

# 14. 자바스크립트 기본 문법

14-1 변수 알아보기

14-2 자료형 이해하기

14-3 연산자 알아보기

14-4 조건문 알아보기

14-5 반복문 알아보기



# 변수 알아보기

## 변수란

- 변수(variable) : 값이 여러 번 달라질 수 있는 데이터
- 상수(constant) : 값을 한번 지정하면 바뀌지 않는 데이터



## 변수 선언의 규칙

- 변수 이름
  - 영어 문자, 언더스코어(\_), 숫자를 사용한다
  - 첫 글자는 영문자, \_기호, \$기호를 사용한다
  - 띄어쓰기나 기호는 허용하지 않는다
  - 예) `now`, `_now`, `now25` (사용할 수 있음)
  - 예) `25now`, `now 25`, `*now` (사용할 수 없음)
- 영어 대소문자를 구별하며 예약어는 변수 이름으로 사용할 수 없다
- 여러 단어를 연결할 때는 하이픈이나 언더스코어를 사용할 수 있고 중간에 대문자를 섞어 쓸 수도 있다
- 예) `total-area`, `total_area`, `totalArea` 등
- 변수 이름은 의미있게 작성한다



# 변수 알아보기

## 변수 선언하기

- var 뒤에 변수 이름 작성
- var를 한번만 쓰고 뒤에 여러 개의 변수를 한꺼번에 선언할 수도 있음

기본형 var 변수명

### 변수 선언하기

```
var currentYear; // 올해 연도 변수 선언
var birthYear;   // 태어난 연도 변수 선언
var age;         // 계산한 나이 변수 선언
```

### 변수 한꺼번에 선언하기

```
var currentYear, birthYear, age; // 올해 연도, 태어난 연도, 계산한 나이 변수 선언
```

## 변수에 값 할당

'=' 기호 다음에 값을 저장



### 변수 선언과 값 할당 따로 하기

```
var birthYear; // 태어난 연도 변수 선언
birthYear = 1995; // 변수에 값 할당
```



### 변수 선언과 값 할당 같이 하기

```
var currentYear = 2021; // 올해 연도 변수 선언하고 값 할당하기
```

# 자료형 이해하기

## 자료형이란

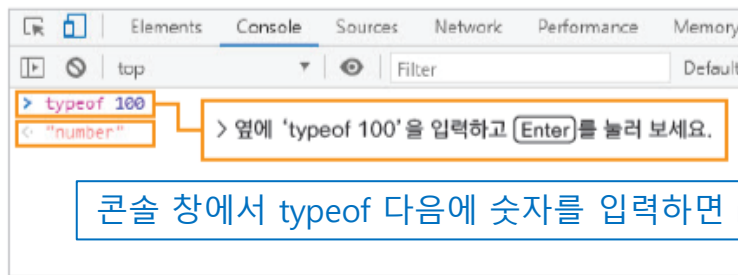
컴퓨터가 처리할 수 있는 자료의 형태

| 종류       |           | 설명  | 예시  |
|----------|-----------|---|---|
| 기본<br>유형 | 숫자형       | 따옴표 없이 숫자로만 표기합니다.  | <code>var birthYear = 2000;</code>  |
|          | 문자열       | 작은따옴표(' ')나 큰따옴표(" ")로 묶어서 나타냅니다. 숫자를 따옴표로 묶으면 문자로 인식합니다.               | <code>var greeting = "Hello!";</code><br><code>var birthYear = "2000";</code> |
|          | 논리형       | 참(true)과 거짓(false)이라는 2가지 값만 있는 유형입니다. 이때 true와 false는 소문자로만 표시합니다.     | <code>var isEmpty = true;</code>  |
| 복합<br>유형 | 배열        | 하나의 변수에 여러 개의 값을 저장합니다.   | <code>var seasons = ['봄', '여름', '가을', '겨울'];</code>                           |
|          | 객체        | 함수와 속성을 함께 포함합니다.   | <code>var date = new Date();</code>   |
| 특수<br>유형 | undefined | 자료형이 지정되지 않았을 때의 상태입니다. 예를 들어 변수 선언만 하고 값을 할당하지 않은 변수는 undefined 상태입니다. |   |
|          | null      | 값이 유효하지 않을 때의 상태입니다.  |   |

## 숫자형(number)

숫자

- 정수 : 소수점 없는 숫자
  - 실수 : 소수점이 있는 숫자
- ※ 자바스크립트는 실수를 정밀하게 계산하지 못함



콘솔 창에서 typeof 다음에 숫자를 입력하면 number라고 표시됨

# 자료형 이해하기

## 문자열(string)

작은따옴표(' ')나 큰따옴표(" ")로 묶은 데이터

```
top Filter Default levels
> typeof "안녕하세요?"
< "string"
> typeof "12345"
< "string"
> |
```

콘솔 창에서 typeof 다음에 따옴표로 묶은 내용을 입력하면 string이라고 표시됨

## 논리형(boolean)

- 참true이나 거짓false의 값을 표현하는 자료형. 불린 유형이라고도 함.
- 조건을 확인해서 조건이 맞으면 true, 맞지 않으면 false라는 결괏값 출력

```
top Filter Default levels
> 100 < 10
< false
> 100 > 10
< true
> |
```

조건에 따라 true나 false 값을 표시하는 논리형

## undefined 유형

- 자료형이 정의되지 않았을 때의 데이터 상태
- 변수 선언만 하고 값이 할당되지 않은 자료형

## null 유형

- 데이터 값이 유효하지 않은 상태
- 변수에 할당된 값이 유효하지 않다는 의미

# 자료형 이해하기

## 배열(array)

하나의 변수에 여러 값을 저장할 수 있는 복합 유형

기본형 배열명["값1 ", "값2", .....]

배열명[ ]

빈 배열 선언

### 예) 계절 이름을 프로그램에 사용할 경우

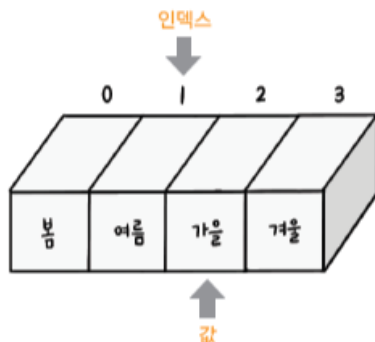
배열을 사용하지 않는다면

```
var spring = "봄";  
var summer = "여름";  
var fall = "가을";  
var winter = "겨울";
```

배열을 사용한다면

```
var season = ["봄", "여름", "가을", "겨울"];
```

변수 이름 → Season



### 자바스크립트의 데이터 유형 자동 변환

자바스크립트의 편리한 점이면서도 약점인 부분이 데이터 유형이 유연하다는 것입니다. 다시 말해 변수의 데이터 유형이 중간에 바뀔 수 있다는 것이죠.

책에 있는 '나이 계산 프로그램'에서는 프롬프트 창을 통해 사용자의 태어난 해를 입력받는데, 이때 프롬프트 창에서 입력받은 값은 문자열이지만 사칙연산에 사용된 문자열은 자동으로 숫자형으로 변환되어 계산됩니다

# 연산자 알아보기


## 산술 연산자

수학 계산을 할 때 사용하는 연산자

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

나누기 연산자(/) : 나눈 값 자체

나머지 연산자(%) : 나눈 후에 남은 나머지 값

 나누기 연산자와 나머지 연산자 비교하기

```
var numberOne = 15 / 2; // numberOne은 7입니다
var numberTwo = 15 % 2; // numberTwo는 1입니다
```

## 할당 연산자(대입 연산자)


연산자 오른쪽의 실행 결과를 왼쪽 변수에 할당하는 연산자

| 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>    |

## 연결 연산자

둘 이상의 문자열을 합쳐서 하나의 문자열로 만드는 연산자

'+' 기호 사용

 연결 연산자 사용하여 출력하기

```
document.write (birthYear + "년에 태어난 사람의 나이는 " + age + "세입니다.");
```

# 연산자 알아보기

## 비교 연산자

피연산자 2개의 값을 비교해서 true나 false로 결과값 반환

| 종류  | 설명                                   | 예시        |       |
|-----|--------------------------------------|-----------|-------|
|     |                                      | 조건식       | 결과값   |
| ==  | 피연산자가 서로 같으면 true입니다.                | 3 == "3"  | true  |
| === | 피연산자도 같고 자료형도 같으면 true입니다.           | a === "3" | false |
| !=  | 피연산자가 서로 같지 않으면 true입니다.             | 3 != "3"  | false |
| !== | 피연산자가 같지 않거나 자료형이 같지 않으면 true입니다.    | 3 !== "3" | true  |
| <   | 왼쪽 피연산자가 오른쪽 피연산자보다 작으면 true입니다.     | 3 < 4     | true  |
| <=  | 왼쪽 피연산자가 오른쪽 피연산자보다 작거나 같으면 true입니다. | 3 <= 4    | true  |
| >   | 왼쪽 피연산자가 오른쪽 피연산자보다 크면 true입니다.      | 3 > 4     | false |
| >=  | 왼쪽 피연산자가 오른쪽 피연산자보다 크거나 같으면 true입니다. | 3 >= 4    | false |

### == 연산자 와 != 연산자

피연산자의 자료형을 자동으로 변환해서 비교

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

### === 연산자 와 !== 연산자

피연산자의 자료형을 변환하지 않음

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

← 프로그램에서 값을 비교할 때 더 많이 사용

## 논리 연산자

true와 false가 피연산자인 연산자  
조건을 처리할 때 사용

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



# 조건문 알아보기

## if 문과 if ~ else 문

피연산자 2개의 값을 비교해서 true나 false로 결과값 반환  
하나의 if ~ else 문 안에 다른 if ~ else 문을 넣을 수 있다

```
기본형 if (조건) {  
    조건 결과값이 true일 때 실행할 명령  
}
```

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



Do it! 3의 배수 확인하기 2

예제 파일 14\if-2.html

(... 생략 ...)

<script>

```
var userNumber = prompt("숫자를 입력하세요.");
```

```
if(userNumber !== null) { // 입력값이 null이 아니면 if~else 문을 실행
```

```
    if(userNumber % 3 === 0)
```

```
        alert("3의 배수입니다.");
```

```
    else
```

```
        alert("3의 배수가 아닙니다.");
```

```
}
```

```
else
```

```
    alert("입력이 취소됐습니다."); // 입력값이 null이면 알림 창을 보여 줌
```

</script>

(... 생략 ...)

if~else 문 안에 중첩된 if~else 문을 사용합니다.

127.0.0.1:5500 내용:

숫자를 입력하세요.

14

확인

취소

127.0.0.1:5500 내용:

3의 배수가 아닙니다.

확인

# 조건문 알아보기

## 조건 연산자로 조건 체크하기

조건이 하나이고 true일 때와 false일 때 실행할 명령이 각각 하나뿐일 때 간단하게 사용할 수 있음

기본형 (조건) ? true일 때 실행할 명령 : false일 때 실행할 명령

### Do it! 3의 배수 확인하기 2

예제 파일 14\if-2.html

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

  if(userNumber !== null) { // 입력값이 null이 아니면 if~else 문을 실행
    if(userNumber % 3 === 0)
      alert("3의 배수입니다.");
    else
      alert("3의 배수가 아닙니다.");
  }
  else
    alert("입력이 취소됐습니다."); // 입력값이 null이면 알림 창을 보여 줌
</script>
(... 생략 ...)
```

if~else 문 안에 중첩된 if~else 문을 사용합니다.



### Do it! 조건 연산자를 사용해 3의 배수 확인하기

예제 파일 14\if-3.html

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

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

조건 연산자를 사용했습니다.

127.0.0.1:5500 내용:  
숫자를 입력하세요.

14

확인

취소

127.0.0.1:5500 내용:  
3의 배수가 아닙니다.

확인

# 조건문 알아보기

## 논리 연산자로 조건 체크하기

- 조건을 2개 이상 체크할 경우에는 조건 연산자를 사용해 조건을 만들
- 두 조건이 true일 경우, 조건 1개만 true일 경우처럼 여러 경우를 따질 때 논리 연산자 사용

### AND 연산자 (&&)

피연산자 2개 중에서 false가 하나라도 있으면 결과값은 false

| op 1  | op 2  | op 1 && op 2 |
|-------|-------|--------------|
| false | false | false        |
| false | true  | false        |
| true  | false | false        |
| true  | true  | true         |

### OR 연산자 (||)

피연산자 2개 중에서 true가 하나라도 있으면 결과값은 true

| op 1  | op 2  | op 1    op 2 |
|-------|-------|--------------|
| false | false | false        |
| false | true  | true         |
| true  | false | true         |
| true  | true  | true         |

### NOT 연산자 (!)

피연산자를 반대로 뒤집음

| op    | !op   |
|-------|-------|
| false | true  |
| true  | false |

# 조건문 알아보기

## switch문

- 처리할 명령이 많을 경우 switch 문이 편리

```
기본형 switch (조건)
{
    case 값1: 명령1
        break
    case 값2: 명령2
        break
    .....
    default: 명령n
}
```

- 조건은 case 문의 값과 일대일로 일치해야 함
- case 문의 명령 실행 후 switch 문 빠져나옴
- 조건과 일치하는 case 문이 없다면 default 문 실행
- default 문에는 break 문이 없음



Do it! switch 문으로 조건 체크하기

예제 파일 14\switch.html

(... 생략 ...)

<script>

```
var session = prompt("관심 세션을 선택해 주세요. 1-마케팅, 2-개발, 3-디자인");
```

```
switch(session) {
```

```
    case "1": document.write("<p>마케팅 세션은 <strong>201호</strong>에서 ..... </p>")
               break;
```

```
    case "2": document.write("<p>개발 세션은 <strong>203호</strong>에서 ..... </p>")
               break;
```

```
    case "3": document.write("<p>디자인 세션은 <strong>205호</strong>에서 ..... </p>")
               break;
```

```
    default: alert("잘못 입력했습니다."); // 1, 2, 3이 아닌 값을 입력받으면 출력
```

```
}
```

</script>

(... 생략 ...)

127.0.0.1:5500 내용:

관심 세션을 선택해 주세요. 1-마케팅, 2-개발, 3-디자인

2

확인

취소

개발 세션은 **203호**에서 진행됩니다.

# 반복문 알아보기

## for 문

기본형

```
for(초깃값; 조건; 증가식) {  
    실행할 명령  
}
```

- 1 초깃값: 카운터 변수를 초기화합니다. 초깃값은 0이나 1부터 시작합니다.
- 2 조건: 명령을 반복하기 위해 조건을 체크합니다. 이 조건을 만족해야 그다음에 오는 명령을 실행할 수 있습니다.
- 3 증가식: 명령을 반복한 후 실행합니다. 보통 카운터 변수를 1 증가시키는 용도로 사용합니다.



for 문을 사용해 1부터 5까지 숫자 더하기

예제 파일 14\repeat-2.html

(... 생략 ...)

<script>

var i;

var sum = 0;

```
for(i = 1; i < 6; i++) {  
    sum += i;  
}
```



for 문을 사용해  
코드를 간단하게 작성했죠!

document.write("1부터 5까지 더하면 " + sum);


</script>

(... 생략 ...)

- 1 카운터로 사용할 변수 i에 초깃값 1 지정
- 2 i = 1 → i < 6 체크 → (조건 만족함) → sum += i 실행 → i++ 실행
- 3 i = 2 → i < 6 체크 → (조건 만족함) → sum += i 실행 → i++ 실행
- 4 i = 3 → i < 6 체크 → (조건 만족함) → sum += i 실행 → i++ 실행
- 5 i = 4 → i < 6 체크 → (조건 만족함) → sum += i 실행 → i++ 실행
- 6 i = 5 → i < 6 체크 → (조건 만족함) → sum += i 실행 → i++ 실행
- 7 i = 6 → i < 6 체크 → (조건 만족하지 않음) → for 문을 빠져나옴

# 반복문 알아보기

## 중첩된 for 문

 **Do it!** for 문 2개로 구구단 만들기

예제 파일 14\gugudan-1.html

(... 생략 ...)

```
<h1>구구단</h1>
```

```
<script>
```

```
var i, j;
```

```
for (i = 1; i <= 9; i++) {  
    document.write("<h3>" + i + "단</h3>");
```

```
    for (j = 1; j <= 9; j++) {  
        document.write(i + " X " + j + " = " + i*j + "<br>");
```

```
    }
```

```
}
```

```
</script>
```

(... 생략 ...)

## 구구단

### 1단

1 X 1 = 1  
1 X 2 = 2  
1 X 3 = 3  
1 X 4 = 4  
1 X 5 = 5  
1 X 6 = 6  
1 X 7 = 7  
1 X 8 = 8  
1 X 9 = 9

### 2단

2 X 1 = 2  
2 X 2 = 4  
2 X 3 = 6  
2 X 4 = 8  
2 X 5 = 10  
2 X 6 = 12  
2 X 7 = 14  
2 X 8 = 16  
2 X 9 = 18

### 3단

3 X 1 = 3  
3 X 2 = 6  
3 X 3 = 9  
3 X 4 = 12

...

# while 문과 do ~ while 문

## while 문

조건을 체크하고 true라면 { }안의 명령 실행

→ 조건이 false라면 명령은 한 번도 실행하지 않을 수 있음

```
기본형 while (조건) {  
    실행할 명령  
}
```

## do ~ while 문

일단 명령을 한번 실행한 후 조건 체크.

true라면 { } 안의 명령 실행, false라면 { }을 빠져나옴

→ 조건이 false라도 명령은 최소한 한 번은 실행

```
기본형 do {  
    실행할 명령  
} while (조건)
```



Do it! while 문으로 팩토리얼 만들기

예제 파일 14\factorial.html

(... 생략 ...)

<h1>while 문을 사용한 팩토리얼 계산</h1>

<script>

var n = prompt("숫자를 입력하세요.");

var msg = "";

if(n !== null) { // 취소 버튼을 누르지 않았는지 체크

var nFact = 1; // 곱값

var i = 1; // 카운터

while(i <= n) {

nFact \*= i;

i++;

}

msg = n + "! = " + nFact; // 곱값을 표시할 문자열

}

else

msg = "값을 입력하지 않았습니다.";

document.write(msg); // 결과 표시

</script>

(... 생략 ...)

# break 문과 continue 문

## break 문

종료 조건이 되기 전에 반복문을 빠져 나와야 할 때 사용

기본형 break



Do it! break 문으로 구구단을 3단까지만 표시하기

예제 파일 14\gugudan-3.html

(... 생략 ...)

```
<script>
```

```
var i, j;
```

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

```
if(i === 3) break;
```

i값이 3이면 break 문을 실행합니다.

```
</script>
```

## continue 문

조건에 해당되는 값을 만나면 반복문의 맨 앞으로 이동

→ 결과적으로 반복 과정을 한 차례 건너 뛴

기본형 continue



Do it! 1부터 10까지 짝수만 더하기

예제 파일 14\even.html

(... 생략 ...)

```
<h1>짝수끼리 더하기</h1>
```

```
<script>
```

```
var i;
```

```
var n = 10;
```

```
var sum = 0;
```

```
for(i = 1; i <= n; i++) {  
    if (i % 2 === 1) // i가 홀수라면 반복문을 건너뛴  
        continue  
    sum += i;
```

```
document.write(i + " ----- " + sum + "<br>");  
}
```

```
</script>
```

(... 생략 ...)