* ways to construct Objects

function ninja (name, rank)= {

var obj = {};

obj.name = name;

obj.rank = rank;

return obj;

}

var Jack = ninja(‘Jack’, ‘master’);

function ninja(name, rank) = {

var obj = {

name : name,

rank : rank,

}

return obj;

}

* Private variable and global ones

function ninja (name, rank, clan)= {

var obj = {};

obj.name = name;

obj.rank = rank;

return obj;

var clan = clan

* + using the var keyword when you’re creating an object like this makes it a private variable, anything outise doesn’t have the scope to change this.

obj.clan\_show = function() {return clan}

* + since clan is a private variable, we can’t change it, but we can access it using a method like clan\_show.

}

* Using this and new to create a constructor object

function NewNinja(name, rank){

this.name = name;

this.rank = rank;

}

var jack = new NewNinja(‘Jack’, ‘master’)

* + using this inside the constructor references the constructor itself. Also, since NewNinja is capitalized, javascript automatically recognizes it as a constructor function. Lastly, using new when you’re declaring a new ninja, automatically turns NewNinja into an object to be plugged into the variable jack.
* using prototype

you can use prototype outside of the initial writing of the constructor function to make it faster and more efficient, and every instance of the function will have access to it.

function NewNinja(name, rank){

this.name = name;

this.rank = rank;

}

NewNinja.prototype.sayHi = function(){console.log(‘Hi’)}

* using prototype using just like declaring this method inside the the function NewNinja, now every instance of it, such as the one below, can use it. Also, everytime we create a new instance, the computer doesn’t have to read through this method, since it isn’t written in that block of code. If we had 100 methods written, it would make a difference in speed/computing.

var jack = new NewNinja(‘Jack’, ‘master’)