

02

기본 자료형과 연산자

목차

1. 기본 출력 방법
2. 기본 자료형, 변수, 상수
3. 연산자
4. 자료형 변환

기본 용어

■ 자바스크립트 코드 위치

- 자바스크립트 코드 위치 : head, body
- head태그 내부의 자바스크립트는 문서 로딩이 완료 안된 상태에서 실행되므로 body태그내의 문서 요소를 참조할 수 없습니다
- head태그 내부의 자바스크립트 코드는 함수 정의, 이벤트 핸들러 정의등이 선언됩니다

기본 용어

■ 표현식과 문장

■ 표현식

```
273
```

```
10 + 20 + 30 * 2
```

```
"JavaScript Programming"
```

- 문장 : 표현식이 하나 이상 모일 경우, 마지막에 종결 의미로 세미콜론(;)
- 문장을 구분하는 ;는 가독성을 위해서 사용합니다.(생략가능)

기본 용어

■ 키워드

- 자바스크립트에서 특별한 의미가 부여된 단어

break	else	instanceof	true
case	false	new	try
catch	finally	null	typeof
continue	for	return	var
default	function	switch	void
delete	if	this	while
do	in	throw	with
let	const		

abstract	enum	int	short
boolean	export	interface	static
byte	extends	long	super
char	final	native	synchronized
class	float	package	throws
const	goto	private	transient
debugger	implements	protected	volatile
double	import	public	

기본 용어

■ 식별자

- 이름을 붙일 때 사용하는 단어, 변수와 함수 이름 등으로 사용

- 키워드를 사용 안됨
- 특수 문자는 _와 \$만 허용
- 숫자로 시작하면 안됨
- 공백은 입력하면 안됨
- 대소문자 구별

alpha	break
alpha10	273alpha
_alpha	has space
\$alpha	
AlPha	
ALPHA	

○ X

기본 용어

■ 식별자 사용 규칙

- 생성자 함수의 이름은 항상 대문자로 시작
- 변수, 함수, 속성, 메소드의 이름은 항상 소문자로 시작
- 여러 단어로 된 식별자는 각 단어의 첫 글자를 대문자로 함

구분	단독으로 사용	다른 식별자와 사용
식별자 뒤에 괄호 없음	변수 또는 상수	속성
식별자 뒤에 괄호 있음	함수	메소드

<code>alert('Hello World')</code>	⇒ 함수
<code>Array.length</code>	⇒ 속성
<code>input</code>	⇒ 변수 또는 상수
<code>prompt('Message', 'Defstr')</code>	⇒ 함수
<code>Math.PI</code>	⇒ 속성
<code>Math.abs(-273)</code>	⇒ 메소드

기본 용어

■ 주석

- 프로그램의 진행에 영향을 주지 않는 코드

방법	표현
한 줄 주석 처리	// 주석
여러 줄 주석 처리	/* 주석 주석 */

```
// 주석은 코드의 실행에 영향을 주지 않습니다.  
/*  
console.log("JavaScript Programming")  
console.log("JavaScript Programming")  
console.log("JavaScript Programming")  
*/
```


출력

■ 출력 메소드

- `console` : 웹 어플리케이션 개발시 디버깅등을 위한 출력 객체
- `log()` : 로깅 함수로 문자열, 숫자, 객체, 배열 등 모든 타입의 데이터를 문자열 형식으로 데이터를 출력할 수 있습니다
- `dir()` : 객체의 속성과 구조를 계층적으로 탐색하여 출력합니다.

```
console.log(message, ...optionalParams)
```

```
const user = { name: 'Alice', age: 25 };  
console.log('Hello, world!');  
console.log('User:', user);  
console.log('Age:', user.age);
```

```
console.dir(obj, [options])
```

```
console.dir(window);  
console.dir(document);
```

출력

■ 출력 메소드

- `document.write()` : HTML 문서의 콘텐츠를 변경하는 데 사용됩니다
- `window.alert()` : 경고 대화 상자를 표시합니다.

```
document.write(content)
```

```
<body>  
  <script>  
    document.write('<h1>Hello, World!</h1>');  
    document.write('<p>This is a paragraph.</p>');  
  </script>  
</body>
```

```
window.alert(message)
```

```
<script>  
  window.alert('Hello, this is an alert!');  
</script>
```

입력

■ 입력 메소드

- `prompt()` : 사용자에게 텍스트 입력을 요청하는 대화 상자를 표시합니다.
- `confirm()` : 사용자가 특정 작업을 확인하거나 취소할 수 있는 대화 상자를 표시합니다

```
prompt(message, default)
```

```
<body>  
  <script>  
    let name = prompt('What is your name?', 'Guest');  
  </script>  
</body>
```

```
window.alert(message)
```

```
<script>  
  let result = confirm('Do you really want to delete this item?');  
</script>
```

입력

● 입력

- 웹 브라우저에서 작동하는 자바스크립트 : `prompt()` 이름의 함수를 받음
- Node.js에서 작동하는 자바스크립트 : 단순한 코드로 입력을 받을 수 없음

```
// 모듈을 추출합니다.
const repl = require('repl');

// 입력을 시작합니다.
repl.start({
  prompt: "<입력 때 앞에 출력할 문자열>",
  eval: (command, context, filename, callback) => {
    // 입력(command)을 받았을 때 처리를 수행합니다.

    // 처리 완료
    callback();
  }
});
```

입력

- 입력

```
// 모듈을 추출합니다.
const repl = require('repl');

// 입력을 시작합니다.
repl.start({
  prompt: "숫자 입력> ",
  eval: (command, context, filename, callback) => {
    // 입력(command)을 받았을 때 처리를 수행합니다.
    let number = Number(command);

    // 입력이 숫자인지 확인합니다.
    if (isNaN(number)) {
      console.log("숫자가 아닙니다.");
    } else {
      console.log("숫자입니다.");
    }

    // 처리 완료
    callback();
  }
});
```

기본 자료형

■ 기본 자료형

자료형	설명
string	텍스트 데이터를 저장하고 조작하는 데 사용됩니다. 작은따옴표('"'), 큰따옴표('\"'), 또는 백틱(`)으로 감쌀 수 있습니다.
number	정수 및 부동 소수점 숫자를 포함하는 숫자를 나타내는 자료형
boolean	논리적 참(true) 또는 거짓(false)을 나타내는 자료형
object	키-값 쌍의 컬렉션을 나타내는 복합 자료형 여러 속성 및 메서드를 가질 수 있습니다.
function	코드 블록을 캡슐화하여 나중에 실행할 수 있는 특별한 자료형 함수는 일급 객체로, 변수에 할당하거나 다른 함수에 인수로 전달할 수 있습니다
undefined	값이 할당되지 않은 변수를 나타냅니다. 변수를 선언했으나 초기화하지 않은 자료형

자료형 검사

- 자료형 확인 연산자

표 2-16 자료형 확인 연산자

연산자	설명
typeof	해당 변수의 자료형을 추출합니다.

강제 자료형 변환

강제 자료형 변환 함수

함수	설명
Number()	숫자로 자료형 변환합니다.
String()	문자열로 자료형 변환합니다.
Boolean()	불로 자료형 변환합니다.

■ Number() 함수와 NaN

```
console.log(Number("52"));  
console.log(Number("52.273"));  
console.log(Number(true));  
console.log(Number(false));  
console.log(Number("안녕하세요"));
```


기본 자료형

■ 이스케이프 문자

- 문자열 내에서 특별한 의미를 갖는 문자나 특수 문자를 포함하기 위해 이스케이프 문자를 사용합니다.
- 백슬래시(\)로 시작

Escape문자	설명
\n	새 줄(줄 바꿈)을 나타냅니다. 문자열을 여러 줄로 나누고자 할 때 사용
\t	탭 문자를 나타냅니다. 문자열 내에 탭 공간을 삽입할 때 사용
\\	백슬래시 자체를 나타냅니다. 백슬래시를 문자열에 포함시키기 위해 사용
\"	큰따옴표를 나타냅니다. 큰따옴표를 포함한 문자열을 정의할 때 사용
\'	작은따옴표를 나타냅니다. 작은따옴표를 포함한 문자열을 정의할 때 사용
\r	캐리지 리턴을 나타냅니다. 주로 새 줄과 함께 사용되며, 텍스트의 줄의 시작으로 커서를 이동시킵니다.
\b	백스페이스 문자를 나타냅니다. 이전 문자를 삭제합니다.

```
console.log("이름\t나이");  
console.log("안녕\n하세요");  
console.log("\\\\");
```

변수

var 변수	let 변수
같은 이름의 변수를 중복 선언할 수 있으며, 재할당도 가능합니다.	같은 스코프 내에서 중복 선언 할 수 없습니다.
함수 스코프를 따릅니다. (함수 내에서 선언된 변수는 해당 함수 내에서만 유효하며 함수 외부에서는 접근할 수 없습니다. 블록 스코프를 무시합니다. 블록 {} 내에서 선언된 var 변수는 블록 외부에서도 접근 가능합니다.	블록 스코프를 따릅니다. 블록 스코프는 중괄호 {}로 둘러싸인 코드 블록 내에서만 변수가 유효하며, 블록 외부에서는 접근할 수 없습니다.
변수 호이스팅이 발생하여 변수 선언이 해당 스코프의 최상위로 끌어올려집니다. 하 할당은 원래 위치에서 이루어집니다.	
초기값이 할당되기 전에 접근하면 'undefined'가 반환됩니다	
전역 객체(window 객체 또는 global 객체)의 속성이 됩니다.	let으로 선언된 전역 변수는 전역 객체의 속성이 되지 않습니다.

```
let 식별자;
```

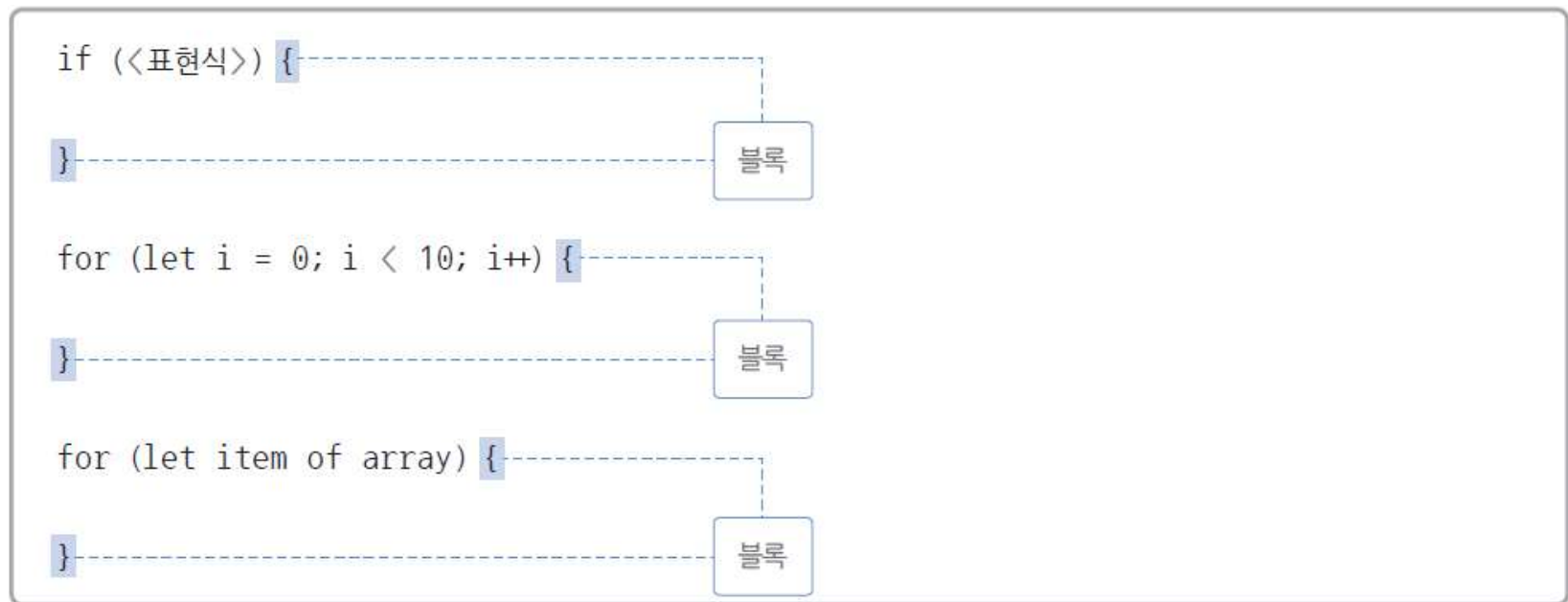
Scope

● 스킵프 Scope

- 변수를 사용할 수 있는 범위
- 스킵프 == 블록

■ 블록

- 중괄호로 둘러싸는 부분



Scope

- 블록 내부에 선언된 변수는 해당 변수 내부에서만 사용가능

```
{  
  let a = 10;  
}  
  
console.log(a);
```

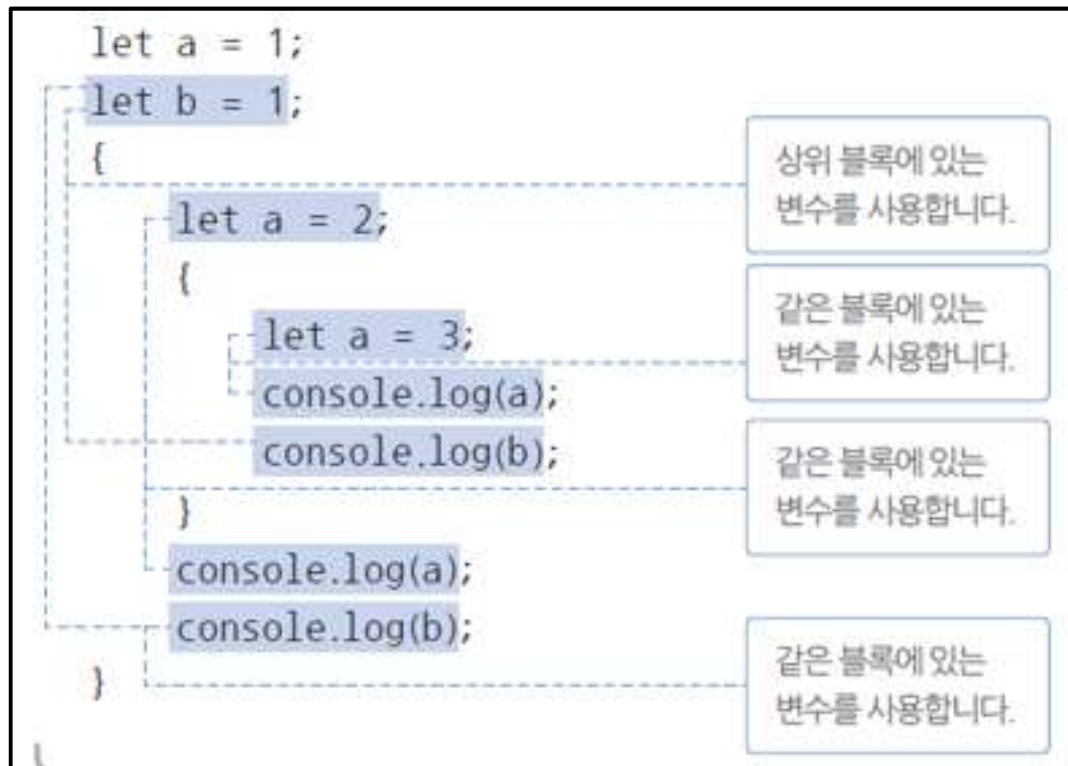
- 반복문에 활용된 변수는 해당 블록에 있으므로 외부에서 활용할 수 없음

```
for (let i = 0; i < 3; i++) {  
  console.log(i);  
}  
  
console.log(i);
```

변수 i는 해당 블록 외부에서
사용할 수 없습니다.

Scope

- Scope



Scope

- 과거 자바스크립트에서는 var 키워드를 사용했으나 스코프 문제가 많아 현재는 let 키워드를 사용함

```
{  
  let a = 10;  
}  
  
console.log(a);
```

```
{  
  var a = 10;  
}  
  
console.log(a);
```

Scope

- 호이스팅 Hoisting

- 해당 블록에서 사용할 변수를 미리 확인해서 실행 전에 생성하는 작업

```
let a = 1;  
{  
  console.log(a);  
  let a = 2;  
}
```

기본 자료형

■ 문자열 연산자

연산자	설명
문자열[숫자]	문자열 선택 연산자
+	문자열 연결 연산자

```
console.log("안녕하세요"[0]);  
console.log("안녕하세요"[1]);  
console.log("안녕하세요"[3]);
```

```
> `올해는 ${new Date().getFullYear()}년입니다.`
```


기본 자료형

■ 비교 연산자

연산자	설명
==	같습니다.
!=	다릅니다.
>	왼쪽 피연산자가 큼니다.
<	오른쪽 피연산자가 큼니다.
>=	왼쪽 피연산자가 크거나 같습니다.
<=	오른쪽 피연산자가 크거나 같습니다.

기본 자료형

■ 논리 연산자

연산자	설명
!	논리 부정 연산자
	논리합 연산자
&&	논리곱 연산자

연산자

■ 대입 연산자

숫자에 적용하는 복합 대입 연산자

연산자	설명
<code>+=</code>	숫자 덧셈 후 대입 연산자
<code>-=</code>	숫자 뺄셈 후 대입 연산자
<code>*=</code>	숫자 곱셈 후 대입 연산자
<code>/=</code>	숫자 나눗셈 후 대입 연산자

문자열에 적용하는 복합 대입 연산자

연산자	설명
<code>+=</code>	문자열 연결 후 대입 연산자

증감 연산자

■ 증감 연산자

표 2-15 증감 연산자

연산자	설명
변수++	기존 변수 값에 1을 더합니다(후위).
++변수	기존 변수 값에 1을 더합니다(전위).
변수--	기존 변수 값에서 1을 뺍니다(후위).
--변수	기존 변수 값에서 1을 뺍니다(전위).

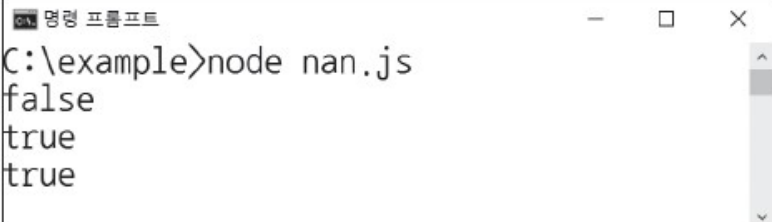
강제 자료형 변환

- '숫자로 변환할 수 없는 문자열'을 Number() 함수로 변환하면 'NaN'을 출력
- NaN(Not a Number)은 '숫자 자료형이지만 숫자가 아닌 것'을 의미
- NaN의 특징
 - NaN과 NaN의 비교연산은 false
 - NaN인지 확인할 때는 isNaN() 함수를 사용

코드 2-24 Not a Number

nan.js

```
// NaN 변수를 만듭니다.  
let nan = Number("안녕하세요");  
  
// NaN끼리 비교합니다.  
console.log(nan == nan);  
console.log(nan != nan);  
  
// isNaN() 함수로 NaN인지 확인합니다.  
console.log(isNaN(nan));
```



```
명령 프롬프트  
C:\example>node nan.js  
false  
true  
true
```

강제 자료형 변환

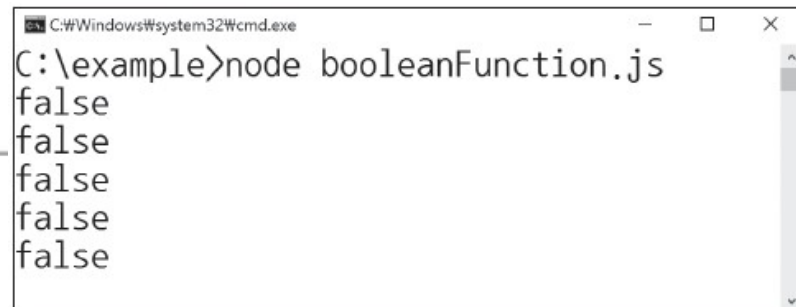
■ Boolean() 함수

- Boolean() 함수를 사용하면 5개의 요소는 false로 변환
- 0, NaN, ""[빈 문자열], null, undefined

코드 2-25 Boolean() 함수

booleanFunction.js

```
// 변수를 선언합니다.  
let nan = Number("안녕하세요");  
let undefinedVariable;  
  
// Boolean() 함수를 사용합니다.  
console.log(Boolean(0));  
console.log(Boolean(nan));  
console.log(Boolean(""));  
console.log(Boolean(null));  
console.log(Boolean(undefinedVariable));
```



```
C:\Windows\system32\cmd.exe  
C:\example>node booleanFunction.js  
false  
false  
false  
false  
false
```

강제 자료형 변환

■ 숫자와 문자열 자료형 자동 변환

- 숫자와 문자열에 '+' 연산자를 적용하면 자동으로 숫자가 문자열로 변환

```
console.log(52 + 273);  
console.log("52" + 273);  
console.log(52 + "273");  
console.log("52" + "273");
```

자동 자료형 변환

■ 논리 자료형의 형 변환

```
// 변수를 선언합니다.  
let nan = Number("안녕하세요");  
let undefinedVariable;  
  
// 부정 연산자를 두 번 사용합니다.  
console.log(!!0);  
console.log(!!nan);  
console.log(!!"");  
console.log(!!null);  
console.log(!!undefinedVariable);
```


일치 연산자

■ 일치 연산자

- 값과 자료형을 함께 비교

일치 연산자

연산자	설명
===	자료형과 값이 같은지 비교합니다.
!==	자료형과 값이 다른지 비교합니다.

```
console.log(`52 == "52": ${52 == "52"}`);  
console.log(`52 === "52": ${52 === "52"}`);  
console.log();  
console.log(`${0} == "" : ${0 == ""}`);  
console.log(`${0} === "" : ${0 === ""}`);
```

상수

■ 상수

- 상수 : '항상 같은 수'라는 의미, 변수와 반대되는 개념
- `const` : 상수constant를 만드는 키워드
- 변하지 않을 대상에 상수를 적용

```
// 상수를 선언합니다.  
const constant = "변경할 수 없어요";  
constant = "";  
  
// 출력합니다.  
console.log(constant);
```