부분에 2개의 요소가 추가

## SECTION 4-1 배열(6)

- 배열 뒷부분에 요소 추가하기

- 인덱스를 사용해 배열 뒷부분에 요소 추가하기

---

```
> const fruitsA = ['사과', '배', '바나나']
```

```
Undefined
```

```
> fruitsA[10] = '귤'
```

```
'귤'
```

```
> fruitsA
```

```
(11) ['사과', '배', '바나나', empty × 7, '귤']
```

---

—————→ 배열 10번째 인덱스에 "귤"을 추가

---

```
> const fruitsB = ['사과', '배', '바나나']
```

```
Undefined
```

```
> fruitsB[fruitsB.length] = '귤'.
```

```
'귤'
```

```
> fruitsB
```

```
(4) ['사과', '배', '바나나', '귤']
```

---

—————→ fruitsB의 요소는 3개이므로 fruitsB[3]에 "귤"을 추가

## SECTION 4-1 배열(7)

- 배열 요소 제거하기
  - 인덱스로 요소 제거하기

---

배열.splice(인덱스, 제거할 요소의 개수)

---

---

```
> const itemsA = ['사과', '배', '바나나']  
undefined
```

```
> itemsA.splice(2, 1)  
['바나나']
```

→ 배열의 2번째 인덱스로부터 1개 요소를 제거  
→ 제거된 요소가 배열로 리턴

```
> itemsA  
(2) ['사과', '배']
```

---

→ 배열의 값을 확인해보면 요소가 제거된 것을 알 수 있음

## SECTION 4-1 배열(8)

### 배열 요소 제거하기

#### 값으로 요소 제거하기

---

```
const 인덱스 = 배열.indexOf(요소)  
배열.splice(인덱스, 1)
```

---

```
> const itemsB = ['사과', '배', '바나나']
```

```
undefined
```

```
> const index = itemsB.indexOf('바나나')
```

```
Undefined
```

```
> index
```

```
2
```

→ 배열 내부에 바나나가 있으므로 해당 요소의 인덱스를 출력

```
> itemsB.splice(index, 1)
```

```
['바나나']
```

→ 배열의 2번째 인덱스에 있는 1개의 요소를 제거

```
> itemsB
```

```
(2) ['사과', '배']
```

→ 배열에서 바나나가 제거

```
> itemsB.indexOf('바나나')
```

```
-1
```

→ 바나나는 배열에 없으므로 -1을 출력

---

## SECTION 4-1 배열(8)

- 배열의 특정 위치에 요소 추가하기
  - 값으로 요소 제거하기

---

```
배열.splice(인덱스, 0, 요소)
```

---

---

```
> const itemsD = ['사과', '귤', '바나나', '오렌지'];  
Undefined
```

```
> itemsD.splice(1, 0, '양파');  
[]
```

```
> itemsD  
(5) ['사과', '양파', '귤', '바나나', '오렌지'];
```

---

→ 1번째 인덱스에 양파가 추가

## SECTION 4-2 반복문(1)



- for in 반복문

- 배열 요소를 하나하나 꺼내서 특정 문장을 실행할 때 사용

```
for (const 반복 변수 in 배열 또는 객체) {  
  문장  
}
```

- for in 반복문

```
01 <script>  
02  const todos = ['우유 구매', '업무 메일 확인하기', '필라테스 수업']  
03  
04  for (const i in todos) {  
05    console.log(`${i}번째 할 일: ${todos[i]}`)  
06  }  
07 </script>
```

 실행 결과 

0번째 할 일: 우유 구매  
1번째 할 일: 업무 메일 확인하기  
2번째 할 일: 필라테스 수업

## SECTION 4-2 반복문(2)

- for of 반복문

- 요소의 값을 반복할 때 안정적으로 사용

```
for (const 반복 변수 of 배열 또는 객체) {  
  문장  
}
```

→ for in 반복문과 다르게 반복 변수에 요소의 값이 들어감

- for of 반복문

```
01 <script>  
02  const todos = ['우유 구매', '업무 메일 확인하기', '필라테스 수업']  
03  for (const todo of todos) {  
04    console.log(`오늘의 할 일: ${todo}`)  
05  }  
06 </script>
```

### 실행 결과

오늘의 할 일: 우유 구매  
오늘의 할 일: 업무 메일 확인하기  
오늘의 할 일: 필라테스 수업

## SECTION 4-2 반복문(3)

### ◦ for 반복문

- 특정 횟수만큼 반복하고 싶을 때 사용하는 범용적인 반복문

```
for (let i = 0; i < 반복 횟수; i++) {  
  문장  
}
```

→ 다른 반복문과 다르게 반복 변수를 let 키워드로 선언합니다

- for 반복문 기본

```
01 <script> 0 5  
02 for (let i = 0; i < 5; i++) {  
03   console.log(`${i}번째 반복입니다.`)  
04 }  
05 </script>
```

→ 0부터 시작해서 5 미만이면 반복합니다. → 불 값

**실행 결과**

0번째 반복입니다.  
1번째 반복입니다.  
2번째 반복입니다.  
3번째 반복입니다.  
4번째 반복입니다.



## SECTION 4-2 반복문(4)

- for 반복문
  - 1부터 N까지 더하기

```
01 <script>
02 let output = 0
03 for (let i = 1; i <= 100; i++) {
04   output += i
05 }
06 console.log(`1~100까지 숫자를 모두 더하면 ${output}입니다.`)
07 </script>
```

1부터 100까지 반복

실행 결과

1~100까지 숫자를 모두 더하면 5050입니다.

## SECTION 4-2 반복문(5)

- for 반복문과 함께 배열 사용하기

- for 반복문과 배열

```
01 <script>
02  const todos = ['우유 구매', '업무 메일 확인하기', '필라테스 수업']
03
04  for (let i = 0; i < todos.length; i++) {
05    console.log(`${i}번째 할 일: ${todos[i]}`)
06  }
07 </script>
```

### 실행 결과

0번째 할 일: 우유 구매  
1번째 할 일: 업무 메일 확인하기  
2번째 할 일: 필라테스 수업

- for 반복문으로 배열을 반대로 출력하기

```
01 <script>
02  const todos = ['우유 구매', '업무 메일 확인하기', '필라테스 수업']
03
04  for (let i = todos.length - 1; i >= 0; i--) {
05    console.log(`${i}번째 할 일: ${todos[i]}`)
06  }
07 </script>
```

### 실행 결과

2번째 할 일: 필라테스 수업  
1번째 할 일: 업무 메일 확인하기  
0번째 할 일: 우유 구매

배열의 마지막 요소부터 0까지 하나씩 빼면서 반복

## SECTION 4-2 반복문(6)

### ◦ while 반복문

- if 조건문과 다른 점은 문장을 한 번만 실행하고 끝나는 것이 아니라 불 표현식이 true면 계속해서 문장을 실행
- 조건이 변하지 않는다면 무한히 반복 실행하므로 조건을 거짓으로 만드는 내용이 문장에 포함되어야 함.
- 무한 루프: 반복문이 무한 반복되는 것

---

```
while (불 표현식) {  
    문장  
}
```

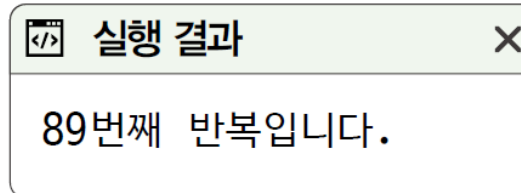
---

- 무한 반복문

---

```
01 <script>  
02  let i = 0  
03  while (true) {  
04    alert(`${i}번째 반복입니다.`)  
05    i = i + 1  
06  }  
07 </script>
```

---

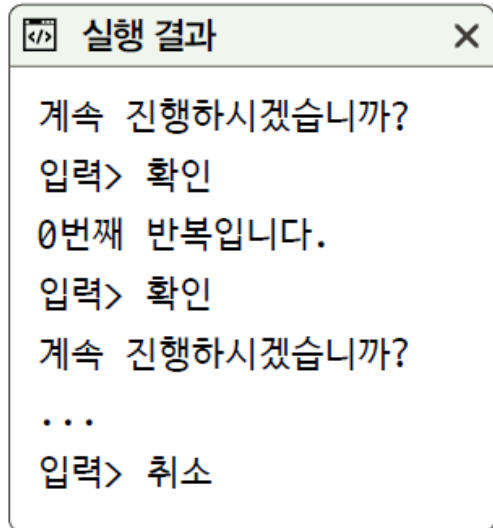


## SECTION 4-2 반복문(7)

### ◦ while 반복문

- confirm() 함수를 사용하여 사용자에게 확인을 받는 대화상자 실행.
- [확인]은 true, [취소]는 false로 입력 받아 조건이 false(거짓)일 때 반복문 종료.
- While 반복문 기본

```
01 <script>
02 let i = 0
03 while (confirm('계속 진행하시겠습니까?')) {
04     // 사용자가 [확인] 버튼을 클릭하면 true가 되어 계속 반복합니다.
05     alert(`${i}번째 반복입니다.`)
06     i = i + 1
07 }
08 </script>
```



## SECTION 4-2 반복문(8)

### ◦ while 반복문

- while 반복문과 함께 배열 사용하기
- 배열과 함께 사용하기

```
01 <script>
02 let i = 0
03 const array = [1, 2, 3, 4, 5]
04
05 while (i < array.length) {
06   console.log(`${i} : ${array[i]}`)
07   i++
08 }
09 </script>
```



```
실행 결과
0 : 1
1 : 2
2 : 3
3 : 4
4 : 5
```

※ 횟수를 기준으로 반복할 때는 코드를 간결하게 구현할 수 있는 for 반복문을 사용하는 것이 훨씬 편함.  
while 반복문은 조건에 큰 비중이 있을 때 사용하는 것이 효과적  
'특정 시간 동안 어떤 데이터를 받을 때까지', '배열에서 어떠한 요소가 완전히 제거될 때까지' 등 조건을  
기반으로 사용하는 반복문에 while 반복문을 사용

## SECTION 4-2 반복문(9)

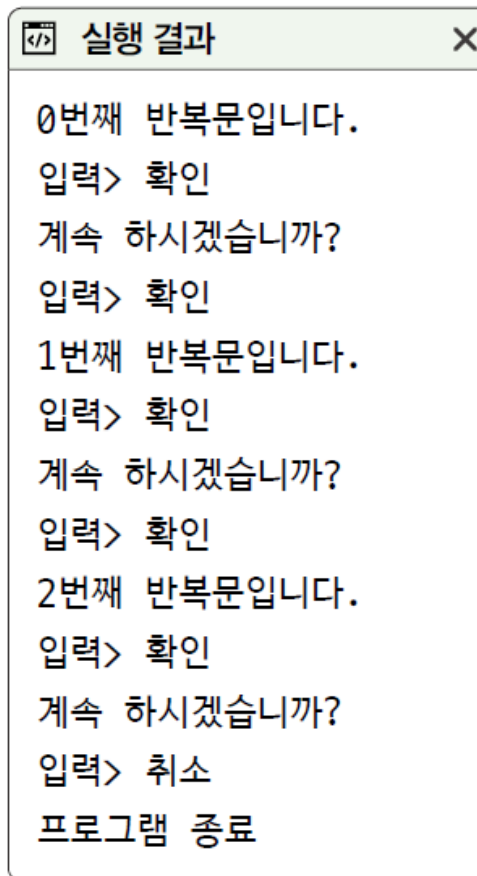
- break 키워드

- switch 조건문이나 반복문을 벗어날 때 사용하는 키워드

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

- break 키워드 활용

```
01 <script>  
02 // 반복문  
03 for (let i = 0; true; i++) {  
04     alert(i + '번째 반복문입니다.')05  
06     // 진행 여부를 물어봅니다.  
07     const isContinue = confirm('계속 하시겠습니까?')  
08     if (!isContinue) {  
09         break  
10     }  
11 }  
12  
13 // 프로그램의 종료를 확인합니다.  
14 alert('프로그램 종료')  
15 </script>
```



## SECTION 4-2 반복문(10)

- continue 키워드

- continue 키워드는 반복문 안의 반복 작업을 멈추고 반복문의 처음으로 돌아가 다음 반복 작업을 진행
- continue 키워드 활용(1)

---

```
01 <script>
02  // 반복문
03  for (let i = 0; i < 5; i++) {
04    // 현재 반복 작업을 중지하고 다음 반복 작업을 수행합니다.
05    continue;
06    alert(i);
07  }
08 </script>
```

---

## SECTION 4-2 반복문(11)

- continue 키워드
  - continue 키워드 활용(2)

```
01 <script>
02  // 변수를 선언합니다.
03  let output = 0
04
05  // 반복문
06  for (let i = 1; i <= 10; i++) {
07    // 조건문
08    if (i % 2 === 1) {
09      // 홀수면 현재 반복을 중지하고 다음 반복을 수행합니다.
10      continue;
11    }
12    output += i
13  }
14
15  // 출력합니다.
16  alert(output);
17 </script>
```

 실행 결과 ×

30



## SECTION 4-2 반복문(12)

- 중첩 반복문을 사용하는 피라미드(누적 예제)
  - 중첩 반복문은 일반적으로 n-차원 처리를 할 때 사용
  - 중첩 반복문 사용하기(1)

---

```
*
**
***
****
*****
*****
*****
*****
*****
```

---

①

외부의 반복문  
: 줄생성(\n)

↓  
l = 1

l = 2

l = 3

l = 4

② 내부의 반복문 : 별 생성(\*)

→  
\* j = 0 → j < i(1) = 1번 반복해서 \* 출력

\*\* j = 0 → j < i(2) = 2번 반복해서 \* 출력

\*\*\* j = 0 → j < i(3) = 3번 반복해서 \* 출력

\*\*\*\* j = 0 → j < i(4) = 4번 반복해서 \* 출력

## SECTION 4-2 반복문(13)

- 중첩 반복문을 사용하는 피라미드(누적 예제)
  - 중첩 반복문 사용하기(1)

---

```
01 <script>
02  // 변수를 선언합니다.
03  let output = "";
04
05  // 중첩 반복문
06  for (let i = 1; i < 10; i++) {
07    for (let j = 0; j < i; j++) {
08      output += '*';
09    }
10    output += '\n';
11  }
12
13  // 출력합니다.
14  console.log(output);
15 </script>
```

---

## SECTION 4-2 반복문(14)

- 중첩 반복문을 사용하는 피라미드(누적 예제)
  - 중첩 반복문 사용하기(2)

[illegible]

②

내부의 반복문  
: 공백 생성

4개  
3개  
2개  
1개

전체 높이  
(4) - j개

③

내부의 두번째 반복문  
: 별 생성(\*)

□ □ □ □ *	1개	} 2k-1개
□ □ □ ***	3개	
□ □ *****	5개	
□ *****	7개	

## SECTION 4-2 반복문(15)

- 중첩 반복문을 사용하는 피라미드(누적 예제)
  - 중첩 반복문 사용하기(2)

---

```
01 <script>
02  // 변수를 선언합니다.
03  let output = ''
04
05  // 반복문
06  for (let i = 1; i < 15; i++) {
07    for (let j = 15; j > i; j--) {
08      output += ' '
09    }
10    for (let k = 0; k < 2 * i - 1; k++) {
11      output += '*'
12    }
13    output += '\n'
14  }
15
16  // 출력합니다.
17  console.log(output)
18 </script>
```

---

## [요점 정리]

- 6가지 키워드로 정리하는 핵심 포인트
  - **for in** 반복문은 **배열의 인덱스**를 기반으로 반복할 때 사용
  - **for of** 반복문은 **배열의 값**을 기반으로 반복할 때 사용
  - **for** 반복문은 **횟수**를 기반으로 반복할 때 사용
  - **while** 반복문은 **조건**을 기반으로 반복할 때 사용
  - **break** 키워드는 switch 조건문이나 반복문을 벗어날 때 사용
  - **continue** 키워드는 반복문 안의 반복 작업을 멈추고 반복문의 처음으로 돌아가 다음 반복 작업을 진행