JavaScript: function returning an object

Asked 6 years, 10 months ago Active 13 days ago Viewed 215k times

64

31

I'm taking some JavaScript/jQuery lessons at codecademy.com. Normally the lessons provide answers or hints, but for this one it doesn't give any help and I'm a little confused by the instructions.

It says to make the function makeGamePlayer return an object with three keys.

```
//First, the object creator
function makeGamePlayer(name,totalScore,gamesPlayed) {
    //should return an object with three keys:
    // name
    // totalScore
    // gamesPlayed
}
```

I'm not sure if i should be doing this

```
//First, the object creator
function makeGamePlayer(name,totalScore,gamesPlayed) {
    //should return an object with three keys:
    // name
    // totalScore
    // gamesPlayed

         this.name =  name;
         this.totalScore = totalScore;
         this.gamesPlayed = gamesPlayed;
}
```

or something like this

```
 //First, the object creator
    function makeGamePlayer(name,totalScore,gamesPlayed) {
        //should return an object with three keys:
        // name
        // totalScore
        // gamesPlayed
```

```
        var obj = {
            this.name =  name;
            this.totalScore = totalScore;
            this.gamesPlayed = gamesPlayed;
         }
    }
```

I have to be able to modify the properties of the object after its created.

javascript

edited Sep 16 '12 at 2:39

David Robinson
62.2k 11 127 161

asked Sep 4 '12 at 22:30

BrainLikeADullPencil
4,785 18 61 116

6 Answers

110

In JavaScript, most functions are both callable and instantiable: they have both a [[Call]] and [[Construct]] internal methods.

As callable objects, you can use parentheses to call them, optionally passing some arguments. As a result of the call, the function can return a value.

```
var player = makeGamePlayer("John Smith", 15, 3);
```

The code above calls function `makeGamePlayer` and stores the returned value in the variable `player`. In this case, you may want to define the function like this:

```
function makeGamePlayer(name, totalScore, gamesPlayed) {
  // Define desired object
  var obj = {
    name:  name,
    totalScore: totalScore,
    gamesPlayed: gamesPlayed
  };
  // Return it
  return obj;
}
```

Additionally, when you call a function you are also passing an additional argument under the hood, which determines the value of `this` inside the function. In the case above, since `makeGamePlayer` is not called as a method, the `this` value will be the global object in sloppy mode, or undefined in strict mode.

As constructors, you can use the `new` operator to instantiate them. This operator uses the [[Construct]] internal method (only available in constructors), which does something like this:

1. Creates a new object which inherits from the `.prototype` of the constructor
2. Calls the constructor passing this object as the `this` value
3. It returns the value returned by the constructor if it's an object, or the object created at step 1 otherwise.

```
var player = new GamePlayer("John Smith", 15, 3);
```

The code above creates an instance of `GamePlayer` and stores the returned value in the variable `player`. In this case, you may want to define the function like this:

```
function GamePlayer(name,totalScore,gamesPlayed) {
  // `this` is the instance which is currently being created
  this.name =  name;
  this.totalScore = totalScore;
  this.gamesPlayed = gamesPlayed;
  // No need to return, but you can use `return this;` if you want
}
```

By convention, constructor names begin with an uppercase letter.

The advantage of using constructors is that the instances inherit from `GamePlayer.prototype`. Then, you can define properties there and make them available in all instances

edited Apr 25 '16 at 23:00

answered Sep 4 '12 at 22:34

Oriol
170k 38 291 384

3 @OP also note that when you are going to invoke it with the `new` keyword I would suggest starting the name with a capital: `MakeGamePlayer`. – PeeHaa Sep 4 '12 at 22:36

2 @Oriol–check your object literal, it has syntax errors. – RobG Sep 4 '12 at 22:38

3 @PeeHaa Good advice, also the more typical naming convention when using the constructor would be `new GamePlayer()`. – Matt Zeunert Sep 4 '12 at 22:39

@RobG Thanks, that's what happens when I copy-paste code without looking at it deeply. – Oriol Sep 4 '12 at 22:41

37

You can simply do it like this with an object literal:

```
function makeGamePlayer(name,totalScore,gamesPlayed) {
    return {
        name: name,
        totalscore: totalScore,
        gamesPlayed: gamesPlayed
    };
}
```

answered Sep 4 '12 at 22:33

PeeHaa

51.4k 43 172 247

1 hello, how can you access the return properties , When I use makeGamePlayer.name it doesnt work – Cameron A Dec 4 '18 at 19:30

5

Both styles, with a touch of tweaking, would work.

The first method uses a Javascript Constructor, which like most things has pros and cons.

```
 // By convention, constructors start with an upper case letter
function MakePerson(name,age) {
  // The magic variable 'this' is set by the Javascript engine and points to a newly
created object that is ours.
  this.name = name;
  this.age = age;
  this.occupation = "Hobo";
}
var jeremy = new MakePerson("Jeremy", 800);
```

On the other hand, your other method is called the 'Revealing Closure Pattern' if I recall correctly.

```
function makePerson(name2, age2) {
  var name = name2;
```

```
  var age = age2;

  return {
    name: name,
    age: age
  };
}
```

edited Sep 4 '12 at 22:44

answered Sep 4 '12 at 22:36

Jeremy J Starcher
19.4k 5 43 64

Is called "revealing module pattern" But it is usually wrapped into private closure (function(){return{}})() – Felipe Quirós Jul 16 at 21:09

2

I would take those directions to mean:

```
function makeGamePlayer(name,totalScore,gamesPlayed) {
      //should return an object with three keys:
      // name
      // totalScore
      // gamesPlayed

       var obj = {  //note you don't use = in an object definition
           "name": name,
           "totalScore": totalScore,
           "gamesPlayed": gamesPlayed
        }
       return obj;
  }
```

edited Jun 29 '16 at 16:14

answered Sep 4 '12 at 22:34

scrappedcola
9,555 1 22 40

1 Why do you have semicolons inside of object? – Alex G Jun 27 '16 at 2:04

@AlexG nice catch, can't believe no one else did in the 4 years since I first posted this answer. It was undoubtedly a bad cut-n-paste job of the OP's original object that I and it looks like a few others made. – scrappedcola Jun 29 '16 at 16:15

The latest way to do this with ES2016 JavaScript

2

```
let makeGamePlayer = (name, totalScore, gamesPlayed) => ({
    name,
    totalScore,
    gamesPlayed
})
```

answered Dec 10 '18 at 9:13

Rob

3,384 4 22 34

0

```
const upadteAgeAndCount = async (id,age) => {
    const user = await User.findByIdAndUpdate(id,{age},{new:true})
    const count = await User.countDocuments({age})
    return ({user,count})
}
```

answered Jul 15 at 15:31

vithu shaji

3 2