Pass a JavaScript function as parameter

Asked 6 years, 11 months ago Active 1 year, 7 months ago Viewed 583k times

608

160

How do I pass a function as a parameter without the function executing in the "parent" function or using `eval()` ? (Since I've read that it's insecure.)

I have this:

```
addContact(entityId, refreshContactList());
```

It works, but the problem is that `refreshContactList` fires when the function is called, rather than when it's used in the function.

I could get around it using `eval()` , but it's not the best practice, according to what I've read. How can I pass a function as a parameter in JavaScript?

javascript function parameters

edited Jun 1 '15 at 23:38
Nic Hartley
4,783 7 39 58

asked Nov 8 '12 at 9:34
imperium2335
8,513 32 94 162

@Ilyaskarim your link does no longer work – smartmeta Feb 17 at 13:07

@smartmeta the site is closed. – Ilyas karim Feb 17 at 15:04

13 Answers

You just need to remove the parenthesis:

This then passes the function without executing it first.

Here is an example:

```
function addContact(id, refreshCallback) {
    refreshCallback();
    // You can also pass arguments if you need to
    // refreshCallback(id);
}

function refreshContactList() {
    alert('Hello World');
}

addContact(1, refreshContactList);
```

edited Nov 8 '12 at 9:43

answered Nov 8 '12 at 9:34
Fenton
169k 48 304 332

6 @stevefenton, consider updating your answer with h2ooooooo comment, it'd be very useful. – Morgan Wilde Nov 8 '12 at 9:37

4 Based on the question, the callback doesn't accept parameters, which is why I have left them out of the example. I'll add a comment about it. – Fenton Nov 8 '12 at 9:42

4 @Veverke It would look like this... `addContact(1, function(id) { console.log(id); });` – Fenton Nov 24 '14 at 10:46

4 @Steve Fenton: after reading your reply I asked myself why did I ask... :-) – Veverke Nov 25 '14 at 15:52

1 The class syntax in ECMAScript wouldn't have an `=` sign... `class myFuncs {` rather than `class myFuncs = {` . You'd also need to be running in an environment that supported the class syntax (not all browsers support it yet). If you are still struggling, it might be better suited to a whole new question as your problem isn't about passing functions - it is general ES syntax. – Fenton Jun 21 '16 at 21:08

If you want to pass a function, just reference it by name without the parentheses:

290

```
function foo(x) {
    func("Hello World!");
}

//alerts "Hello World!"
bar(foo);
```

But sometimes you might want to pass a function *with arguments included*, but not have it called until the callback is invoked. To do this, when calling it, just wrap it in an anonymous function, like this:

```
function foo(x) {
   alert(x);
}
function bar(func) {
   func();
}

//alerts "Hello World!" (from within bar AFTER being passed)
bar(function(){ foo("Hello World!") });
```

If you prefer, you could also use the apply function and have a third parameter that is an array of the arguments, like such:

```
function eat(food1, food2)
{
    alert("I like to eat " + food1 + " and " + food2 );
}
function myFunc(callback, args)
{
    //do stuff
    //...
    //execute callback when finished
    callback.apply(this, args);
}

//alerts "I like to eat pickles and peanut butter"
myFunc(eat, ["pickles", "peanut butter"]);
```

edited Nov 7 '14 at 1:23

answered Jun 6 '14 at 23:11

dallin
4,737 1 22 33

that makes JavaScript so powerful and so great to code in. – TheHansinator Dec 9 '15 at 22:16

@Compynerd255 I agree, this and the ability to quickly create object literals are my two favorite aspects of Javascript. I always miss object literals in languages that don't have them. – dallin Dec 10 '15 at 22:34

3 I am so thankful to Javascript for providing this feature and to you @dallin to letting me know that it exists. – Dipendu Paul Apr 7 '16 at 11:28

5 Definitely this must be the accepted answer. – andreszs Apr 10 '17 at 14:58

51

Example 1:

```
funct("z", function (x) { return x; });

function funct(a, foo){
    foo(a) // this will return a
}
```

Example 2:

```
function foodemo(value){
    return 'hello '+value;
}

function funct(a, foo){
    alert(foo(a));
}

//call funct
funct('world!',foodemo); //=> 'hello world!'
```

look at this

edited May 23 '17 at 12:02
Community ♦
1 1

answered Nov 8 '12 at 9:38
Gadde
1,307 13 34

34

```
function ToBeCalled(){
  alert("I was called");
}

function iNeedParameter( paramFunc) {
   //it is a good idea to check if the parameter is actually not null
   //and that it is a function
   if (paramFunc && (typeof paramFunc == "function")) {
      paramFunc();
   }
}

//this calls iNeedParameter and sends the other function to it
iNeedParameter(ToBeCalled);
```

The idea behind this is that a function is quite similar to a variable. Instead of writing

```
function ToBeCalled() { /* something */ }
```

you might as well write

```
var ToBeCalledVariable = function () { /* something */ }
```

There are minor differences between the two, but anyway - both of them are valid ways to define a function. Now, if you define a function and explicitly assign it to a variable, it seems quite logical, that you can pass it as parameter to another function, and you don't need brackets:

```
anotherFunction(ToBeCalledVariable);
```

edited Sep 18 '13 at 23:44
Colonel Panic
1,476 2 17 31

answered Nov 8 '12 at 9:37
naivists
27.5k 5 48 79

2 Just `typeof paramFunc == "function"` is enough, cause if it isn't callable, then you can ignore it. – Jimmy Knoot Mar 10 '15 at 14:26

15

In addition to Steve Fenton's answer, you can also pass functions directly.

```
function addContact(entity, refreshFn) {
    refreshFn();
}

function callAddContact() {
    addContact("entity", function() { DoThis(); });
}
```

answered Nov 8 '12 at 9:39

series0ne
18.2k 28 113 207

I chopped all my hair off with that issue. I couldn't make the examples above working, so I ended like :

7

```
function foo(blabla){
    var func = new Function(blabla);
    func();
}
// to call it, I just pass the js function I wanted as a string in the new one...
foo("alert('test')");
```

And that's working like a charm ... for what I needed at least. Hope it might help some.

answered Apr 15 '16 at 9:31

Fenix Aoras
136 2 7

5 Upvote for the starting sentence xD – GedankenNebel Jul 31 '17 at 10:44

I suggest to put the parameters in an array, and then split them up using the `.apply()` function. So now we can easily pass a function with lots of parameters and execute it in a simple way.

```
}

function refreshContactList(int, int, string) {
    alert(int + int);
    console.log(string);
}

addContact([1,2,"str"], refreshContactList); //parameters should be putted in an array
```

answered Sep 26 '15 at 17:44

Naramsim
3,436 4 21 35

5

You can also use `eval()` to do the same thing.

```
//A function to call
function needToBeCalled(p1, p2)
{
    alert(p1+"="+p2);
}

//A function where needToBeCalled passed as an argument with necessary params
//Here params is comma separated string
function callAnotherFunction(aFunction, params)
{
    eval(aFunction + "("+params+")");
}

//A function Call
callAnotherFunction("needToBeCalled", "10,20");
```

That's it. I was also looking for this solution and tried solutions provided in other answers but finally got it work from above example.

answered Jan 23 '14 at 11:29

NullPointer
2,016 3 26 51

2

```
function a(first,second)
{
return (second)(first);
}

a('Hello',function(e){alert(e+ ' world!');}); //=> Hello world
```

edited Jan 15 '14 at 16:21

answered Jan 15 '14 at 16:14

cochon
21 2

2

In fact, seems like a bit complicated, is not.

get method as a parameter:

```
 function JS_method(_callBack) {

           _callBack("called");

        }
```

You can give as a parameter method:

```
JS_method(function (d) {
       //Finally this will work.
       alert(d)
});
```

edited Oct 21 '14 at 7:42

answered Oct 14 '14 at 9:45

Hakkı Eser
259 2 11

1 Please, explain your answer. – MillaresRoo Oct 14 '14 at 10:16

Sorry. I've corrected – Hakkı Eser Oct 21 '14 at 7:45

2

The other answers do an excellent job describing what's going on, but one important "gotcha" is to make sure that whatever you pass through is indeed a reference to a function.

For instance, if you pass through a string instead of a function you'll get an error:

```
function function1(my_function_parameter){
    my_function_parameter();
}

function function2(){
 alert('Hello world');
}

function1(function2); //This will work

function1("function2"); //This breaks!
```

See JsFiddle

answered Mar 23 '15 at 20:54

Victor
416 4 17

0

Some time when you need to deal with event handler so need to pass event too as an argument , most of the modern library like react, angular might need this.

I need to override OnSubmit function(function from third party library) with some custom validation on reactjs and I passed the function and event both like below

ORIGINALLY

```
    <button className="img-submit" type="button"  onClick=
 {onSubmit}>Upload Image</button>
```

MADE A NEW FUNCTION `upload` and called passed `onSubmit` and event as arguments

```
upload(event,fn){
  //custom codes are done here
  fn(event);
}
```

edited Feb 20 '18 at 23:56

answered Feb 20 '18 at 23:50

sumit
9,269 9 38 84

-2

You can use a JSON as well to store and send JS functions.

Check the following:

```
var myJSON =
{
    "myFunc1" : function (){
        alert("a");
    },
    "myFunc2" : function (functionParameter){
        functionParameter();
    }
}



function main(){
    myJSON.myFunc2(myJSON.myFunc1);
}
```

This will print 'a'.

The following has the same effect with the above:

```
var myFunc1 = function (){
    alert('a');
}

var myFunc2 = function (functionParameter){
    myFunc2(myFunc1);
}
```

Which is also has the same effect with the following:

```
function myFunc1(){
    alert('a');
}


function myFunc2 (functionParameter){
    functionParameter();
}

function main(){
    myFunc2(myFunc1);
}
```

And a object paradigm using Class as object prototype:

```
function Class(){
    this.myFunc1 =  function(msg){
        alert(msg);
    }

    this.myFunc2 = function(callBackParameter){
        callBackParameter('message');
    }
}


function main(){
    var myClass = new Class();
    myClass.myFunc2(myClass.myFunc1);
}
```

edited May 9 '13 at 8:18

answered May 8 '13 at 16:18

gazgas
168 1 6