03. 연산자와 제어문

연산자

연산자

아래와 같은 식이 있을 때

age = currentYear - 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 : 변숫값을 1 증가시킴
- 감소(--) 연산자 : 변숫값을 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
```

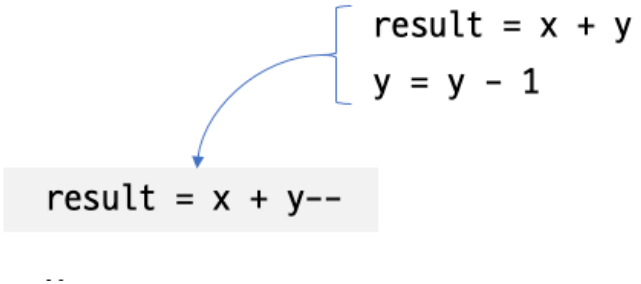
```
a = 10
++a                // 9
a                  // 9
```

산술 연산자 - 증감 연산자

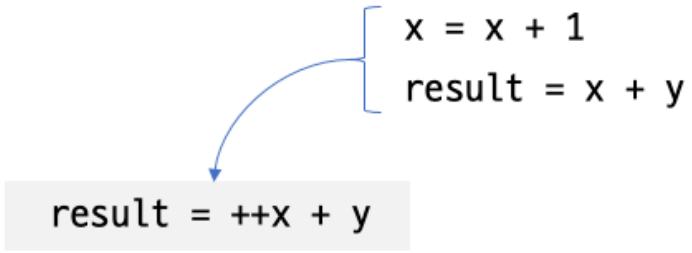
증감 연산자는 연산자가 어느 위치에 붙느냐에 따라 처리 방법이 달라짐

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

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




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



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.	y = x + 3
+=	y = y + x를 의미합니다.	y += x
-=	y = y - x를 의미합니다.	y -= x
*=	y = y * x를 의미합니다.	y *= x
/=	y = y / x를 의미합니다.	y /= x
%=	y = y % x를 의미합니다.	y %= x

```
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 + "세
입니다.");
```

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

```
result = 10;
userNumber = prompt(“10보다 작은 수 입력”)
result = result + userNumber    // result += userNumb
er
```

비교 연산자

- 피연산자 2개의 값을 비교해서 true나 false로 결괏값 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

문자열의 비교

컴퓨터에서 문자를 숫자에 일대일로 대응한 값을 가리킨다.
아스키값을 정리한 표를 '아스키 코드 테이블'이라고 하며 인터넷에서 검색할 수 있다

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

불리언boolean 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
}
```

[실습] if문 연습하기

17₩age-1.html 문서에서 연습합니다.

age-1.html의 소스는 앞에서 공부했던 나이 계산 소스입니다.

```
<script>
  const currentYear = 2022;
  let birthYear;
  let age;

  birthYear = parseInt(prompt("태어난 연도를 입력하세요. (YYYY)", "")
);
  age = currentYear - birthYear + 1;
</script>
```

예) 나이가 20 미만이면 '미성년입니다.'라고 표시한 후, 현재 나이를 표시하고
20 이상이면 그냥 현재 나이만 표시하기

```
<script>
  ……
  age = currentYear - birthYear + 1;
  if (age < 20) {
    alert("성인이 아닙니다. ")
  }
  alert(`${currentYear}년 현재,${age}세입니다.`);
</script>
```

if ... else 문

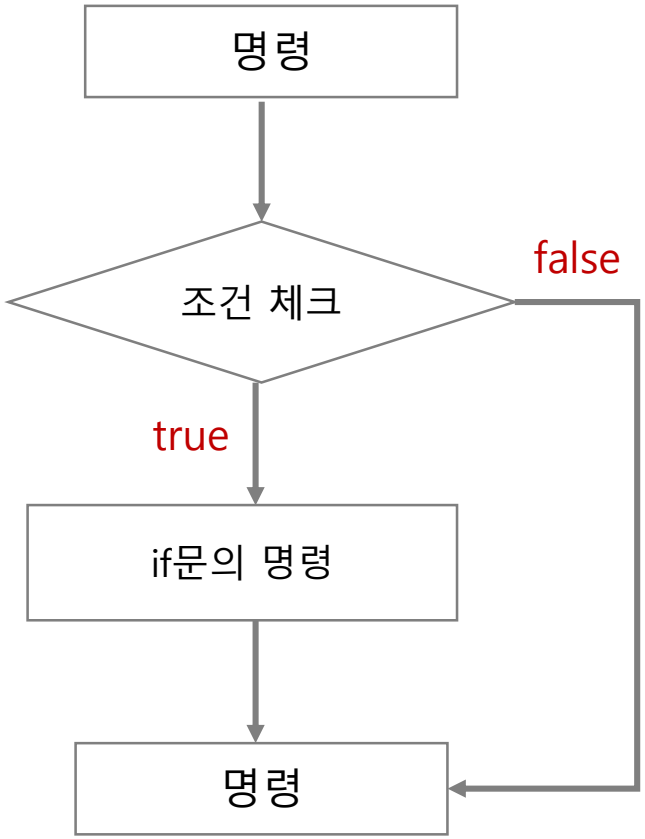
if 문은 결괏값이 true일 때만 실행하므로 true가 아닐 때 명령을 따로 수행할 수 없다.
if~else 문은 if 조건의 결괏값이 true가 아닐 때 실행할 명령을 else 문 다음에 추가한다.

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

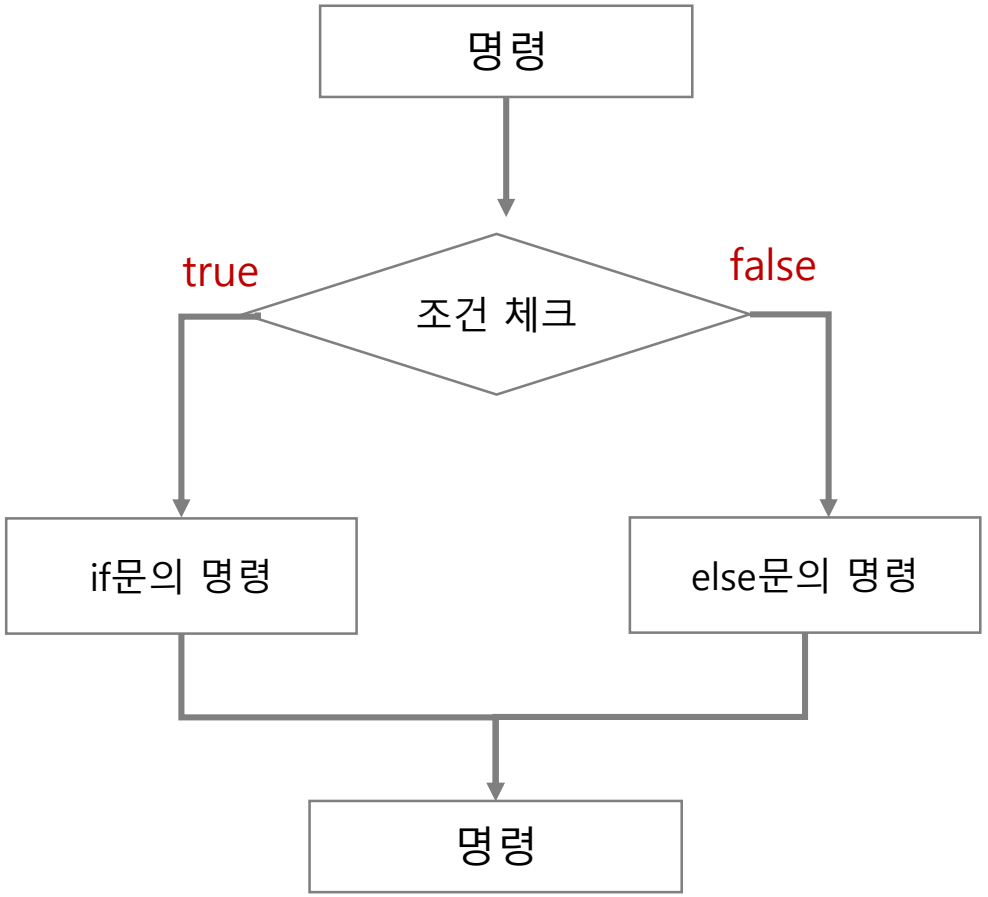
예) if문 다음에 else 문을 사용해서 20 이상이면 '성인입니다.'라고 표시하기

```
<script>
  ……
  if (age < 20) {
    alert("성인이 아닙니다. ")
  } else {
   alert(“성인입니다.”);
  }
  alert(`${currentYear}년 현재,${age}세입니다.`);
</script>
```

if문은 조건을 체크한 후 조건에 맞을 경우에만 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 {
   …
}
```

[실습] 3의 배수 체크하기

[미리 생각해 보기]

- 숫자 입력을 어떻게 할까? – prompt() 문
- 3의 배수를 어떻게 체크할까? – 3으로 나누어 떨어지면 3의 배수, 그외에는 3의 배수 아님
- if … else 문을 사용해서 3의 배수일 때와 아닐 때 할 일을 분리
- 결과는 어떻게 표시할까? – alert()문 (이외에 document.write()나 console.log()도 가능)

1) 프롬프트 문을 사용해 숫자를 입력 받는다

2) 혹시 사용자가 [취소]를 클릭하는 경우도 있으므로, [취소]를 누르지 않았을 때만 실행하자.

3) 숫자를 3으로 나눴을 때 나머지가 0이면 → 3의 배수, 나머지가 0이 아니면 → 3의 배수가 아님.

[실습] 3의 배수 체크하기

1) 프롬프트 문을 사용해 숫자를 입력 받는다

1-1) 프롬프트 창에서 반환한 결괏값을 저장할 변수가 필요하다.

1-2) userNumber 변수에 프롬프트 문 할당

```
<script>
  let userNumber = prompt("숫자를 입력하세요.");
</script>
```

[실습] 3의 배수 체크하기

1) 프롬프트 문을 사용해 숫자를 입력 받는다

2) 혹시 사용자가 [취소]를 클릭하는 경우도 있으므로, [취소]를 누르지 않았을 때만 실행하자.

2-1) if ... else 문 사용

2-2) null인 경우보다 null이 아닌 경우가 많으므로 조건을 'null이 아닌 경우'로 지정

```
<script>
  let userNumber = prompt("숫자를 입력하세요.");

  if (userNumber !== null) {

  }
  else {
    alert("입력이 취소됐습니다.");
  }
</script>
```

[실습] 3의 배수 체크하기

1) 프롬프트 문을 사용해 숫자를 입력 받는다
2) 혹시 사용자가 [취소]를 클릭하는 경우도 있으므로,
   [취소]를 누르지 않았을 때만 실행하자.
3) 숫자를 3으로 나눴을 때
   3-1) 나머지 값을 구하려면 나머지 연산자 % 사용
   3-2) 프롬프트에서 가져온 값을 숫자로 변환한 후 계산

```
<script>
  let userNumber = prompt("숫자를 입력하세요.");

  if (userNumber !== null) {
    userNumber = parseInt(userNumber);
    if (userNumber % 3 === 0)
      alert("3의 배수입니다.");
    else
      alert("3의 배수가 아닙니다.");
  }
  else {
    alert("입력이 취소됐습니다.");
  }
</script>
```

truthy 값, falsy 값

프롬프트 창에서 값을 입력하지 않고 [확인]을 누른 경우도 체크하려면
값이 입력되었는지를 체크할 때 truthy 값을 사용할 수도 있다.

```
let input = prompt("이름을 입력하세요.")
if (input) {                    // input에 값이 들어있다면 truthy
  alert(`${input}님, 어서오세요.`);
}
else {
  alert(`이름을 입력하지 않았습니다. `)
}
```

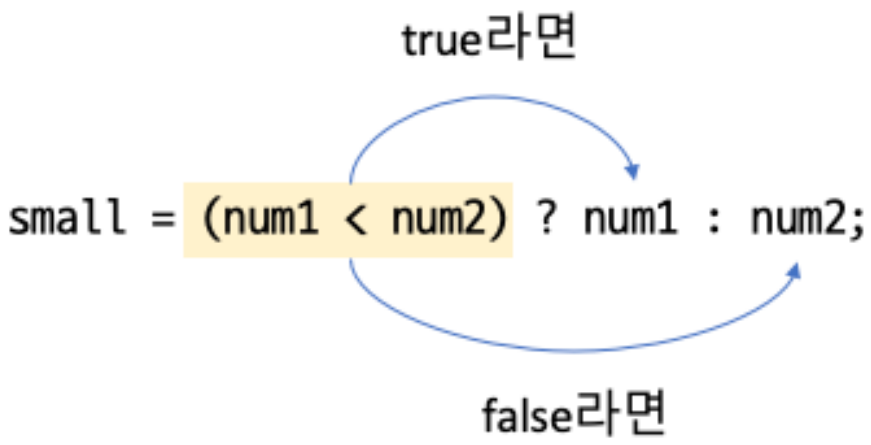
조건 연산자

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 : nu
m2;
```



[실습] 짝수, 홀수 구분하는 프로그램

[미리 생각해 보기]

- 사용자가 [취소] 버튼을 클릭했다면 어떻게 해야 할까?
- 짝수와 홀수는 어떻게 구별할까?

```
let userNumber = prompt("숫자를 입력하세요");


if ( userNumber !== null) {
  userNumber = parseInt(userNumber);
  (userNumber % 2 === 0) ? alert (`${userNumber} : 짝수`) : alert(`${userNumber} : 홀수
`);
}
```

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
 }
```

예) 프롬프트 창을 통해서 신청 과목을 숫자로 입력받고 표시하기

```
let subject = prompt("신청 과목 선택. 숫자만 입력(1-HTML, 2-CSS, 3-Javascript)");

if (subject !== null) {
  switch(subject) {
    case "1" : document.write("HTML을 신청했습니다.");
      break;
    case "2" : document.write("CSS를 신청했습니다.");
      break;
    case "3" : document.write("Javascript를 신청했습니다.");
      break;
    default : document.write("잘못 입력했습니다. 다시 입력해 주세요.")
  }
}
```

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

피연산자1 피연산자2

op1	op2	**op1 || op2**	**op1 && op 2**	**!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 결괏값 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 결괏값 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
```

(예) 입력한 두 개의 숫자가 모두 짝수인지 체크하기

```
const num1 = parseInt(prompt("첫번째 양의 정수 : "));
const num2 = parseInt(prompt("두번째 양의 정수 : "));
let str;

// AND 연산. 둘다 true여야 결괏값 true
if (num1 % 2 === 0 && num2 % 2 === 0) {
  str = "두 수 모두 짝수입니다." ;
} else {
  str = "짝수가 아닌 수가 있습니다.";
}
alert(str);
```

단축 평가값 활용하기

조건식은 왼쪽에서 오른쪽으로 진행하면서 처리한다.
첫 번째 조건만 보고도 true인지, false인지 결정할 수 있다면 좀 더 빠르게 조건식을 처리할 수 있음
두 가지 이상의 조건을 함께 체크하는 조건식을 만들 때에는 첫 번째 조건을 보고 빠르게 판단할 수 있도록 작성하는 것이 좋다.

```
let x = 10
let y = 20
if ( x > 15 && y > 15) alert("둘 다 15보다 큽니다.")          // y > 15는 실행하지 않음
if ( y > 15 || x > 15) alert("둘 중 하나는 15보다 큽니다.")  // x > 15는 실행하지 않음
```

반복문

반복문

- 반복문은 같은 동작을 여러 번 실행하기 위해 사용하는 문이다.
- 반복문을 사용하면 불필요하게 여러 명령들을 늘어놓지 않아도 명령을 반복 실행할 수 있다.
- 그만큼 소스도 깔끔해지고 소스가 짧아지는만큼 실행도 빨라진다.

for 문

- 자바스크립트에서 가장 많이 사용하는 반복문
- 조건값이 일정하게 커지면서 명령을 반복 실행할 때 편리하다.
- for문에서는 몇 번 반복했는지 기록하기 위해 카운터를 사용하고 for문의 첫 번째 항에서 카운터 변수를 지정한다.

```
for (초깃값; 조건; 증가식) { ... }
```

초깃값은 처음에 한 번만 할당하고 조건 체크와 명령 실행, 증가식을 계속 반복한다!

- 초깃값: 몇 번 반복할지 지정하는 카운터 변수를 선언하고 초기화한다. 초깃값은 0이나 1부터 시작한다.
- 조건: 문장을 반복하기 위해 체크할 조건 부분. 이 조건을 만족해야 for문에 있는 명령을 반복할 수 있다.
- 증가식: 문장을 실행한 후 카운터 변수를 증가시키는 부분. 보통 카운터값을 하나 더 증가시킴

for 문

1부터 10까지 순서대로 콘솔 창에 표시하는 for문을 생각해 보자

1) 카운터 변수 선언 및 할당
대부분 카운터 변수를 i로 사용한다. 1부터 시작하자.

```
for(let i = 1; )
```

2) 조건을 지정한다.
여기에서는 10까지 표시 (끝나는 값 10 포함)

```
for(let i = 1; i <= 10; i++)
```

3) 카운터 값 증가시키기. 1씩 증가

```
for(let i = 1; i <= 10; i++)
```

4) 실행할 명령 작성.
콘솔 창에 i값 표시. 줄바꿈 필요하면 넣기

```
for(let i = 1; i <= 10; i++) {
  console.log(i + '₩n');
}
```

(예) 배열에서 값 가져오기

배열의 인덱스를 사용해서 배열 요소 값에 접근할 수 있다.

배열 인덱스는 0부터 시작!

언제까지 반복할까? → 배열 요소의 갯수만큼 → 배열 갯수는 length 속성에 들어 있다.

```
const students = ["Park", "Kim", "Lee", "Kang"];

for (let i = 0; i < students.length; i++) {
  document.write(`${students[i]}, `);
}
```

Park. Kim. Lee. Kang.

forEach 문

배열의 크기(length)가 정해져 있지 않을 경우에 사용한다.

콜백 함수란 다른 함수 안에 사용할 수 있는 함수를 가리킴

```
배열명.forEach(콜백 함수) { ..
. }
```

예) 배열 요소의 값 표시하기

```
const students = ["Park", "Kim", "Lee", "Kang"
];

students.forEach(function(student) {
  document.write(`${student}. `)
});
```

students 배열에 있는 각 요소를 student라고 하고

```
students.forEach(function(student) {
  document.write(`${student}.`)
});
```

student 표시

ES6의 화살표 함수를 사용하면 더 간단히 표현 가능

```
const students = ["Park", "Kim", "Lee", "Kang"];

students.forEach(student => document.write(`${student}. `));
```

for...in문 사용해서 객체 값 가져오기

객체에서 사용할 수 있는 반복문.

for...in문은 반복해서 객체의 키를 가져온다.

각 키의 값을 알고 싶다면 가져온 키를 사용해서 객체에 접근한다.

배열도 객체이기 때문에 배열에서도 for...in문을 사용할 수 있다.

```
for (변수 in 객체) { ...
}
```

```
const gitBook = {
  title : "깃&깃허브 입문",
  pubDate : "2019-12-06",
  pages : 272,
  finished : true
}

for(key in gitBook) {
  document.write(`${key} : ${gitBook[key]}<br>
`);
}
```

title : 깃&깃허브 입문
pubDate : 2019-12-06
pages : 272
finished : true

for...of문 사용해서 반복 가능 객체 값 가져오기

- 문자열이나 배열처럼 그 안의 값이 순서대로 나열되어 있는 객체를 이터러블(iterable) 객체라고 함.
- 이터러블 객체에서는 for...of 문을 사용할 수 있다.

```
for (변수 of 객체) { ... }
```

```
const students = ["Park", "Kim", "Lee", "Kang"
];

// students에 있는 student가 있는 동안 계속 반복
for (let student of students) {
  document.write(`${student}. `);
}
```

Park. Kim. Lee. Kang.

for문 중첩하기

for문 안에 또다른 for문을 사용하는 것을 "중첩한다"고 한다.

(예) *가 30개 표시되는 줄을 5개 만들려면?

```
for(let k = 0; k < 5; k++) {
    for(var i = 0; i < 30; i++)
    {
        document.write('*');
    }
    document.write('<br>'); // 줄바꿈
}
```

다른 for문의 카운터 변수와 겹치지 않게

① 바깥쪽 for문 실행 (k = 0)

② 안쪽 for문을 실행해 30번 반복하고 빠져나온다.

③
 태그를 추가해서 줄을 바꾼다.

④ 바깥쪽 for문의 조건식이 false가 될 때까지 반복한다.

[실습] 구구단 만들기

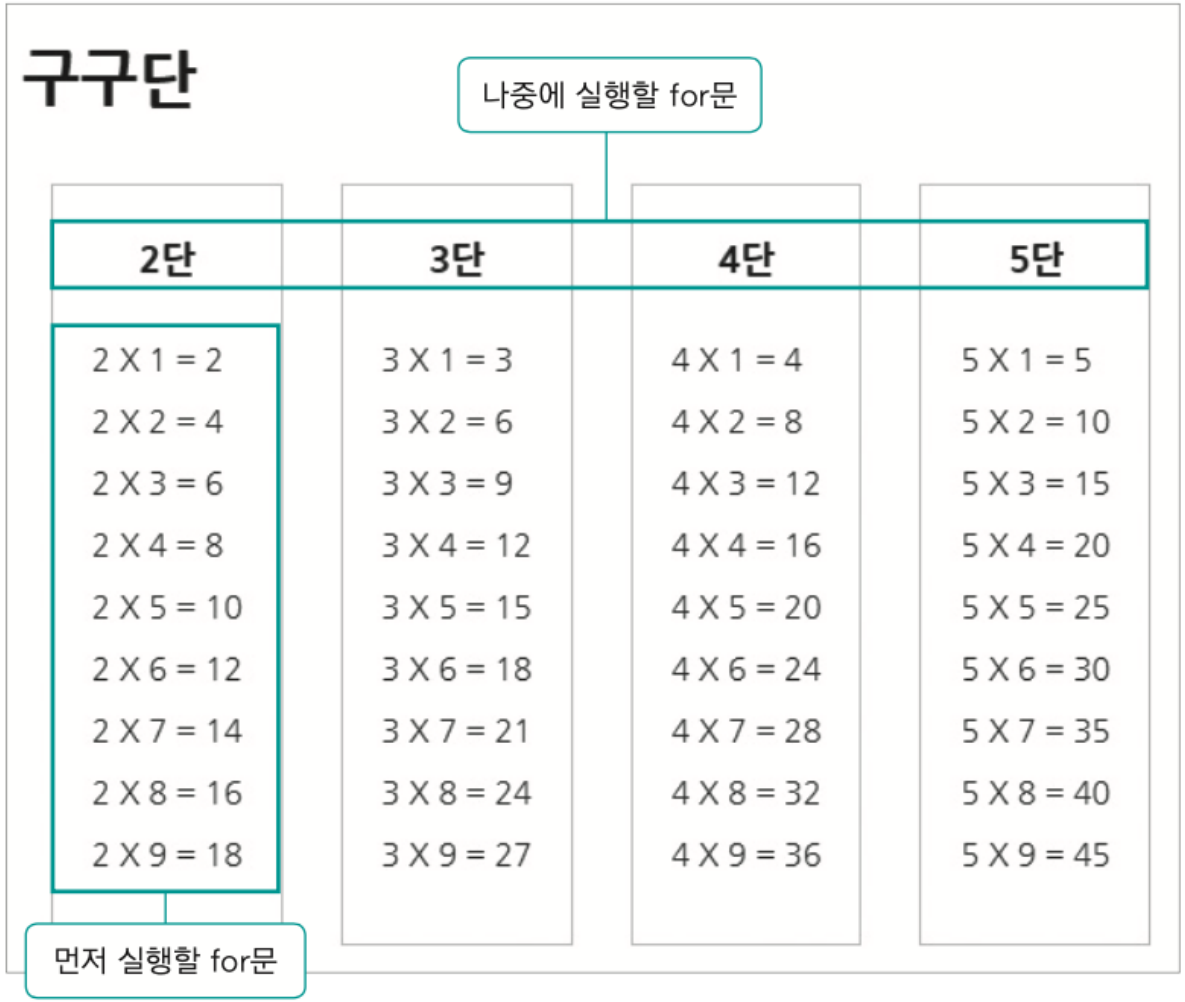
구구단은 1단부터 9단 까지 이루어져 있고,
각 단은 1부터 9까지의 곱으로 이루어짐.
→ 중첩된 for문을 사용해서 작성

안쪽 for문

- 각 단에서 1부터 9까지 곱하는 부분
- 카운터 변수를 j로 놓고 for문 작성

바깥쪽 for문

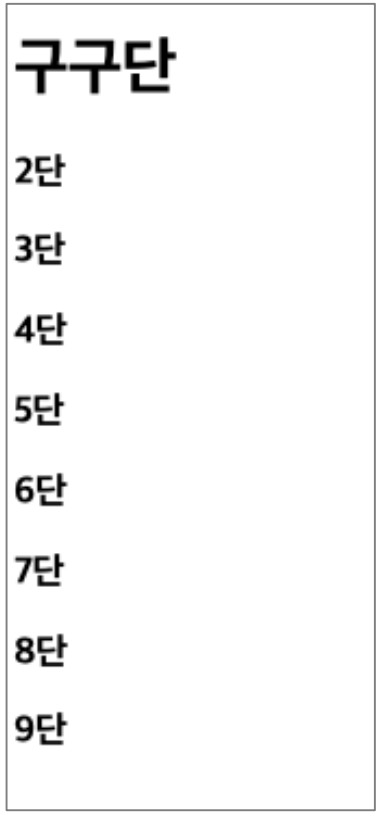
- 구구단의 '단'에 해당하는 부분
- 카운터 변수를 i로 놓고 for문 작성



[실습] 구구단 만들기

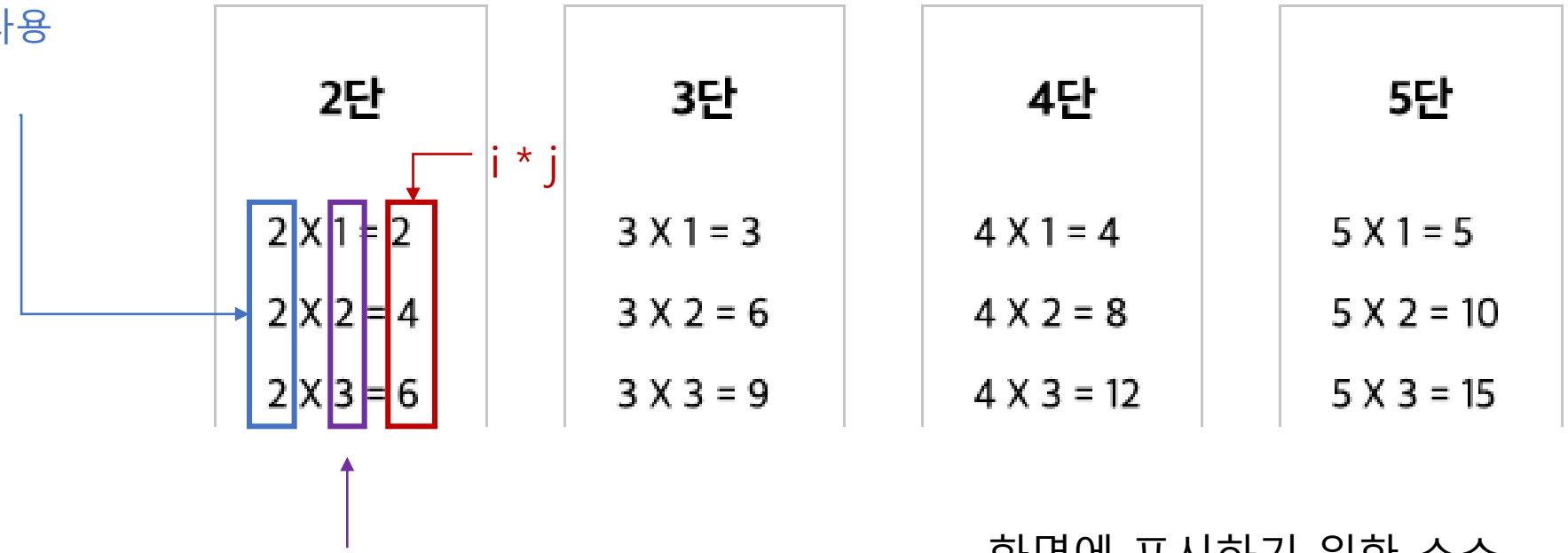
```
for(var i = 2; i <= 9; i++) {    // 2단부터 9단까지
  document.write(“<h3>” + i + “단</h3”>);
  for (var j = 1; j <= 9; j++) {  // 1부터 9까지


  }
}
```



[실습] 구구단 만들기

'단'을 나타내므로,
i변수 사용



'곱하는 값'을 나타내므로, j변수 사용

화면에 표시하기 위한 소스

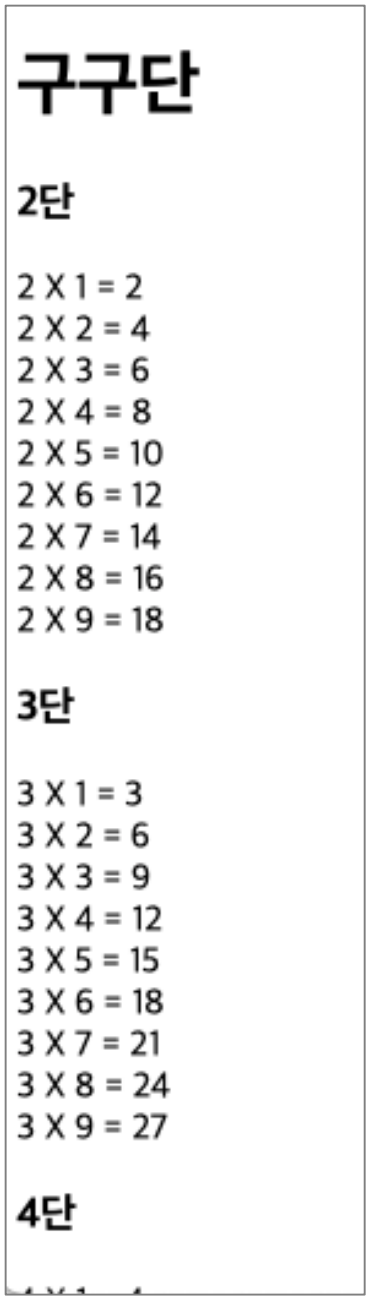
```
i +" X " + j + " = " + i * j + "
<br>"
```

또는

```
`${i} X ${j} = ${i * j}<br>`
```

[실습] 구구단 만들기

```
for(var i = 2; i <= 9; i++) {    // 2단부터 9단까지
  document.write("<h3>" + i + "단</h3">);
  for (var j = 1; j <= 9; j++) {  // 1부터 9까지
   document.write(`${i} X ${j} = ${i * j}<br>`);
  }
}
```



[실습] 구구단 만들기

구구단을 가로로 배치하려면? → CSS 사용

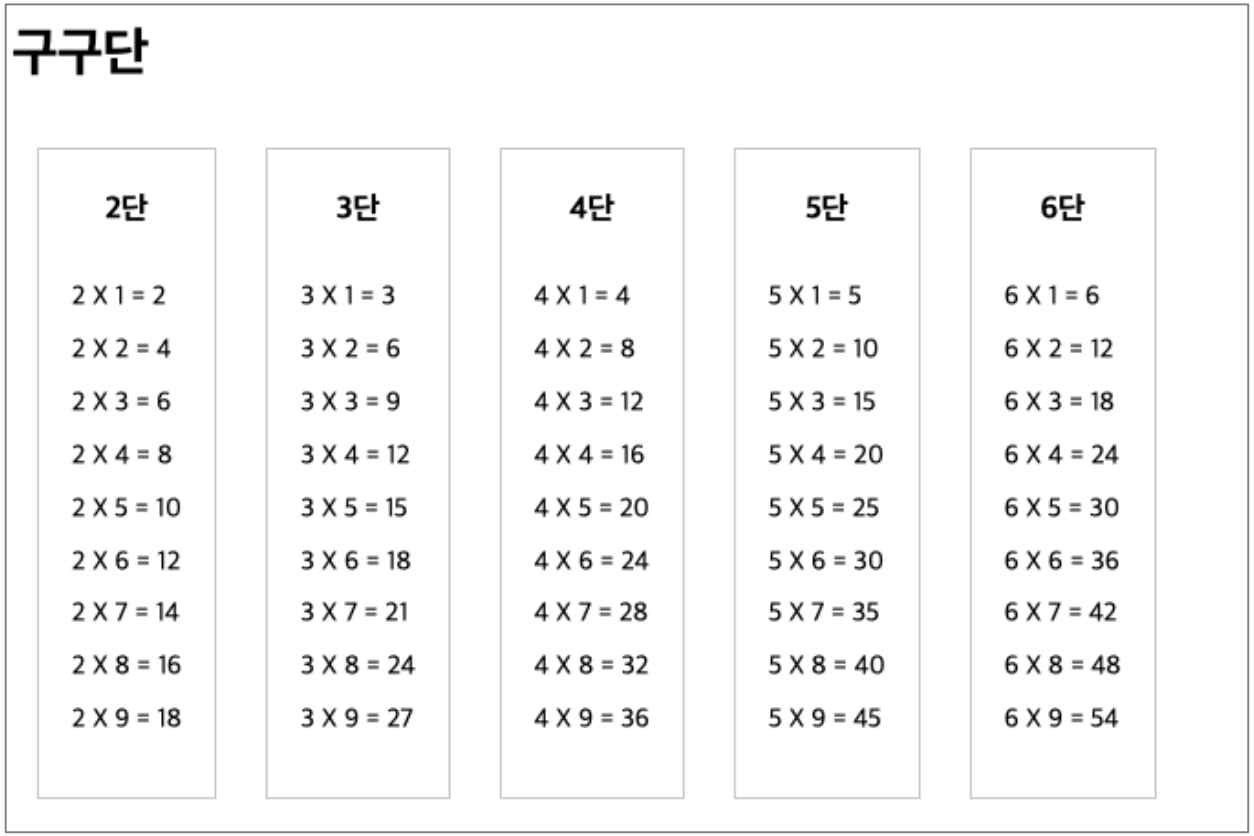
```
<head>
  <meta charset="UTF-8">
  ……
  <title>구구단 - for문</title>
  <link rel="stylesheet" href="css/gugudan.
css">
</head>
```

```
for(var i = 2; i <= 9; i++) {
  document.write("<div>");
  document.write("<h3>" + i + "단</h3>");
  for (var j = 1; j <= 9; j++) {
    document.write(`${i} X ${j} = ${i * j}<
br>`);
  }
  document.write("</div>");
}
```

[실습] 구구단 만들기

css₩gugudan.css

```
div {
  display:inline-block;
  padding:0 20px 30px 20px;
  margin:15px;
  border:1px solid #ccc;
  line-height:2;
}
div h3 {
  text-align:center;
  font-weight:bold;
}
```



while 문, do ... while 문

while문과 do...while문은 초기값이나 반복 횟수 없이 조건만 주어졌을 때 사용한다.

while

- 조건이 참(true)인 동안 문장 반복
- 조건부터 체크한 후 true일 경우에만 반복
- 조건이 false라면 한 번도 실행하지 않음

```
while (조건) {
    실행할 명령
}
```

do ... while

- 일단 문장을 한번 실행한 후
- 조건이 참(true)인 동안 문장 반복
- 조건이 false라도 최소한 한번은 실행됨

```
do {
    실행할 명령
} while (조건)
```

```
let stars = parseInt(prompt("별의 개수 : "
));


while(stars > 0) {
  document.write('*');
  stars--;
}
```

- 별의 갯수를 5로 했을 때 → 5개 찍힘
- 별의 갯수를 0으로 했을 때 → 아무것도 안 찍힘

```
let stars = parseInt(prompt("별의 개수 :
"));


do {
  document.write('*');
  stars--;
} while(stars > 0)
```

- 별의 갯수를 5로 했을 때 → 5개 찍힘
- 별의 갯수를 0으로 했을 때 → 1개 찍힘

break문

반복문은 주어진 조건에 따라 문장을 반복하기 때문에 종료 조건이 돼야 반복이 끝남.

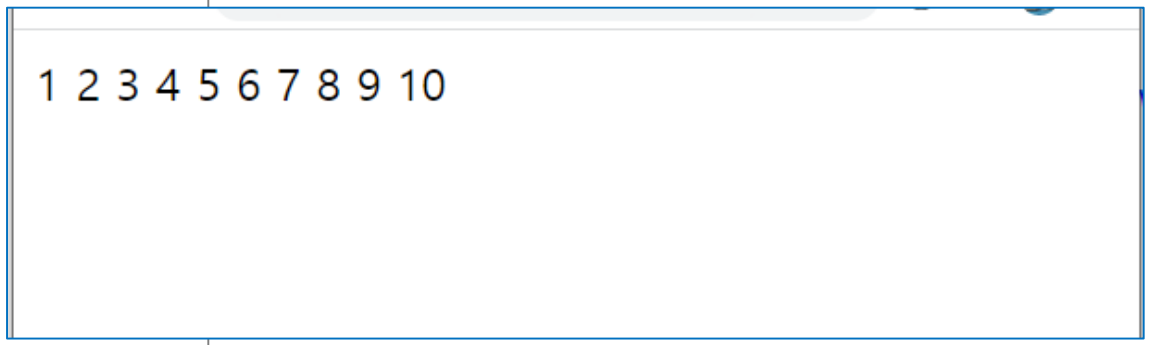
종료 조건이 되기 전에 반복문을 빠져나와야 한다면? → break문 사용

(switch문에서도 case에 맞는 값이 있을 경우 명령 실행 후 break문을 사용해 switch문 빠져나옴)

```
break
```

예) 숫자 1부터 50까지 화면에 표시하다가 10에서 멈추게 하려면??

```
let n = 50;
for(let i = 1; i <= n; i++) {
  document.write(`${i} `);
    if (i === 10) {
        break;
      }
}
```



continue문

특정 조건이 됐을 때 실행하던 반복 문장을 더 이상 실행하지 않고 반복문의 맨 앞으로 되돌아감

→ 반복 과정을 한 차례 건너뛰게 됨.

`continue`

예) 숫자 1부터 10까지 중에서 짝수만 표시하려면??

```
for(let i = 1; i <= 10; i++) {
  if (i % 2 === 1) {
    continue;
  }
  document.write(`${i}<br>`);
}
```