

\* static type binding & dynamic type binding

Java :

int a ;  
↓  
고정 (static)

a = 100 ; //ok

~~a = true ;~~

"static type binding"  
↓  
변수 선언시에 결정된  
↓  
변경 불가 !

JavaScript :

var a ;  
↑  
undefined

String a = "hello" ; //ok  
↓  
동적 (dynamic)

boolean a = true ; //ok  
↓  
동적

"dynamic type binding"  
↓  
같은 변수에 서로 다른  
변수의 타입이  
가정된다  
↓  
변수의 타입은  
값에 따라  
가변적이다

\* 파라미터 (parameter) 와 아규먼트 (argument)

function plus(a, b) {  
 return a+b;  
}

아규먼트를 받은 변수  $\Rightarrow$  파라미터

함수에서는 꼭 써야 한다.

var result = plus(100, 200);

호출할 때  
넘겨주는 값  $\Rightarrow$  아규먼트



\* 함수 이름과 함수 객체

function plus(a, b) { ... }

함수 객체의  
주소를  
가진  
레퍼런스가.

plus  
200

함수에 대한 정보와 코드를 담은  
객체 생성

함수 객체

200  
파라미터: a, b  
코드: { ... }

var ok;

ok = plus;

~~200~~  
↓  
400

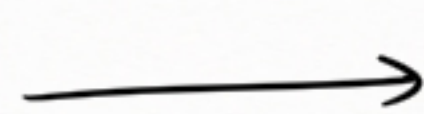
var haha = ok;

200

400  
"hello"  
가치

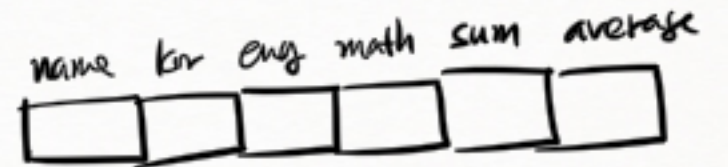
\* 상제/로 기본 객체 생성 vs 프로그래밍 기본 객체 생성  
원형  
(기본형)

Java :  
 class Score {  
   String name;  
   int kor;  
   int eng;  
   int math;  
   int sum;  
   float average;  
 }



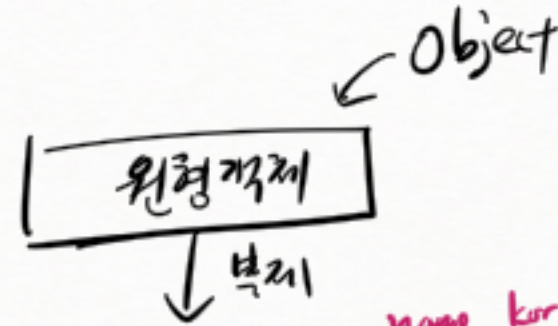
new Score()

상제/로에 따라  
 ↓  
 객체 생성



JavaScript:

new Score()



원형 객체  
(기본형) +



## \* 외부 스크립트 요청

