What is a callback function?

What is a callback function?

language-agnostic callback

edited May 5 '09 at 16:14
Cheekysoft
18.4k ● 14 ● 52 ● 72

asked May 5 '09 at 10:18
paul

17 Answers

Many are confused by what a callback is because of the name of the damned thing.

A callback method is one which is passed as an argument in another method and which is invoked after some kind of event. The 'call back' nature of the argument is that, once its parent method completes, the function which this argument represents is then called; that is to say that the parent method 'calls back' and executes the method provided as an argument.

Psuedocode:

```
// An innocuous looking method which will become known as a callback method
// because of the way in which we will invoke it.
function meaningOfLife() {
    log("The meaning of life is: 42");
}


// An innocuous looking method which just takes an int and prints it to screen, and
// takes a function reference to be executed when printANumber completes
function printANumber(int number, function callbackFunction()) {
    print("The number you provided is: " + number);
}

function event() {
   printANumber(6, meaningOfLife);
}
```

Result:

```
The number you provided is: 6
The meaning of life is: 42
```

Callbacks are so-called due to their usage with pointer languages. If you don't use one of those, don't labour over the name 'callback'. Just understand that it is just a name to describe a method that's supplied as an argument to another method, such that when the parent method is called (whatever condition, such as a button click, a timer tick etc) and its method body completes, the callback method is then invoked, or in other words "called at the back" of the other function.

edited Feb 26 at 13:22
user3663765
299 ● 1 ● 1 ● 14

answered Sep 26 '11 at 1:04
7SpecialGems
2,970 ● 2 ● 21 ● 42

3 Should not the `Print_A_Number(meaningOfLife);` be `Print_A_Number(meaningOfLife());` instead? Thanks. – Kraken Mar 13 '13 at 16:33

3 Your example is great but I don't see why the terminology is "callback". When is meaningOfLife "called back"? – Imray Apr 11 '13 at 14:51

2 callback is because of the name of the damned thing - cannot agree with you more. – JavaDeveloper Oct 28 '13 at 5:40

1 @RohitChatterjee Hi Rohit, thanks for your comment. I've cleaned up the language in the answer to try and better explain. I've also changed the example to use pseudocode. To everyone else: Rohit has provided a great answer of his own which commentates on the use of callbacks in present day functional/scripting languages, particuarly in an asynchronous context. – 7SpecialGems May 7 '14 at 13:57 ✎

1 So it's a "callback" because the function that is passed isn't evaluated AT PASSING and simply brought

into the called function as a simple return value, but is actually evaluated whenever the CALLED function CALLS BACK the passed function? – AllTradesJack Sep 9 '14 at 16:57

Note that callback is one word.

The wikipedia callback page explains it very well.

quote from wikipedia page:

> In computer programming, a callback is a reference to executable code, or a piece of executable code, that is passed as an argument to other code. This allows a lower-level software layer to call a subroutine (or function) defined in a higher-level layer.

edited Dec 2 '11 at 11:09

answered May 5 '09 at 10:19
danio
3,325 ● 2 ● 17 ● 39

9 Nice way to present an answer. – Chathuranga Chandrasekara May 5 '09 at 10:24

1 And this also leads to the answer in a different way. The noun "callback" is that which has been "called back", in the same way something that's gone through shutdown has been shut down and something that's used to log in is a login. – Anonymous May 5 '09 at 10:43

9 This could have been a comment - basically it's a link to Wikipedia – Imray Apr 11 '13 at 14:37

3 I don't rate this answer at all.. – adaam Mar 7 '14 at 13:37

Wikipedia has actually got some really awesome programming stuff in its troves. I always felt like the term "callback" was best explained using the phrase, "I am going to call back to..." – Thomas Feb 23 at 21:09

Opaque Definition

A callback function is a function you provide to another piece of code, allowing it to be called by that code.

Contrived example

Why would you want to do this? Let's say there is a service you need to invoke. If the service returns immediately, you just:

1. Call it
2. Wait for the result
3. Continue once the result comes in

For example, suppose the service were the `factorial` function. When you want the value of `5!`, you would invoke `factorial(5)`, and the following steps would occur:

1. Your current execution location is saved (on the stack, but that's not important)
2. Execution is handed over to `factorial`
3. When `factorial` completes, it puts the result somewhere you can get to it
4. Execution comes back to where it was in [1]

Now suppose `factorial` took a really long time, because you're giving it huge numbers and it needs to run on some supercomputing cluster somwhere. Let's say you expect it to take 5 minutes to return your result. You could:

1. Keep your design and run your program at night when you're asleep, so that you're not staring at the screen half the time
2. Design your program to do other things while `factorial` is doing its thing

If you choose the second option, then callbacks might work for you.

End-to-end design

In order to exploit a callback pattern, what you want is to be able to call `factorial` in the following way:

```
factorial(really_big_number, what_to_do_with_the_result)
```

The second parameter, `what_to_do_with_the_result`, is a function you send along to `factorial`, in the hope that `factorial` will call it on its result before returning.

Yes, this means that `factorial` needs to have been written to support callbacks.

Now suppose that you want to be able to pass a parameter to your callback. Now you can't, because you're not going to be calling it, `factorial` is. So `factorial` needs to be written to allow you to pass your parameters in, and it will just hand them over to your callback when it invokes it. It might look like this:

```
factorial (number, callback, params)
{
    result = number!   // i can make up operators in my pseudocode
    callback (result, params)
}
```

Now that `factorial` allows this pattern, your callback might look like this:

```
logIt (number, logger)
{
    logger.log(number)
}
```

and your call to `factorial` would be

```
factorial(42, logIt, logger)
```

What if you want to return something from `logIt`? Well, you can't, because `factorial` isn't paying attention to it.

Well, why can't `factorial` just return what your callback returns?

Making it non-blocking

Since execution is meant to be handed over to the callback when `factorial` is finished, it really shouldn't return anything to its caller. And ideally, it would somehow launch its work in another thread / process / machine and return immediately so that you can continue, maybe something like this:

```
factorial(param_1, param_2, ...)
{
    new factorial_worker_task(param_1, param_2, ...);
    return;
}
```

This is now an "asynchronous call", meaning that when you call it, it returns immediately but hasn't really done its job yet. So you do need mechanisms to check on it, and to obtain its result when its finished, and your program has gotten more complex in the process.

And by the way, using this pattern the `factorial_worker_task` can launch your callback asynchronously and return immediately.

So what do you do?

The answer is to stay within the callback pattern. Whenever you want to write

```
a = f()
g(a)
```

and `f` is to be called asynchronously, you will instead write

```
f(g)
```

where `g` is passed as a callback.

This fundamentally changes the flow-topology of your program, and takes some getting used to.

Your programming language could help you a lot by giving you a way to create functions on-the-fly. In the code immediately above, the function `g` might be as small as `print (2*a+1)`. If your language requires that you define this as a separate function, with an entirely unnecessary name and signature, then your life is going to get unpleasant if you use this pattern a lot.

If, on the other hand, you language allows you to create lambdas, then you are in much better shape. You will then end up writing something like

```
f( func(a) { print(2*a+1); })
```

which is so much nicer.

How to pass the callback

How would you pass the callback function to `factorial`? Well, you could do it in a number of ways.

1. If the called function is running in the same process, you could pass a function pointer
2. Or maybe you want to maintain a dictionary of `fn name --> fn ptr` in your program, in which case you could pass the name
3. Maybe your language allows you to define the function in-place, possible as a lambda! Internally it is creating some kind of object and passing a pointer, but you don't have to worry about that.
4. Perhaps the function you are calling is running on an entirely separate machine, and you are calling it using a network protocol like HTTP. You could expose your callback as an HTTP-callable function, and pass its URL.

You get the idea.

The recent rise of callbacks

In this web era we have entered, the services we invoke are often over the network. We often do not have any control over those services i.e. we didn't write them, we don't maintain them, we can't ensure they're up or how they're performing.

But we can't expect our programs to block while we're waiting for these services to respond. Being aware of this, the service providers often design APIs using the callback pattern.

JavaScript supports callbacks very nicely e.g. with lambdas and closures. And there is a lot of activity in the JavaScript world, both on the browser as well as on the server. There are even JavaScript platforms being developed for mobile.

As we move forward, more and more of us will be writing asynchronous code, for which this understanding will be essential.

answered May 4 '14 at 6:34
Rohit Chatterjee
603 ● 4 ● 8

I believe this "callback" jargon has been mistakenly used in a lot of places. My definition would be something like:

> A callback function is a function that you pass to someone and let them call it at some point of time.

I think people just read the first sentence of the wiki definition:

> a callback is a reference to executable code, or a piece of executable code, that is passed as an argument to other code.

I've been working with lots of APIs, see various of bad examples. Many people tend to name a function pointer (a reference to executable code) or anonymous functions(a piece of executable code) "callback", if they are just functions why do you need another name for this?

Actually only the second sentence in wiki definition reveals the differences between a callback function and a normal function:

> This allows a lower-level software layer to call a subroutine (or function) defined in a higher-level layer.

so the difference is who you are going to pass the function and how your passed in function is going to be called. If you just define a function and pass it to another function and called it directly in that function body, don't call it a callback. The definition says your passed in function is gonna be called by "lower-level" function.

I hope people can stop using this word in ambiguous context, it can't help people to understand better only worse.

edited Jul 20 '12 at 12:46 answered Jul 20 '12 at 10:58
Zane XY
694 ● 7 ● 4

1 Your answer makes sense... but I'm having trouble picturing it. Can you give an example? – Imray Apr 11 '13 at 14:56

1 @Zane Wong :: In the last you have written "The definition says your passed in function is gonna be called by "lower-level" function." Can you please explain what lower-level function indicates ? Its better if you give an example . – vivek Jan 23 '14 at 14:44

A callback function is one that should be called when a certain condition is met. Instead of being called immediately, the callback function is called at a certain point in the future.

Typically it is used when a task is being started that will finish asynchronously (ie will finish some time after the calling function has returned).

For example, a function to request a webpage might require its caller to provide a callback function that will be called when the webpage has finished downloading.

answered May 5 '09 at 16:07
Thomas Bratt
13.2k ● 21 ● 69 ● 97

In your first sentence, you say `"...when a condition is met"` but i thought callbacks are called when the parent function finishes executing and are not dependent of conditions (?). – ojonugwa ochalifu Jul 25 '14 at 8:59

The 'certain condition' just means they generally get called for a reason, rather than at random. A callback could be called when the parent/creator is still executing - this could lead to a race condition if the programmer is not expecting it. – Thomas Bratt Jul 26 '14 at 12:37

Okay.Thanks for the clarification – ojonugwa ochalifu Jul 26 '14 at 13:10

A layman response would be that it is a function that is not called by you but rather by the user or by the browser after a certain event has happened or after some code has been processed.

answered Sep 15 '11 at 16:48
NateH
221 ● 2 ● 2

6 +1 for "layman response" – CyprUS Apr 18 '12 at 7:36

cool enof :) good one – JavaDeveloper Oct 28 '13 at 5:42

Great explanation at javascriptissexy.com/…; which i will repost here; A callback function is a function that is passed to another function as a parameter, and the callback function is called or executed inside the otherFunction. //Note that the item in the click method's parameter is a function, not a variable. //The item is a callback function $("#btn_1").click(function() { alert("Btn 1 Clicked"); }); As you see in the preceding example, we pass a function as a parameter to the click method for it to execute – MarcoZen Apr 22 at 16:25

A callback function is a function you specify to an existing function/method, to be invoked when an action is completed, requires additional processing, etc.

In Javascript, or more specifically jQuery, for example, you can specify a callback argument to be called when an animation has finished.

In PHP, the `preg_replace_callback()` function allows you to provide a function that will be called when the regular expression is matched, passing the string(s) matched as arguments.

edited Mar 10 '11 at 5:38 answered May 5 '09 at 10:21

alex
214k ● 93 ● 534 ● 702

This makes callbacks sound like return statements at the end of methods.

I'm not sure that's what they are.

I think Callbacks are actually a call to a function, as a consequence of another function being invoked and completing.

I also think Callbacks are meant to address the originating invocation, in a kind of "hey! that thing you asked for? I've done it - just thought I would let you know - back over to you".

answered Oct 27 '10 at 23:18
Mauritico
81 ● 1 ● 1

+1 for questioning Callbacks vs Return statements. I used to get caught out by this and so do many graduates who I work with. – 7SpecialGems Sep 26 '11 at 0:38

1 Good answer - helped me understand it unlike many of the other answers! – adaam Mar 7 '14 at 13:38

Callbacks are most easily described in terms of the telephone system. A function call is analogous to calling someone on a telephone, asking her a question, getting an answer, and hanging up; adding a callback changes the analogy so that after asking her a question, you also give her your name and number so she can call you back with the answer. -- Paul Jakubik , "Callback Implementations in C++"

answered Jan 11 '14 at 1:13
DejanLekic
6,857 ● 1 ● 14 ● 33

The simple answer to this question is that a callback function is a function that is called through a function pointer. If you pass the pointer (address) of a function as an argument to another, when that pointer is used to call the function it points to it is said that a call back is made

answered Mar 10 '11 at 5:27
Zain Ali
4,805 ● 5 ● 44 ● 67

Call After would be a better name than the stupid name, **callback**. When or if condition gets met within a function, call another function, the **Call After** function, the one received as argument.

Rather than hard-code an inner function within a function, one writes a function to accept an already-written **Call After** function as argument. The **Call After** might get called based on state changes detected by code in the function receiving the argument.

edited Nov 19 '13 at 1:59
Michael Petrotta
38.8k ● 9 ● 89 ● 139

answered Nov 19 '13 at 1:40
Smack MacDougal
41 ● 2

Assume we have a function `sort(int *arraytobesorted,void (*algorithmchosen)(void))` where it can accept a function pointer as its argument which can be used at some point in `sort()` 's implementation . Then , here the code that is being addressed by the function pointer `algorithmchosen` is called as **callback function** .

And see the advantage is that we can choose any algorithm like:

```
1.    algorithmchosen = bubblesort
2.    algorithmchosen = heapsort
3.    algorithmchosen = mergesort   ...
```

Which were, say,have been implemented with the prototype:

```
1.   `void bubblesort(void)`
```

```
2.   `void heapsort(void)`
3.   `void mergesort(void)`   ...
```

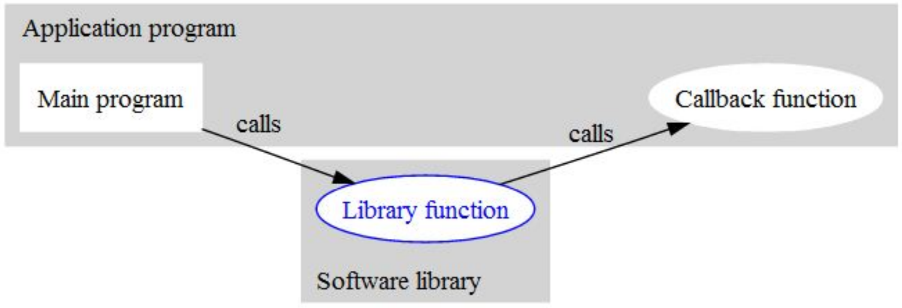
This is a concept used in achieving Polymorphism in Object Oriented Programming

edited May 4 '13 at 8:33

answered May 4 '13 at 6:29
E F
98 ● 10

Great explanation at javascriptissexy.com/…; which i will repost here; A callback function is a function that is passed to another function as a parameter, and the callback function is called or executed inside the otherFunction. //Note that the item in the click method's parameter is a function, not a variable. //The item is a callback function $("#btn_1").click(function() { alert("Btn 1 Clicked"); }); As you see in the preceding example, we pass a function as a parameter to the click method for it to execute – – MarcoZen Apr 22 at 16:27

look at the image :)



Main program calls library function (which might be system level function also) with callback function name. This callback function might be implemented in multiple way. The main program choose one callback as per requirement.

Finally, the library function calls the callback function during execution.

edited Aug 12 '14 at 7:06
Saumya Suhagiya
45 ● 10

answered Dec 23 '11 at 6:29
jeet.mg
341 ● 6 ● 20

7 Would you mind also adding a *text* explanation to this? If the image vanishes, this answer loses all context. – Tim Post ♦ Dec 23 '11 at 7:34

text from other people explains it the best. the only thing i felt is lacking is the image :) – jeet.mg Dec 23 '11 at 8:46

4 Mirror: en.wikipedia.org/wiki/File:Callback-notitle.svg – XP1 Feb 1 '12 at 5:16

This concept wasn't taught me in school, but when I started working I saw its being used at quite frequently at many places.
One important usage area is that you register one of your function as a handle (i.e. a callback) and then send a message / call some function to do some work or processing. Now after the processing is done, the called function would call our registered function (i.e. now call back is done), thus indicating us processing is done.
This wikipedia link explains quite well graphically.

answered May 5 '09 at 10:34
Chintan Parikh
392 ● 1 ● 2 ● 12

In the managed and unmanaged code scenario, the word "callback" may be easier to understand.
http://msdn.microsoft.com/en-us/library/d186xcf0(v=vs.110).aspx

answered Jan 12 at 3:46
Ross Gao
8 ● 2

Answers with reference to a foreign site are looked down upon on SO. One of the reasons is that the content of the answer depends on the state of the site being referenced and is subject to change or disappear. Please consider expanding your answer if possible. – Karolis Koncevičius Jan 12 at 4:11

A callback function, also known as a higher-order function, is a function that is passed to another function as a parameter, and the callback function is called (or executed) inside the parent function.

```
$("#button_1").c`enter code here`lick(function() {
  alert("button 1 Clicked");
});
```

Here we have pass a function as a parameter to the click method. And the click method will call (or execute) the callback function we passed to it.

answered Apr 26 at 20:21
ChiragK
3 ● 2

> A **callback function**, also known as a higher-order function, is a function that is passed to another function (let's call this other function “otherFunction”) as a parameter, and the callback function is called (or executed) inside the otherFunction.

Or

> A **callback function** is a function passed as a parameter to another function.

answered 15 hours ago
premraj
479 ● 7 ● 21