


혼자 공부하는 자바스크립트

Chapter 02 자료와 변수





Contents

- CHAPTER 02: 자료와 변수

SECTION 2-1 기본 자료형

SECTION 2-2 상수와 변수

SECTION 2-3 자료형 변환



CHAPTER 02 자료와 변수

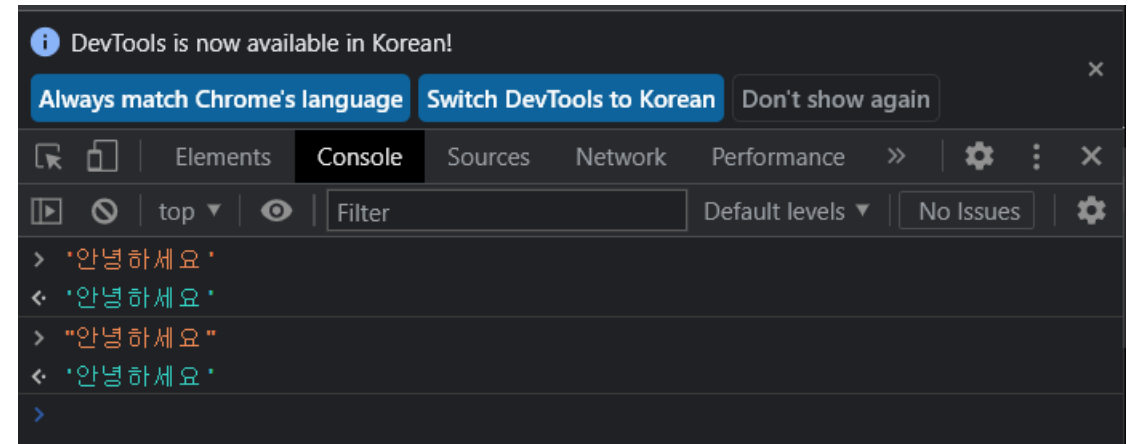
프로그램 개발의 첫걸음. 자료형과 변수 학습

SECTION 2-1 기본 자료형(1)

- 자료(data): 프로그래밍에서 프로그램이 처리할 수 있는 모든 것
- 자료형(data type): 자료 형태에 따라 나눠 놓은 것
 - 숫자(number), 문자열(string), 불(Boolean) 자료형
- 문자열 자료형
 - 문자열 만들기
 - 자바스크립트는 2가지 방법으로 문자열을 생성
 - 큰따옴표를 사용
 - 작은따옴표를 사용

```
> '안녕하세요'
'안녕하세요'
> '안녕하세요'
'안녕하세요'
```

콘솔 출력이 따옴표로 감싸져 있으면
이는 문자열을 의미



▲ 콘솔에서 실행한 결과

SECTION 2-1 기본 자료형(2)

- 문자열 자료형
 - 작은 따옴표 사용
 - 특수 문자
 - 이스케이프\ : 따옴표를 문자 그대로 사용해야 할 때
 - \n: 줄바꿈 \t: 탭 \\: 역슬래시(\) 그 자체를 의미
 - 문자열 연산자
 - 숫자 자료와 마찬가지로 문자열도 기호를 사용해서 연산 처리

```
> '가나다' + '라마' + '바사아' + '자차카타' + '파하'
'가나다라마바사아자차카타파하'
```

- 문자 선택 연산자
 - 문자열 내부의 문자 하나를 선택

```
> '안녕하세요'[0]
'안'
> '안녕하세요'[1]
'녕'
```

SECTION 2-1 기본 자료형(3)

- 문자열 자료형
 - 문자열 길이 구하기

```
> '안녕하세요'.length  
5  
> '자바스크립트'.length  
6  
> ''.length  
0
```

빈 문자열도 문자열이라는 것을 기억!!!

- Uncaught SyntaxError: Unexpected identifier(구문 오류)
 - 식별자가 예상하지 못한 위치에서 등장했다는 오류
 - 예를 들어 이스케이프 문자를 사용하지 않고 한 종류의 따옴표만 사용하면 다음과 같이 오류가 발생

오류

```
> 'This is 'string''
```

⊗ Uncaught SyntaxError: Unexpected identifier

SECTION 2-1 기본 자료형(4)

- 숫자 자료형
 - 소수점이 있는 숫자와 없는 숫자를 모두 같은 자료형으로 인식
 - 숫자 연산자

| 연산자 | 설명 | 연산자 | 설명 |
|-----|---------|-----|---------|
| + | 더하기 연산자 | * | 곱하기 연산자 |
| - | 빼기 연산자 | / | 나누기 연산자 |

| 연산자 | 설명 |
|-----|---------|
| % | 나머지 연산자 |

SECTION 2-1 기본 자료형(5)

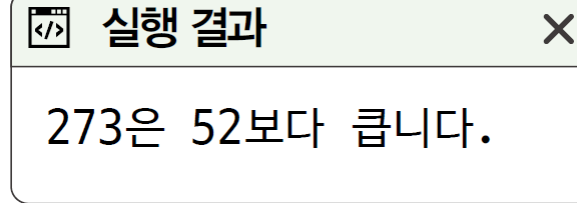
- 불 자료형
 - 자바스크립트에서는 참과 거짓 값을 표현할 때 불 자료형을 사용
 - 불 만들기
 - 비교 연산자

| 연산자 | 설명 |
|-----|------------------|
| === | 양쪽이 같다 (자료형 일치) |
| !== | 양쪽이 다르다 (자료형 일치) |
| > | 왼쪽이 더 크다 |
| < | 오른쪽이 더 크다 |
| >= | 왼쪽이 더 크거나 같다 |
| <= | 오른쪽이 더 크거나 같다 |

SECTION 2-1 기본 자료형(6)

- 불 자료형
 - 불 표현식 이해하기(소스 코드 2-1-1.html)

```
01 <script>
02  if (273 < 52) {
03    alert('273은 52보다 작습니다.')
04  }
05  if (273 > 52) {
06    alert('273은 52보다 큼니다.')
07  }
08 </script>
```



- 불 부정 연산자
 - 논리 부정 연산자는 ! 기호를 사용하며 참을 거짓으로, 거짓을 참으로 바꿈
- 불 논리합/논리곱 연산자

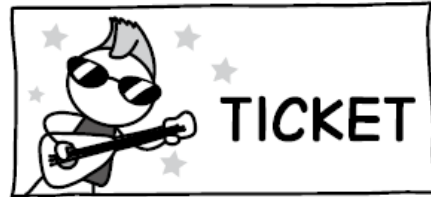
| 연산자 | 설명 |
|-----|---------|
| && | 논리곱 연산자 |
| | 논리합 연산자 |

SECTION 2-1 기본 자료형(7)

- 불 자료형

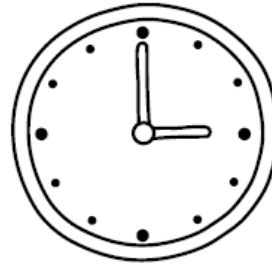
- 논리 연산자의 활용
- && 연산자

- 조건: "티켓을 1장만 구매하면서 오후 3시 이후부터"



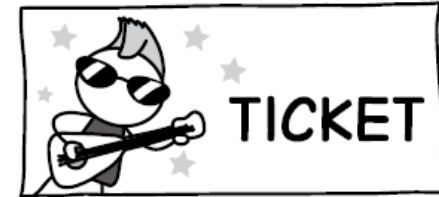
티켓 1장 이하

&&



오후 3시 이후

=



티켓 구매 가능

- || 연산자

- 조건: "우리카드나 신한카드로 결제하면 10% 할인"



우리카드

||



신한카드

=

D.C
10%
할인

SECTION 2-1 기본 자료형(8)

- 자료형 검사
 - typeof 연산자

```
> typeof('문자열')  
'string' ← 문자열을 의미  
typeof(273)  
'number' ← 숫자를 의미  
> typeof(true)  
'boolean'
```

- typeof 연산자는 결과로 string, number, boolean, undefined, function, object, symbol, bigint라는 8가지 중에 하나를 출력

[좀 더 알아보기]

- 템플릿 문자열은 백틱(`) 기호로 감싸 만들
 - 문자열 내부에 `\${...}` 기호를 사용하여 표현식을 넣으면 표현식이 문자열 안에서 계산됨

```
> console.log(`표현식 273 + 52의 값은 ${273 + 52}입니다...!`)
표현식 273 + 52의 값은 325입니다...!
```

- == 연산자와 != 연산자 (이중 등호)
 - '값이 같은지'를 비교하는 연산자
 - 다음 코드들은 모두 true를 출력

```
> 1 == '1'
```

```
true
```

```
> false == '0'
```

```
true
```

```
> "" == []
```

```
true
```

```
> 0 == []
```

```
true
```

← 다음 코드는 자료형이 달라도 어떻게든 변환을 하고 나면 값이 같아지므로 true

← false가 0으로, '0'이 0으로 변환된 뒤에 비교

← 빈 문자열은 false, 비어있는 배열 []는 false로 변환된 뒤에 비교

← 0은 false, 비어있는 배열 []는 false로 변환된 뒤에 비교



[요점 정리]

- 4가지 키워드로 정리하는 핵심 포인트
 - 자료형이란 자료의 종류를 의미
 - 문자를 표현할 때는 문자열 자료형을 사용
 - 숫자를 표현할 때는 숫자 자료형을 사용
 - 참과 거짓을 표현할 때는 불 자료형을 사용

SECTION 2-2 상수와 변수(1)

• 상수

- 상수를 만드는 과정을 '선언'이라고 표현하고, `const` 키워드로 다음과 같이 선언

```
const 이름 = 값
```

- 코드 예시: 3.141592라는 숫자 자료를 `pi`라는 이름으로 선언한다면 다음과 같이 코드를 작성

```
> const pi = 3.141592  
undefined  
> pi  
3.141592  
> const r = 10  
undefined  
> 2 * pi * r  
62.83184  
> pi * r * r  
314.1592
```

→ `pi`라는 이름의 상수를 선언하고, 3.141592라는 값을 할당

→ 앞서 선언한 상수 이름을 입력하면 해당 값을 사용 가능

→ 반지름이 10인 상수를 선언

→ 두 상수를 활용해 원의 둘레와 넓이를 구하기

// 반지름으로 원의 둘레 구하기

// 반지름으로 원의 넓이 구하기

SECTION 2-2 상수와 변수(2)

- 상수

- Identifier has already declared(구문 오류)

- 특정한 이름의 상수는 한 파일에서 한 번만 선언. 만약 같은 이름으로 상수를 한 번 더 선언하면 다음과 같은 오류를 발생

```
> const name = 'name'이라는 이름의 상수를 선언해볼게요.'
```

```
undefined
```

```
> const name = '한 번 더 선언해볼게요.'
```

“식별자 'name'은 이미 사용되고 있습니다”라는 오류

```
Uncaught SyntaxError: Identifier 'name' has already been declared
```

SECTION 2-2 상수와 변수(3)

• 상수

- Missing initializer in const declaration(구문 오류)

- 상수는 한 번만 선언할 수 있으므로 선언할 때 반드시 값을 함께 지정해줘야 함. 만약 상수를 선언할 때 값을 지정해주지 않는다면 다음과 같은 오류를 발생

```
const pi
```

```
Uncaught SyntaxError: Missing initializer in const declaration
```

- Assignment to constant variable(예외 처리)

- 한 번 선언된 상수의 자료는 변경할 수 없음. pi에 3.141592라는 값을 지정했다면 이값은 변하지 않으므로, 만약 값을 변경하면 다음과 같은 오류를 발생

```
> const name = 'name0이라는 이름의 상수를 선언해볼게요.'
```

```
undefined
```

```
> name = '그 값을 변경해볼게요.'
```

```
TypeError: Assignment to constant variable.
```

- 이 경우는 상수가 아닌 변수를 사용해야 함

SECTION 2-2 상수와 변수(4)

- 변수

- 변수를 만들 때는 let 키워드를 사용

```
> let pi = 3.141592
```

→ pi라는 이름의 변수를 선언하고, 3.141592라는 값을 지정

```
undefined
```

```
> pi
```

→ 변수 이름을 입력하면 해당 값을 사용할 수 있음

```
3.141592
```

```
> let r = 10
```

→ 반지름이 10인 변수를 선언

```
undefined
```

```
> 2 * pi * r
```

→ // 반지름으로 원의 둘레 구하기

```
62.83184
```

```
> pi * r * r
```

→ // 반지름으로 원의 넓이 구하기

```
314.1592
```

→ 두 변수를 활용해 원의 둘레와 넓이 구하기

- 변수의 값을 변경할 때는 변수 이름 뒤에 = 기호를 입력하고 값을 기입

```
변수 = 값
```

SECTION 2-2 상수와 변수(5)

- 변수

- Identifier has already been declared(구문 오류)

- 상수와 마찬가지로 특정한 이름의 변수는 한 파일에서 한 번만 선언. 만약 같은 이름으로 변수를 한 번 더 선언하면 다음과 같은 오류를 발생

```
<script>
  let name = 'name이라는 이름의 변수를 선언합니다'
  let name = '한 번 더 선언해볼게요'
</script>
```

Uncaught SyntaxError: Identifier 'name'
has already been declared



- 다른 이름의 식별자를 사용해서 변수를 선언하면 해결

```
<script>
  let nameA = 'name이라는 이름의 변수를 선언합니다'
  let nameB = '한 번 더 선언해볼게요'
</script>
```

SECTION 2-2 상수와 변수(6)

- 변수에 적용할 수 있는 연산자

- 복합 대입 연산자

| 복합 대입 연산자 | 설명 | 사용 예 | 의미 |
|-----------|--------------------|--------|---------|
| += | 기존 변수의 값에 값을 더하기 | a += 1 | a = a+1 |
| -= | 기존 변수의 값에 값을 빼기 | a -= 1 | a = a-1 |
| *= | 기존 변수의 값에 값을 곱하기 | a *= 1 | a = a*1 |
| /= | 기존 변수의 값에 값을 나누기 | a /= 1 | a = a/1 |
| %= | 기존 변수의 값에 나머지를 구하기 | a %= 1 | a = a%1 |

- 사용 예시

| | | |
|-------------------------------|---|------------------------|
| > let value = 10 undefined | → | value라는 변수를 10으로 선언 |
| > value += 10 20 | → | value에 10을 더하기 |
| > value 20 | → | value의 값은 10 + 10 = 20 |

SECTION 2-2 상수와 변수(7)

- 변수에 적용할 수 있는 연산자
 - 증감 연산자

| 증감 연산자 | 설명 |
|--------|----------------------|
| 변수++ | 기존의 변수 값에 1을 더하기(후위) |
| ++변수 | 기존의 변수 값에 1을 더하기(전위) |
| 변수-- | 기존의 변수 값에 1을 빼기(후위) |
| --변수 | 기존의 변수 값에 1을 빼기(전위) |

- 증감 연산자 예(1)

```
<script>
// 변수를 선언합니다.
let number = 10

// 연산자를 사용합니다.
number++

// 출력합니다.
alert(number)
</script>
```

- 증감 연산자 예(2)

```
<script>
// 변수를 선언합니다.
let number = 10

// 출력합니다.
alert(number++)
alert(++number)
alert(number--)
alert(--number)
</script>
```

SECTION 2-2 상수와 변수(8)

- undefined 자료형

- 상수와 변수로 선언하지 않은 식별자

- 다음 코드의 "abc"와 "그냥식별자"라는 식별자는 선언하지 않고 사용했으므로 undefined 자료형으로 나타남

```
> typeof(abc)
```

```
'undefined'
```

```
> typeof(그냥식별자)
```

```
'undefined'
```

→ 식별자를 한글로 입력했을 뿐

- 값이 없는 변수

- 변수를 선언하면서 값을 지정하지 않은 경우에 해당 식별자는 undefined 자료형이 됨

```
> let a
```

```
undefined
```

```
> typeof(a)
```

```
'undefined'
```

[요점 정리]

- 4가지 키워드로 정리하는 핵심 포인트
 - 상수는 변하지 않는 값을 저장하는 식별자. **const** 키워드를 사용해 선언
 - 변수는 변하는 값을 저장하는 식별자. **let** 키워드를 사용해 선언
 - 상수 또는 변수를 생성하는 것을 선언이라 함
 - 상수 또는 변수에 값을 넣는 것을 할당이라 함

SECTION 2-3 자료형 변환(1)

- 문자열 입력

- prompt(메시지 문자열, 기본 입력 문자열)
- prompt() 함수 매개변수의 역할 (소스 코드 2-3-1.html 참조)

```
<script>
// 상수를 선언합니다.
const input = prompt('message', '_default');
// 출력합니다.
alert(input);
</script>
```

prompt() 함수는 사용자로부터 내용을 입력받아서 사용

- 리턴(return): 함수를 실행한 후 값을 남기는 것(Chapter 5에서 학습)

SECTION 2-3 자료형 변환(2)

- 불 입력

- confirm() 함수를 사용하면 사용자에게 확인을 요구하는 메시지 창이 나타남

```
<script>
// 상수를 선언합니다.
const input = confirm('수락하시겠습니까?');

// 출력합니다.
alert(input);
</script>
```

- 사용자가 [확인] 버튼을 클릭하면 true를 리턴하고, [취소] 버튼을 클릭하면 false를 리턴

SECTION 2-3 자료형 변환(3)

• 숫자 자료형으로 변환하기

- 다른 자료형을 숫자 자료형으로 변환할 때는 Number() 함수를 사용

```
> Number('273')  
273  
> typeof(Number('273'))  
'number' → 자료형은 숫자
```

- 다른 문자가 들어있어서 숫자로 변환할 수 없는 문자열의 경우, NaN(Not a Number)라는 값을 출력
 - NaN은 자바스크립트에서 숫자이지만, 숫자로 나타낼 수 없는 숫자를 의미
- 숫자 연산자를 사용해 자료형 변환하기

```
> '52' - 0  
52  
> typeof('52' - 0)  
'number'  
> true - 0  
1  
> typeof(true - 0)  
'number'
```

```
> 1 + true  
2  
> 1 + false  
1
```

SECTION 2-3 자료형 변환(4)

- 문자열 자료형으로 변환하기

- 다른 자료형을 문자열 자료형으로 변환할 때는 String() 함수를 사용

| | | |
|------------------|---|----------------------|
| <hr/> | | |
| > String(52.273) | → | 숫자 자료형이 문자열 자료형으로 변환 |
| '52.273' | | |
| > String(true) | → | 불 자료형이 문자열 자료형으로 변환 |
| 'true' | | |
| > String(false) | | |
| 'false' | | |
| <hr/> | | |

- 문자열 연산자를 사용해 자료형 변환하기
 - 문자열 연결 연산자(+)를 사용

| | | |
|-------------|---|-------------------------|
| <hr/> | | |
| > 273 + "" | → | 빈 문자열을 연결해 문자열 자료형으로 변환 |
| '273' | | |
| > true + "" | | |
| 'true' | | |
| <hr/> | | |

SECTION 2-3 자료형 변환(5)

• 불 자료형으로 변환하기

- 다른 자료형을 불 자료형으로 변환할 때는 Boolean() 함수를 사용
 - 대부분의 자료는 불로 변환했을 때 true로 변환되나, 0, NaN, ""(빈 문자열), null, undefined라는 5개의 자료형은 false로 변환됨

| | |
|----------------|-----------------|
| > Boolean(0) | > Boolean(null) |
| false | false |
| > Boolean(NaN) | > let 변수 |
| false | undefined |
| > Boolean("") | > Boolean(변수) |
| false | false |

• 논리 부정 연산자를 사용해 자료형 변환하기

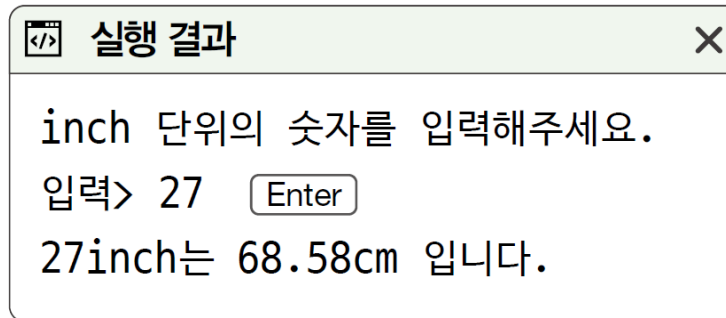
- Boolean() 함수를 사용하지 않고 논리 부정 연산자(!)를 사용해서 다른 자료형을 불 자료형으로 변환

```
> a = 1
1
> !a
false
> !!a
true
```

SECTION 2-3 자료형 변환(누적 예제)

- inch를 cm 단위로 변경하기

```
01 <script>
02  // 숫자를 입력
03  const rawInput = prompt('inch 단위의 숫자를 입력해주세요.')
04
05  // 입력받은 데이터를 숫자형으로 변경하고 cm 단위로 변경
06  const inch = Number(rawInput)
07  const cm = inch * 2.54
08
09  // 출력
10  alert(`${inch}inch는 ${cm}cm 입니다.`)
11 </script>
```



[요점 정리]

- 5가지 키워드로 정리하는 핵심 포인트
 - 사용자로부터 글자를 입력 받을 때는 `prompt()` 함수를 사용
 - 어떤 자료형의 값을 다른 자료형으로 변경하는 것을 자료형 변환이라고 함
 - 숫자 자료형으로 변환할 때 `Number()` 함수를 사용
 - 문자열 자료형으로 변환할 때 `String()` 함수를 사용
 - 불 자료형으로 변환할 때 `Boolean()` 함수를 사용