function foo(){}

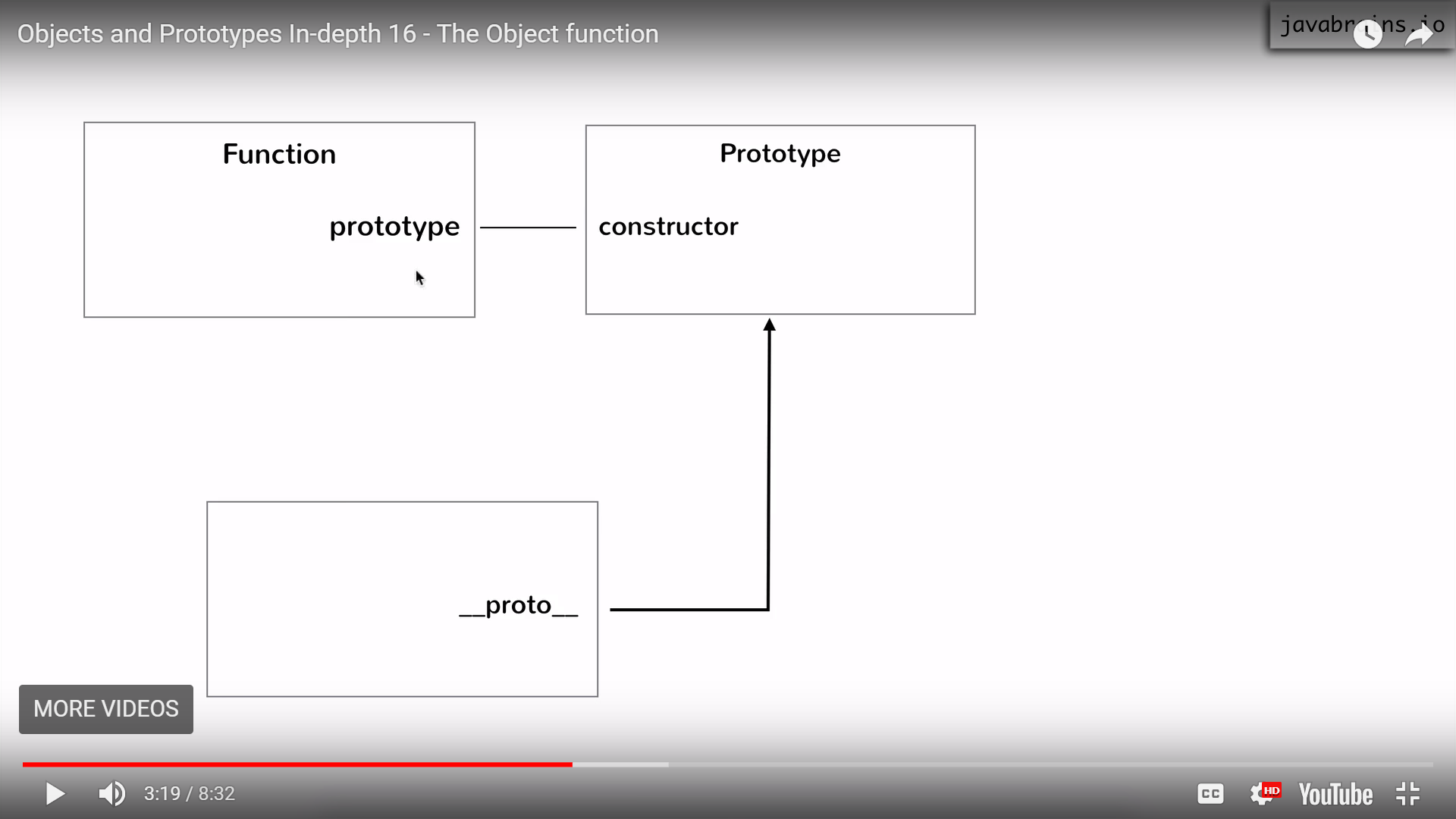
foo.prototype => PROTOTYPE // points to prototype

var a = new foo();

a.\_\_proto\_\_ => PROTOTYPE // Also points to prototype

foo.prototype.constructor => function foo() // constructor property on prototype object points to function of which its the prototype is

refer 3-Closure.js



Object created by using new Function

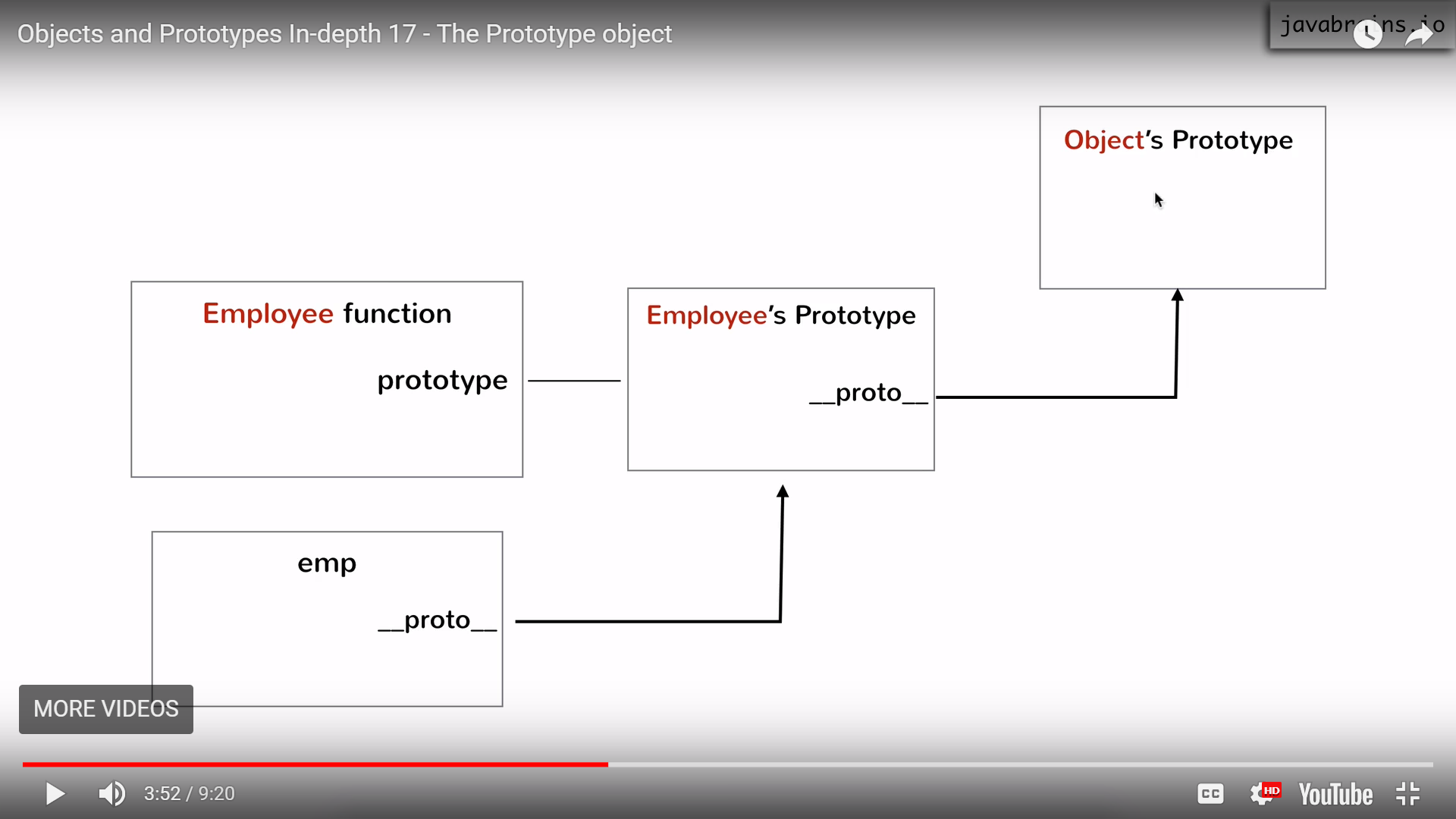
Function Employee(){}

Var emp = new Employee();

so emp.\_\_proto\_\_ === Employee.prototype(parent for all Employee’s objects)

and emp.\_\_proto\_\_.\_\_proto\_\_ === Object.prototype(GrandParent for all object’s created using new(constructor mode))

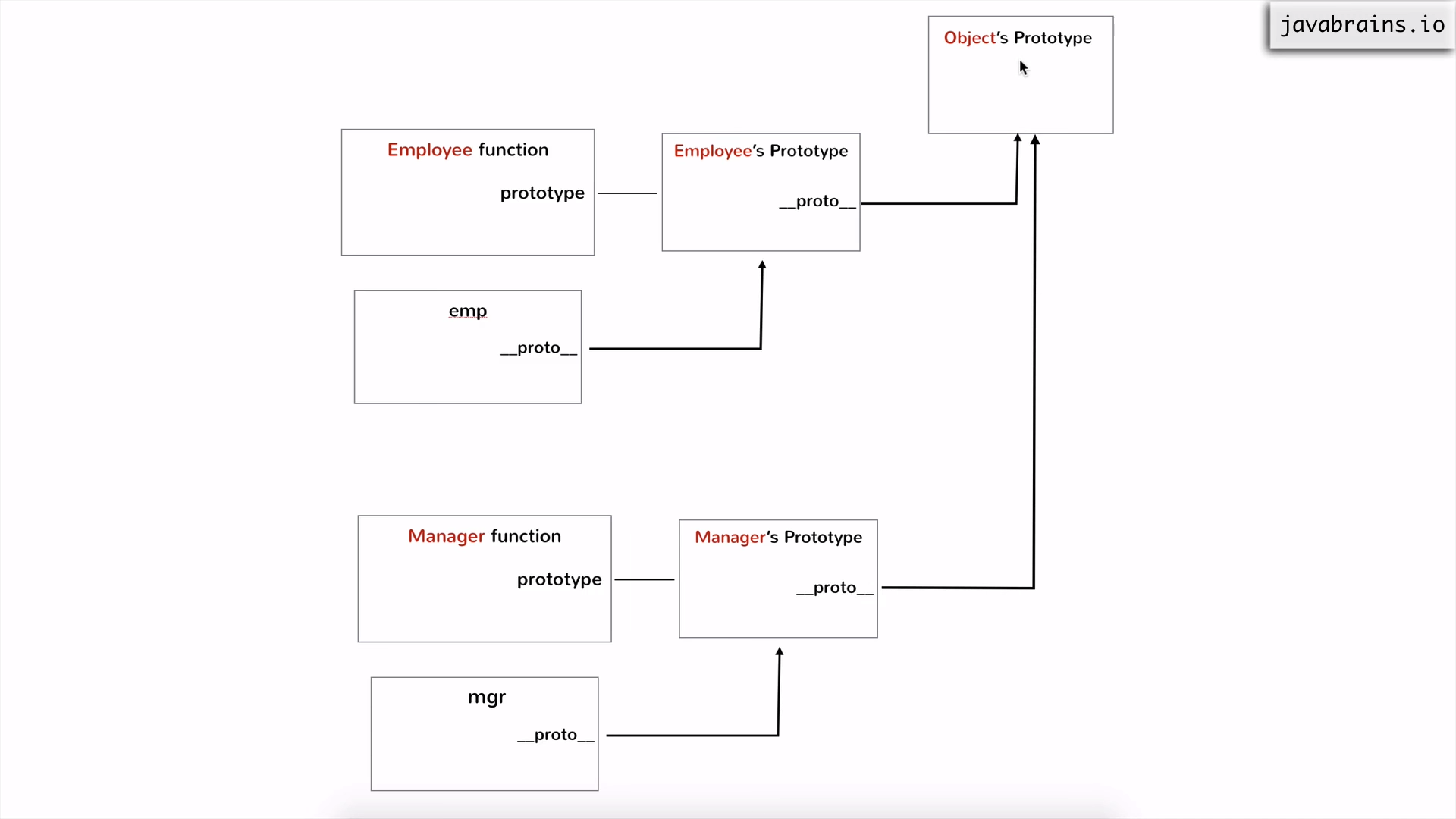
Object.prototype.\_\_proto\_\_ === null (JS puts it by default)



==================================================

Employee and manager Hierarchy

Normal case



Inheritance can be created by changing Manager’s Prototype to point to Employee’s Prototype

So Manager can have access to all property/function of Employee(Prototype) but employee can’t access Manager’s

