웹프로그래밍1

11주차 – 3차시

컴퓨터공학과

JavaScript 자바스크립트 마스터북

기초부터 Ajax/JQuery까지

야마다 요시히로 지음 / 정인식 옮김

Jpub 제이펍

변수에 대한 고찰

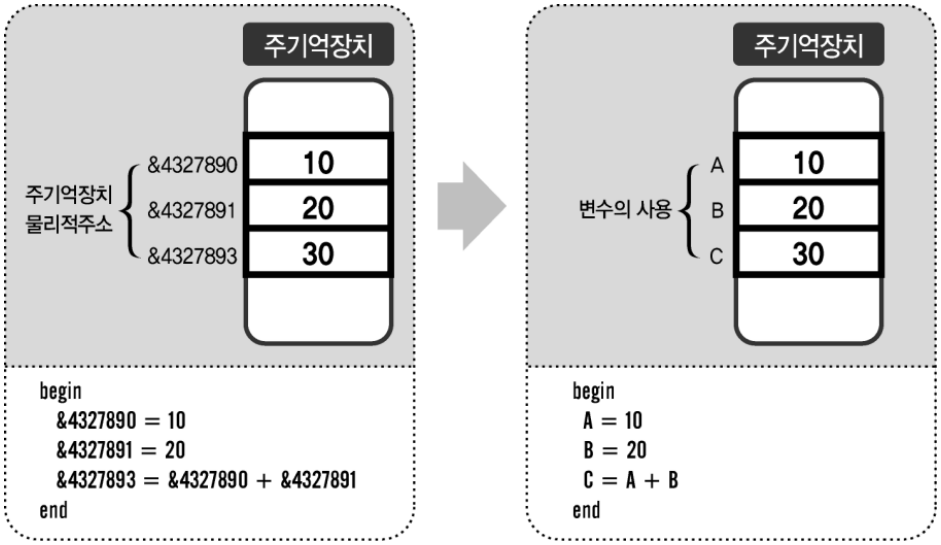
- 변수의 생성
  - $a = 10; // lvalue = rvalue
  - $b = “schoool”;
  - $c = 3481;

Variable Name	Memory address (Left Side Value)	Store Value (Right Side Value)
a	2FC7A8	10
b	2FC7AD	school
c	2FC7C0	3481

참조 = reference = address = lvalue

변수에 대한 고찰

- 프로그램에서 다양한 종류의 데이터를 표현하기 위해 사용
- 변수(Variable)
  - 프로그램에서 필요로 되는 데이터를 저장하기 위한 기억장소에주어진이름
  - 변수를 사용하지 않고, 기억장치의 물리적 주소를 사용하여 프로그램을 작성한다는 것은 불가능한 일이다
  - **주기억장치의 물리적 주소를 사용하는 프로그램과 변수를 사용하는 프로그램**



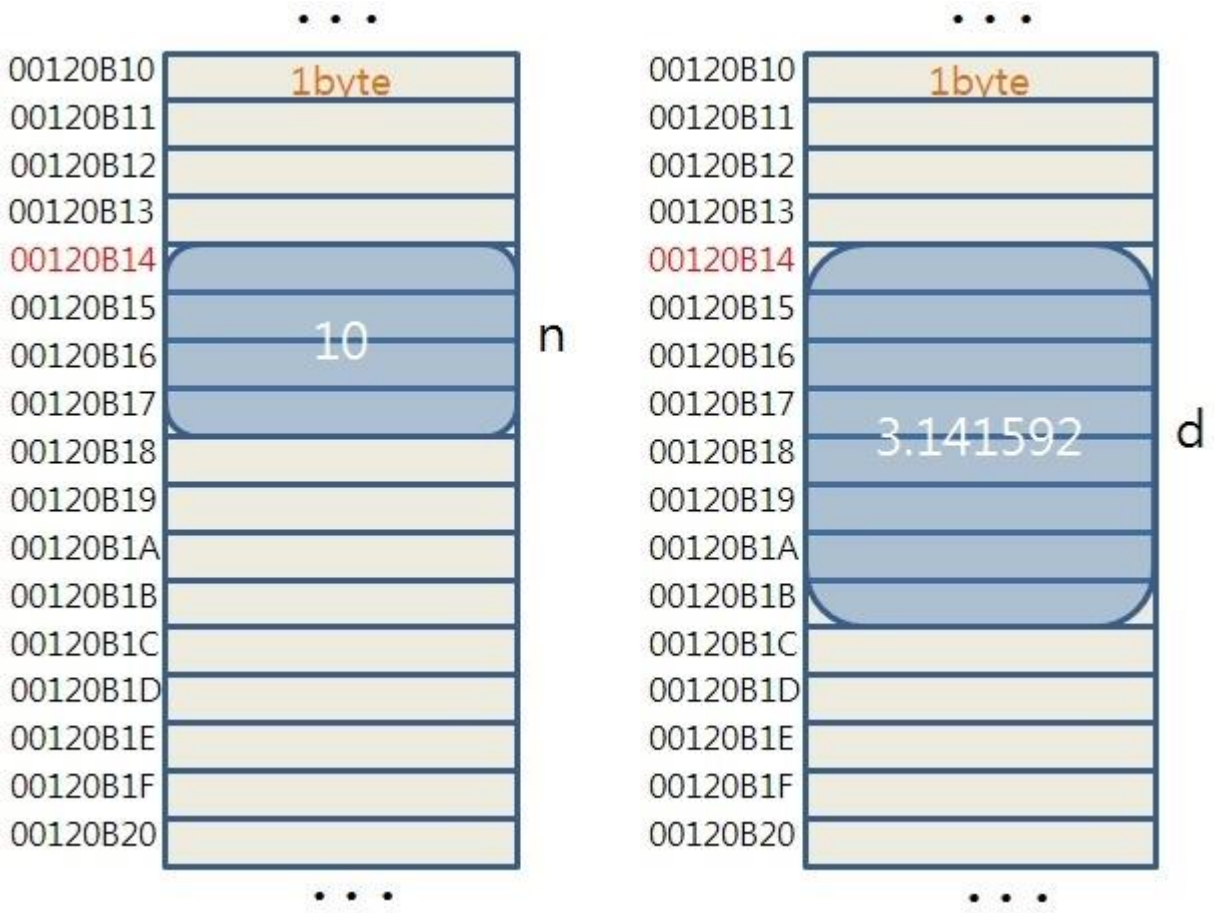
데이터타입 : 기본형/참조형

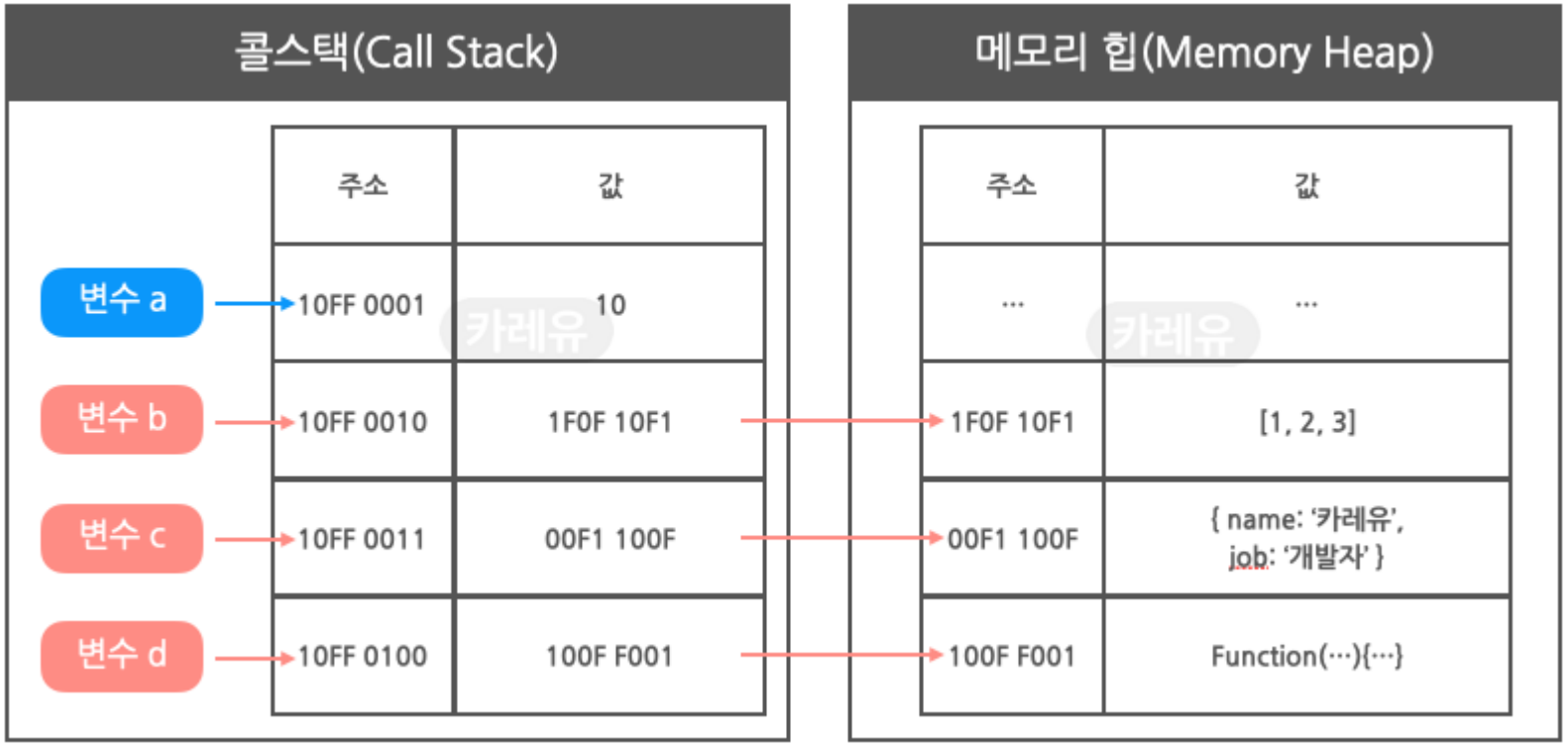
분류	데이터형	개요
기본형	수치형(number)	$\pm 4.94065645841246544 \times 100^{-324} \sim \pm 1.79769313486231570 \times 10^{308}$
	문자열형(string)	작은따옴표/큰따옴표로 감싸인 0개 이상의 문자 집합
	논리형(boolean)	true(참)/false(거짓)
	특수형(null/undefined)	값이 미정의된 것을 나타냄

분류	데이터형	개요
참조형	배열(array)	데이터의 집합(각 요소에는 인덱스 번호로 접근 가능)
	객체(object)	데이터의 집합(각 요소에는 이름으로 접근 가능)
	함수(function)	일련의 처리(절차)의 집합(제4장을 참조)

- Strictly Data Type : Java, C#
  - 숫자변수에는 숫자만, 문자변수에는 문자만 저장
  - 변수와 데이터형이 항상 한 쌍
- Loosely Data Type
  - 문자변수에 나중에 숫자 저장 가능

메모리 주소

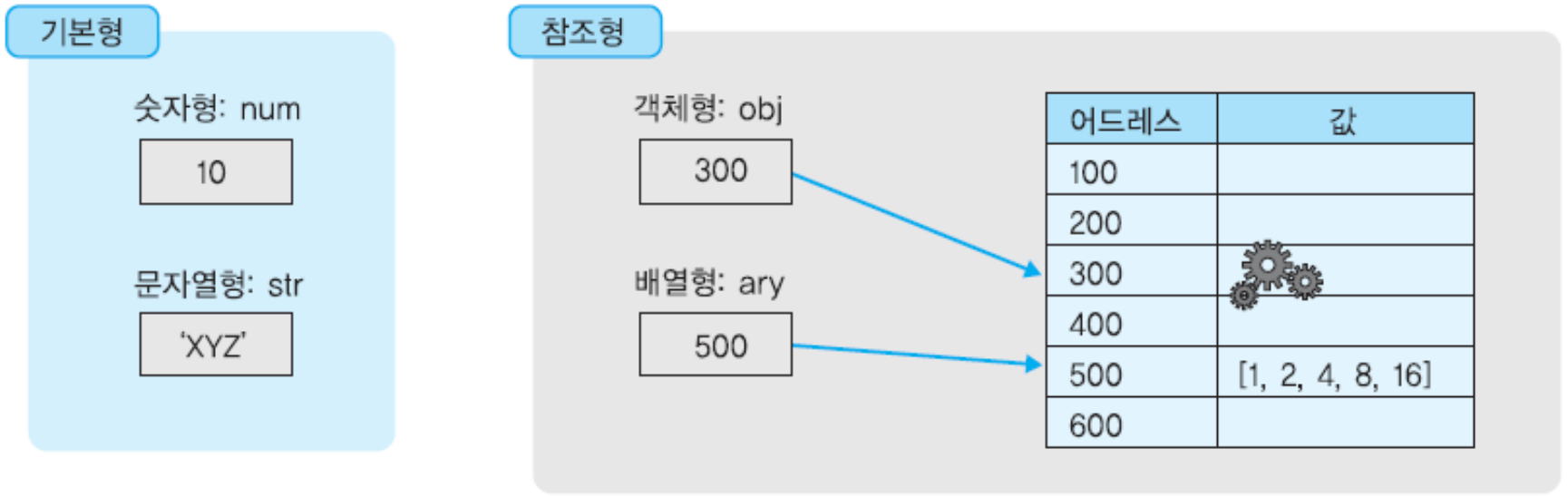

00120B10
00120B11
00120B12
00120B13
00120B14
00120B15
00120B16
00120B17
00120B18
00120B19
00120B1A
00120B1B
00120B1C
00120B1D
00120B1E
00120B1F
00120B20
1byte
10
n
3.141592
d

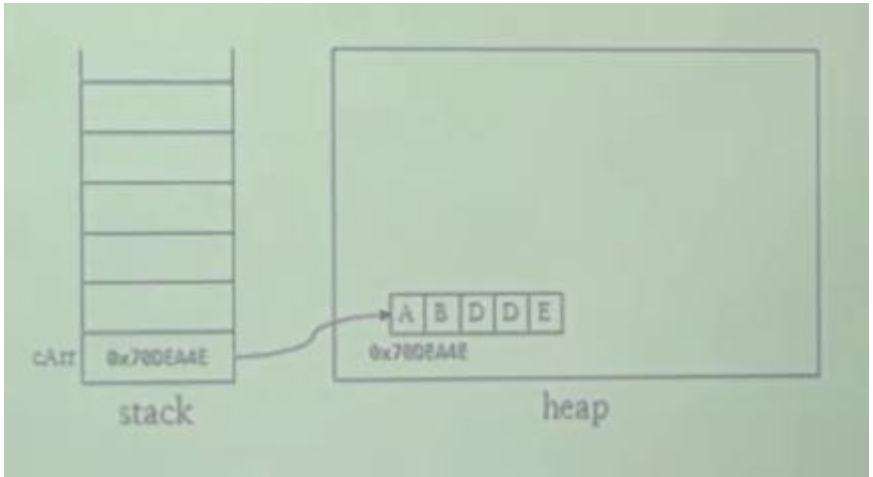

콜스택(Call Stack)
주소
값
변수 a
10FF 0001
10
변수 b
10FF 0010
1F0F 10F1
변수 c
10FF 0011
00F1 100F
변수 d
10FF 0100
100F F001
메모리 힙(Memory Heap)
주소
값
…
…
1F0F 10F1
[1, 2, 3]
00F1 100F
{ name: '카레유',
job: '개발자' }
100F F001
Function(…){…}

자바스크립트 코드

let a = 10;

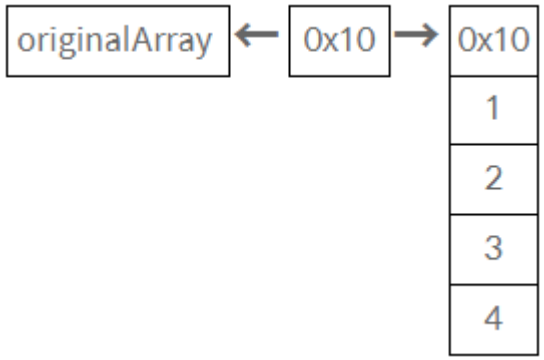
- 기본형과 참조형 : 값을 변수에 대입하는 방법에 차이
  - 기본형 변수 : 값 그 자체가 직접 보관
  - 참조형 변수 : 그 참조값(값을 실제로 보관하고 있는 메모리의 어드레스)을 보관
  - **기본형과 참조형에 따라 데이터의 취급방식이 틀리다.**



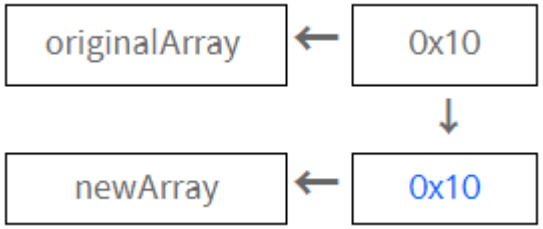


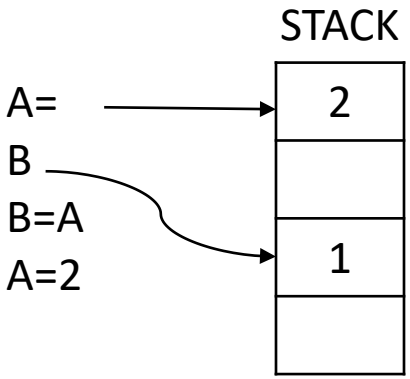
대입연산자

originalArray = [1,2,3,4]

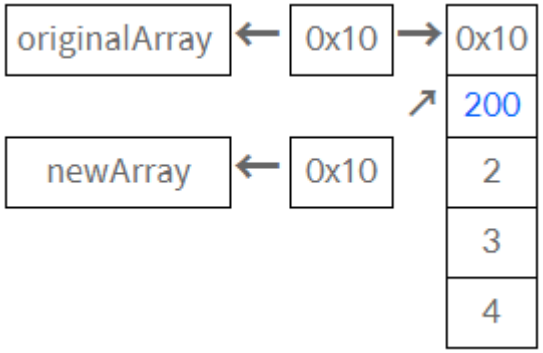


newArray = originalArray





newArray[0] = 200



대입연산자

- 기본형과 참조형에 따른 대입의 차이 -「=」연산자

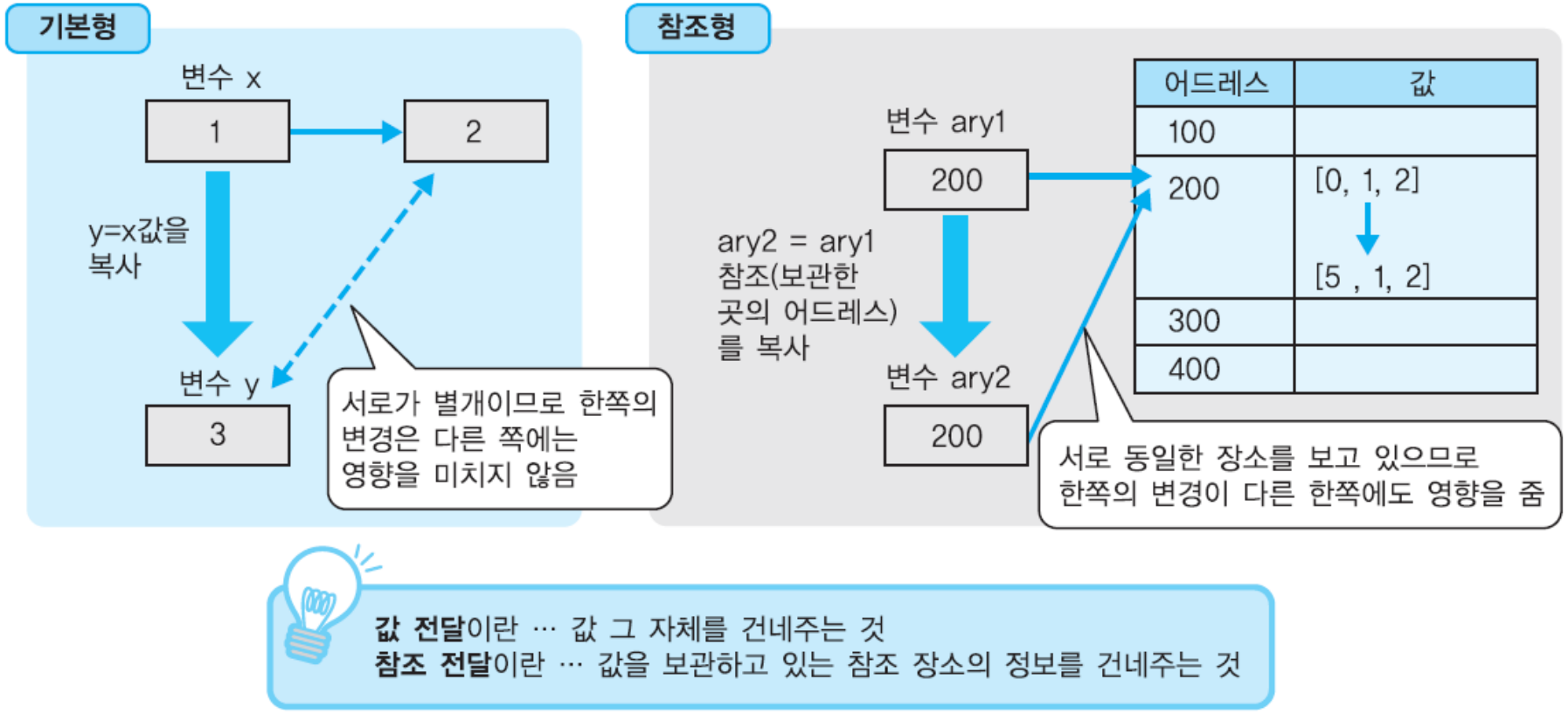
```
var x = 1;
var y = x;
x = 2;
document.writeln(y); // ❶ 1
var ary1 = [0, 1, 2]; // 배열 리터럴을 선언
var ary2 = ary1;
ary1[0] = 5;
document.writeln(ary2); // ❷ 5,1,2
```

- 참조 전달 : call-by-address(reference)
  - 변수 ary1에 보관되어 있는 어드레스를 변수 ary2에 보관하고 있는 것에 불과
- 값 전달 : call-by-value

대입연산자

```
var x = 1;
var y = x;
x = 2;
document.writeln(y); // ❶ 1
var ary1 = [0, 1, 2]; // 배열 리터럴을 선언
var ary2 = ary1;
ary1[0] = 5;
document.writeln(ary2); // ❷ 5,1,2
```

Stack memory
Heap memory



Note

변수의 정체

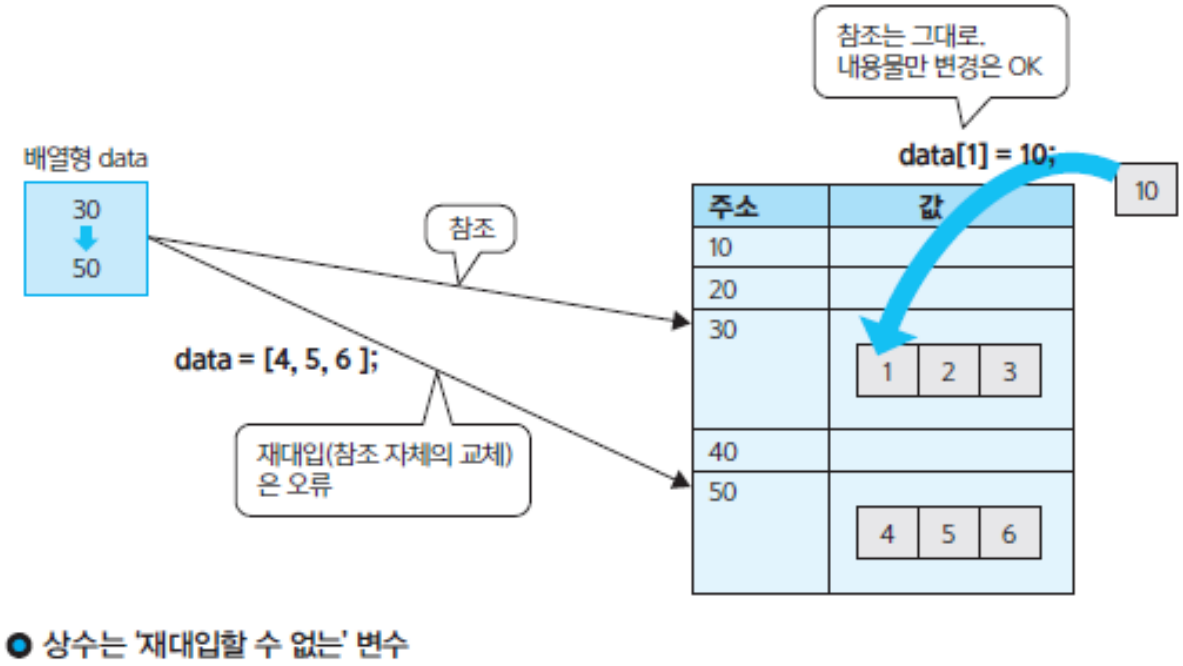
값을 보관하는 것은, 정확하게는 컴퓨터 상에 준비된 메모리의 역할이다. 메모리에는 각각의 장소를 나타내는 번호(어드레스)가 부여되어 있다. 그러나 스크립트에 의미가 없는 번호를 기술하는 것은 보기에도 이해하기 어렵고 타이핑 미스의 원인이 되기도 한다. 그래서 어드레스 대신에 인간이 이해하기 쉽도록 이름을 부여한 것 – 그것이 변수의 정체다.

변수란 「값의 보관처(어드레스)에 대해 부여된 명찰」이라고도 말할 수 있다.

상수 → 재대입할 수 없다?

```
const TAX = 1.08;
TAX = 1.1; //error
```

```
const data = [1, 2, 3];
data = [4, 5, 6];
data[1] = 10;
```



● 상수는 '재대입할 수 없는' 변수

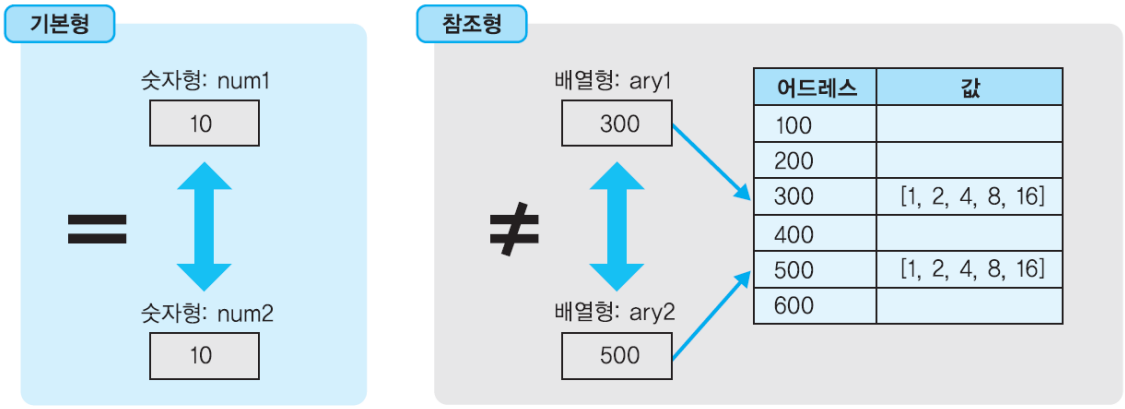
비교 연산자

- 등가 연산자(==)
  - 배열이나 객체 등의 이른바 참조형인 경우에는 주의

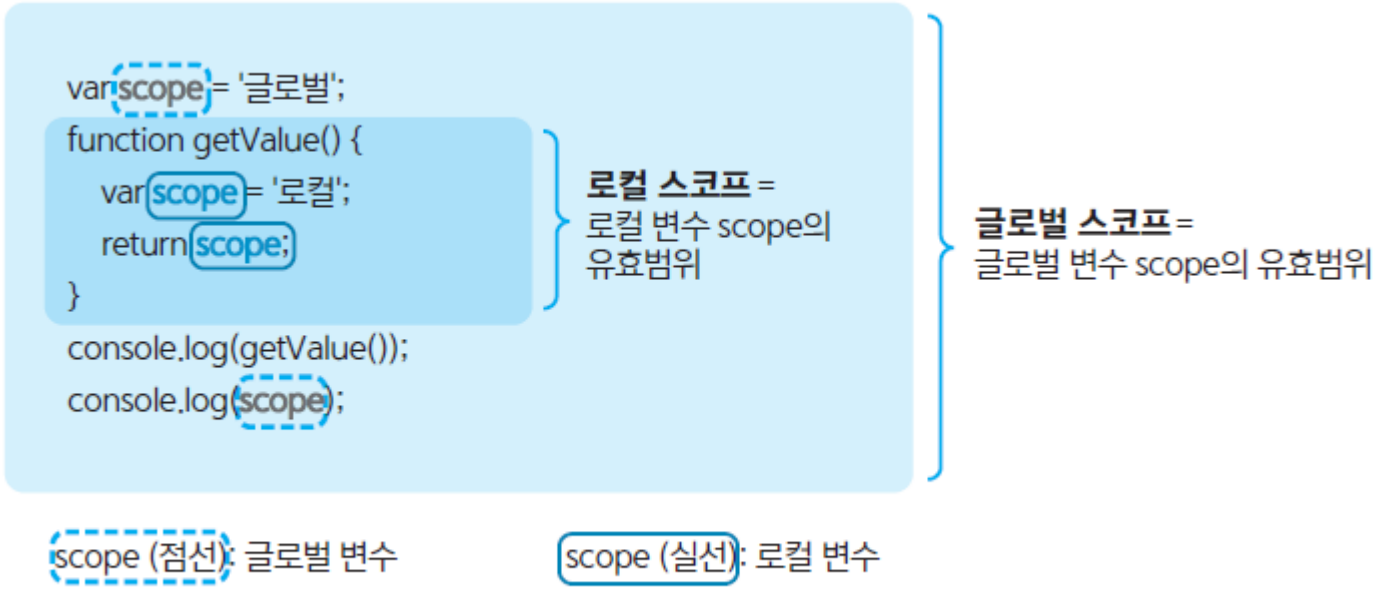
```
var ary1 = ['JavaScript', 'Ajax', 'ASP.NET'];
var ary2 = ['JavaScript', 'Ajax', 'ASP.NET'];
document.writeln(ary1 == ary2); // false
```

equals.html

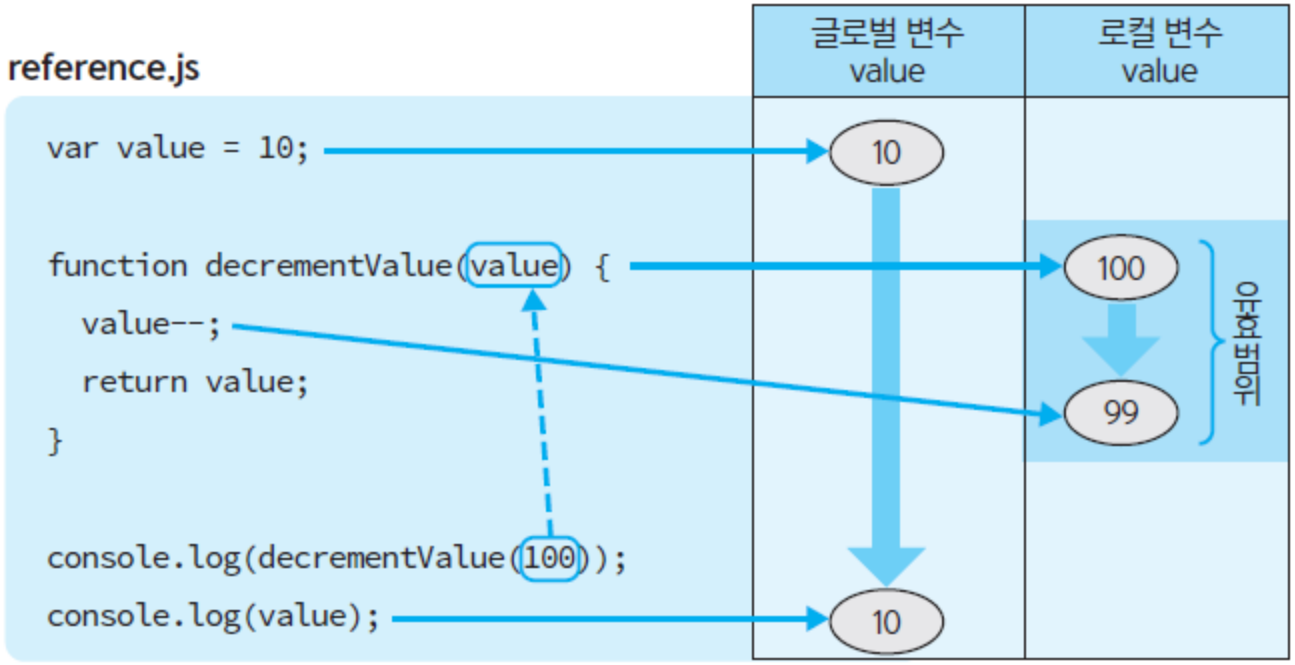
  - 기본형은 변수에 값을 직접 보관한 반면,
  - 참조형에서는 그 참조값(메모리 상의 어드레스)이 보관
    - 메모리 상의 어드레스가 동일한 경우에만 true가 반환
    - 동일한 내용을 포함하고 있는 객체라고 해도 그것이 다른 객체(다른 어드레스로 등록된 것)라면, 등가 연산자는 false를 반환



기본형에서는 … 값 그 자체를 비교하므로 보이는 것과 비교 결과가 일치
참조형에서는 … 값을 보관하고 있는 참조 장소를 비교하므로 보이는 것과 비교 결과가 서로 일치하지 않는 경우가 있다



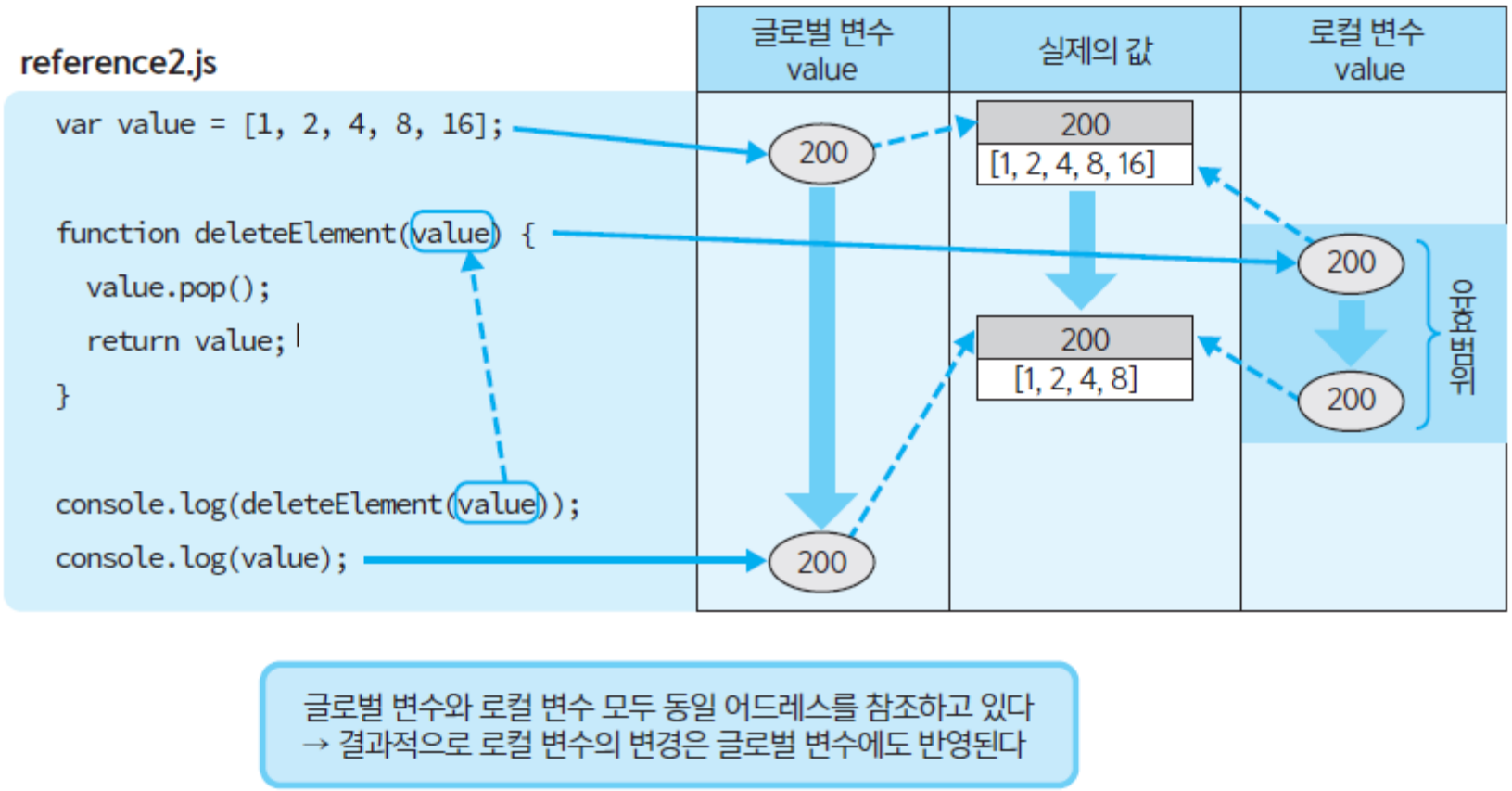
● 글로벌 스코프와 로컬 스코프



글로벌 변수 value와 로컬 변수 value는 **별개**의 것이므로
로컬 변수에서의 변경이 글로벌 변수에 영향을 주지 않는다.

● 글로벌 변수와 로컬 변수(기본형)

Side effect



글로벌 변수와 로컬 변수 모두 동일 어드레스를 참조하고 있다
→ 결과적으로 로컬 변수의 변경은 글로벌 변수에도 반영된다

● 글로벌 변수와 로컬 변수(참조형)