



클라우드보안 과정:

자바스크립트 기본

성결대학교 정보통신공학과

정복래

brjung@sungkyul.ac.kr

학습목표

- 자바스크립트에서 사용하는 기본 용어를 이해합니다.
- 기본 출력 방법을 익힙니다.
- 기본 자료형, 변수, 상수를 배우고 관련된 연산자의 사용 방법을 익힙니다.
- 자료형 변환 방법을 익힙니다.

Contents

내용

- 기본 용어
- 출력
- 기본 자료형
- 변수
- 복합 대입 연산자
- 증감 연산자
- 자료형 검사
- undefined 자료형
- 강제 자료형 변환
- 자동 자료형 변환
- 일치 연산자
- 상수

1. 기본 용어

■ 표현식과 문장

■ 표현식

```
273
```

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

```
"JavaScript Programming"
```

- 문장 : 표현식이 하나 이상 모일 경우, 마지막에 종결 의미로 세미 콜론(;)
- 프로그램 : 문장이 모이면 프로그램이 됨

```
273;
```

```
10 + 20 + 30 + 2;
```

```
let name = "윤" + "인" + "성"
```

```
console.log("Hello World...!")
```

1. 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

```
alpha  
alpha10  
_alpha  
$alpha  
AlPha  
ALPHA
```

O

```
break  
273alpha  
has space
```

X

그림 2-1 식별자로 사용할 수 있는 단어와 사용할 수 없는 단어

1. 기본 용어

■ 식별자

■ 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

```
will out      // willOut  
will return   // willReturn  
i am a boy    // iAmABoy
```

1. 기본 용어

표 2-2 자바스크립트 식별자의 종류

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

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

1. 기본 용어

■ 주석

- 프로그램의 진행에 영향을 주지 않는 코드

표 2-3 주석 처리 방법

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

코드 2-1 주석

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


2. 출력

■ 출력 메소드

- console 객체의 log() 메소드 사용 : console.log() 메소드

```
console.log("문자열")
```

그림 2-2 console.log() 메소드의 형태

- [예제 2-1] 자바스크립트의 기본 출력
 - 문자열 'JavaScript Programming' 출력하기

코드 2-2 기본 출력 예제

output.js

```
console.log("JavaScript Programming");
```

실행 결과

JavaScript Programming

2. 출력

■ REPL을 사용한 출력

- 곧바로 문장을 입력해서 출력

```
> "안녕" + "하세요"
```

```
'안녕하세요'
```

```
> 52 + 273
```

```
325
```

```
>
```

```
> 문장
```

```
출력_결과
```

3. 기본 자료형

■ 숫자

■ 가장 기본적인 자료형

코드 2-3 숫자 생성

```
console.log(52);  
console.log(52.271);
```

표 2-4 기본적인 사칙 연산자

연산자	설명
+	덧셈 연산자
-	뺄셈 연산자
*	곱셈 연산자
/	나눗셈 연산자

```
> 52 + 273  
325
```

3. 기본 자료형

■ 연산자 우선순위

코드 2-4

연산자 우선순위

```
console.log(5 + 3 * 2);
```

■ 나머지 연산자

표 2-5 나머지 연산자

연산자	설명
%	나머지 연산자

```
> 10 % 5  
0  
> 7 % 3  
1
```

3. 기본 자료형

■ [예제 2-2] 숫자와 연산자

코드 2-5

숫자와 연산자

numberOperator.js

```
console.log(1 + 2);  
console.log(1 - 2);  
console.log(1 * 2);  
console.log(1 / 2);  
console.log(1 % 2);
```

실행 결과

```
3  
-1  
2  
0.5  
1
```

3. 기본 자료형

■ 문자열

- 문자의 집합
- 문자열 생성시 큰따옴표나 작은따옴표를 사용

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

```
> console.log("This is 'String'")  
This is 'String'  
undefined  
> console.log('This is "String"')  
This is "String"  
undefined
```

3. 기본 자료형

- 이스케이프 문자
 - 따옴표를 문자 그대로 사용 가능
 - 문자열 줄바꿈 할 경우 사용

```
> console.log("This is \"String\"")
This is "String"
undefined
> console.log('This is \'String\'')
This is 'String'
undefined
```

```
> console.log("동해물과 백두산이\n마르고 닳도록")
동해물과 백두산이
마르고 닳도록
undefined
```

표 2-6 자주 사용하는 이스케이프 문자

이스케이프 문자	설명
\t	수평 탭
\n	줄바꿈
\'	작은따옴표
\"	큰따옴표
\\	역슬래시

3. 기본 자료형

- [예제 2-3] 이스케이프 문자
 - [표 2-6]에서 \t, \n, \\ 를 사용하기

코드 2-8

이스케이프 문자

escape.js

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

실행 결과

```
이름    나이  
안녕  
하세요  
\\
```


3. 기본 자료형

■ 문자열 합하기

표 2-7 문자열 연결 연산자

연산자	설명
+	문자열 연결 연산자

■ [예제 2-4] 문자열 연결 연산자

코드 2-9

문자열 연결 연산자

stringConcat.js

```
console.log("가나다" + "라마" + "바사아" + "자차카타" + "파하");
```

실행 결과

가나다라마바사아자차카타파하

3. 기본 자료형

■ 문자 선택 연산자

표 2-8 문자 선택 연산자

연산자	설명
문자열[숫자]	문자 선택 연산자

■ [예제 2-5] 문자 선택 연산자

- '안녕하세요' 문자열의 0번째, 1번째, 3번째에 있는 문자를 선택

코드 2-10 문자 선택

characterSelector.js

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

실행 결과

안
녕
세

3. 기본 자료형

- 템플릿 문자열

```
> `안녕하세요`  
'안녕하세요'
```

```
> `52 + 273 = ${52 + 273}`  
'52 + 273 = 325'
```

```
> `올해는 ${new Date().getFullYear()}년입니다.`  
'올해는 2022년입니다.'
```

3. 기본 자료형

■ 불

- 참과 거짓의 표현 : true와 false

```
> true  
true  
> false  
false
```

```
52 < 273  
52 > 273
```

- 비교 연산자

표 2-9 비교 연산자

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

3. 기본 자료형

■ [예제 2-6] 불과 비교 연산자

코드 2-11 비교 연산자

relationalOperator.js

```
console.log(52 < 273);  
console.log(52 > 273);  
console.log("하마" < "가방");
```

실행 결과

```
true  
false  
false
```

표 2-10 논리 연산자

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

3. 기본 자료형

■ [예제 2-7] 논리 부정 연산자

코드 2-11

비교 연산자

relationalOperator.js

```
console.log(52 < 273);  
console.log(52 > 273);  
console.log("하마" < "가방");
```

실행 결과

```
true  
false  
false
```

3. 기본 자료형

■ 논리합 연산자 (이항 연산자)

표 2-11 논리합 연산자

왼쪽 피연산자	오른쪽 피연산자	결과
true	true	true
true	false	true
false	true	true
false	false	false

■ 논리곱 연산자 (이항 연산자)

표 2-12 논리곱 연산자

왼쪽 피연산자	오른쪽 피연산자	결과
true	true	true
true	false	false
false	true	false
false	false	false

3. 기본 자료형

- 논리 연산자가 많이 사용되는 부분은 '범위 판단'
 - 예) 비교 연산자의 잘못된 사용

```
30 > 20 > 10
```

```
30 > 20 > 10
(30 > 20) > 10
true > 10
1 > 10
false
```

- 비교 연산자가 여러 개 있을 때 왼쪽부터 차례대로 연산하면서 발생하는 문제

3. 기본 자료형

- 논리 연산자의 사용

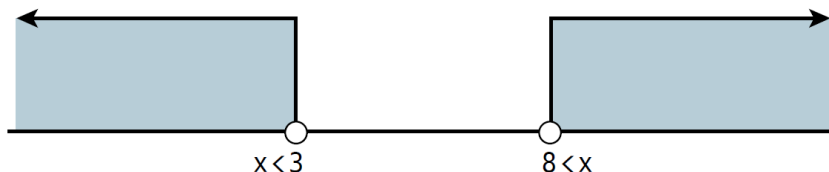


그림 2-3 ' $x < 3$ 또는 $8 < x$ '의 범위

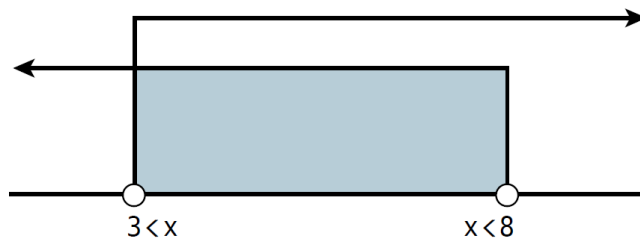


그림 2-4 ' $3 < x$ 그리고 $x < 8$ '의 범위

3. 기본 자료형

- [예제 2-8] 불과 논리 연산자
 - 현재 시간을 구하는 방법

코드 2-13 비교 연산자

logicalOperator.js

```
let hours = (new Date()).getHours();  
  
console.log(hours < 3 || 8 < hours);  
console.log(3 <= hours && hours <= 8);
```

현재 시간을 구하는 코드입니다.

실행 결과

false

true

4. 변수

- 변수 : 값을 저장할 때 사용하는 식별자, 변수 선언 후 변수에 값을 할당
 - 변수 선언
 1. 변수를 선언합니다.
 2. 변수에 값을 할당합니다.
 - 변수 pi를 선언

```
> let pi;  
undefined
```

- 변수 pi에 값을 할당

```
> let pi;  
undefined  
> pi = 3.14159265;  
undefined
```

4. 변수

- 변수 초기화

```
> let pi = 3.14159265;  
undefined
```

- 변수 활용

```
> let pi = 3.14159265;  
undefined  
> console.log(pi);  
3.14159265  
undefined
```

4. 변수

- [예제 2-9] 변수 기본 사용 방법
 - 반지름과 파이 값을 저장하고, 원의 둘레와 넓이를 계산

코드 2-14 변수 기본 사용 방법

variable.js

```
let pi = 3.14159265;  
let radius = 10;  
  
console.log(`둘레: ${2 * pi * radius}`);  
console.log(`넓이: ${pi * radius * radius}`);
```

실행 결과

```
둘레: 62.831853  
넓이: 314.159265
```

5. 복합 대입 연산자

- 변수에 사용할 수 있는 몇 개의 특별한 연산자가 존재
 - $a += 10$ 는 $a = a + 10$ 과 결과가 같음

표 2-13 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

표 2-14 문자열에 적용하는 복합 대입 연산자

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

5. 복합 대입 연산자

■ [예제 2-10] 숫자와 관련된 복합 대입 연산자

코드 2-15 숫자 관련 복합 대입 연산자

assignmentOperator1.js

```
let output = 0;  
output += 52;  
output += 273;  
output += 103;  
  
console.log(output);
```

실행 결과

428

코드 2-16 숫자와 관련된 복합 대입 연산자 풀어쓰기

assignmentOperator2.js

```
let output = 0;  
output = output + 52;  
output = output + 273;  
output = output + 103;  
  
console.log(output);
```

5. 복합 대입 연산자

- [예제 2-11] 문자열과 관련된 복합 대입 연산자

코드 2-17

문자열과 관련된 복합 대입 연산자

stringAssignmentOperator.js

```
let output = "hello ";  
output += "world ";  
output += "!";  
  
console.log(output);
```

실행 결과

hello world !

```
let output = "hello ";  
output = output + "world ";  
output = output + "!";  
  
console.log(output);
```


6. 증감 연산자

■ [예제 2-12 증감 연산자]

- 변수 number를 초기화하고, ++ 연산자와 -- 연산자를 사용
- 각 연산자에서 변수 값이 1만큼 변경됨

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).

코드 2-18 증감 연산자

incrementOperator.js

```
let number = 10;  
  
number++;  
console.log(number);  
  
number--;  
console.log(number);
```

실행 결과

11
10

6. 증감 연산자

- [예제 2-13] 증감 연산자의 전위와 후위

코드 2-19 증감 연산자의 전위와 후위

incrementFront.js

```
// incrementBack.js
let number = 10;

console.log(number);
console.log(number++);
console.log(number--);
console.log(number);
```

```
// incrementFront.js
let number = 10;

console.log(number);
console.log(++number);
console.log(--number);
console.log(number);
```

실행 결과

10
10
11
10

실행 결과

10
11
10
10

6. 증감 연산자

- 후위는 문장을 실행하기 전에 값을 변경하라는 의미
→ `console.log (++number)` 코드는
`console.log (number)`를 실행하고 변수 `number`에 1을 더함
- 이는 [코드 2-20]과 같음

코드 2-20 후위 증감 연산자

```
let number = 10;  
  
console.log(number);  
console.log(number); number += 1;  
console.log(number); number -= 1;  
console.log(number);
```

6. 증감 연산자

- 아래 두 코드 실행 : 차례대로 10, 12, 12, 10을 출력

코드 2-21 증감 연산자 이해도 확인

```
let number = 10;

console.log(number++);
console.log(++number);
console.log(number--);
console.log(--number);
```

코드 2-22 여러 줄로 나누어 적은 증감 연산자

```
let number = 10;

console.log(number);
number++;
number++;
console.log(number);
console.log(number);
number--;
number--;
console.log(number);
```

7. 자료형 검사

■ 자료형 확인 연산자

표 2-16 자료형 확인 연산자

연산자	설명
typeof	해당 변수의 자료형을 추출합니다.

```
> typeof 10
'number'
> typeof "문자열"
'string'
```

■ 보통 연산자 뒤에 괄호를 붙임

```
> typeof(10)
'number'
> typeof("문자열")
'string'
```

7. 자료형 검사

■ 자바스크립트의 여섯 가지 자료형

```
> // 문자열
> typeof('String')
'string'

> // 숫자
> typeof(273)
'number'

> // 불
> typeof(true)
'boolean'

> // 함수
> typeof(function () { })
'function'

> // 객체
> typeof({})
'object'

> // undefined
> let alpha
> typeof(alpha)
'undefined'
```

8. undefined 자료형

■ undefined 자료형

- 변수를 선언했으나 초기화하지 않은 자료형

```
> let a  
undefined  
> typeof(a)  
"undefined"  
  
> typeof(b)  
"undefined"
```

초기화하지 않은 변수입니다.

선언하지 않았던 식별자입니다.

9. 강제 자료형 변환

■ 강제 자료형 변환

■ 강제 자료형 변환 함수

표 2-17 강제 자료형 변환 함수

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

■ String() 함수

```
> String(52)
"52"
> String(273)
"273"

> String(true)
"true"
> String(false)
"false"
```

숫자는 그대로 문자열로 바뀝니다.

불도 그대로 문자열로 바뀝니다.

9. 강제 자료형 변환

■ Number() 함수와 NaN

- [예제 2-14] Number() 함수

코드 2-23

Number() 함수

numberFunction.js

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

숫자로 변환할 수 없는 문자열

실행 결과

```
52  
52.273  
1  
0  
NaN
```

9. 강제 자료형 변환

■ NaN

- '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
 - NaN은 무조건적으로 다름
 - NaN인지 확인할 때는 isNaN() 함수를 사용

9. 강제 자료형 변환

- [예제 2-15] NaN(Not a Number)

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));
```

실행 결과

```
false  
true  
true
```

9. 강제 자료형 변환

■ Boolean() 함수

- Boolean() 함수를 사용하면 다음 요소는 false로 변환됨
 - 0
 - NaN
 - "" (빈 문자열)
 - null
 - undefined
- 이 외에는 모두 true로 변환!!

9. 강제 자료형 변환

■ [예제 2-16] Boolean() 함수

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));
```

실행 결과

```
false  
false  
false  
false  
false
```

10. 자동 자료형 변환

■ 숫자와 문자열 자료형 자동 변환

- 숫자와 문자열에 '+' 연산자를 적용하면 자동으로 숫자가 문자열로 변환
- [예제 2-17] 숫자와 문자열 자료형 변환 – 덧셈 연산자

코드 2-26

숫자와 문자열 자료형 변환 (1)

numberToString.js

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

실행 결과

```
325  
52273  
52273  
52273
```

10. 자동 자료형 변환

■ 숫자와 문자열 자료형 자동 변환

- 숫자와 문자열에 '+' 연산자를 적용하면 자동으로 숫자가 문자열로 변환
- [예제 2-18] 숫자와 문자열 자료형 변환 – 다른 연산자

코드 2-27 숫자와 문자열 자료형 변환 (2)

```
console.log("52" - 273)
console.log("52" * 273)
console.log("52" / 273)
console.log("52" % 273)
```

실행 결과

```
-221
14196
0.19047619047619047
52
```

10. 자동 자료형 변환

■ 불 자료형 자동 변환

- [예제 2-19] 불 자료형 자동 변환
 - ! 연산자를 두 번 사용해 Boolean() 함수를 사용하는 것과 같은 효과

코드 2-28 불 자료형 자동 변환

notnot.js

```
// 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);
```

실행 결과

```
false  
false  
false  
false  
false
```


11. 일치 연산자

일치 연산자

자료형까지 검사

표 2-18 일치 연산자

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

[예제 2-20] 비교 연산자와 일치 연산자의 차이

코드 2-29 비교 연산자와 일치 연산자의 차이

equality.js

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

실행 결과

```
52 == "52": true  
52 === "52": false  
  
0 == "": true  
0 === "": false
```

12. 상수

■ 상수

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

12. 상수

■ [예제 2-21] 상수와 오류 (1)

코드 2-30

상수와 오류 (1)

constant1.js

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

실행 결과

C:\example\constant1.js:3

constant = "";

^

TypeError: Assignment to constant variable.

이후 아래 부분은 생략하겠습니다.

```
at Object.<anonymous> (C:\example\constant1.js:3:10)  
[90m   at Module._compile (internal/modules/cjs/loader.js:1085:14)[39m  
[90m   at Object.Module._extensions..js (internal/modules/cjs/loader.js:1114:10)[39m  
[90m   at Module.load (internal/modules/cjs/loader.js:950:32)[39m  
[90m   at Function.Module._load (internal/modules/cjs/loader.js:790:14)[39m  
[90m   at Function.executeUserEntryPoint [as runMain] (internal/modules/run_main.js:  
76:12)[39m  
[90m   at internal/main/run_main_module.js:17:47[39m
```

12. 상수

■ [예제 2-22] 상수와 오류 (2)

코드 2-31 상수와 오류 (2)

```
const a = "처음 선언할 때 값을 할당해야 합니다";  
const b;
```

실행 결과

C:\example\arrayLength.js:2

```
const b;
```

^

SyntaxError: Missing initializer in const declaration

