# Early exit from function?

Asked 9 years ago Active 1 year, 2 months ago Viewed 692k times

366

I have a function:

```
function myfunction() {
  if (a == 'stop')  // How can I stop the function here?
}
```

57

Is there something like `exit()` in JavaScript?

javascript

edited May 30 '18 at 18:04
Kara
4,144 10 45 53

asked Jul 25 '10 at 17:16
Simon
9,432 34 85 118

3 do you want to stop the execution or return? – Ken Struys Jul 25 '10 at 17:20

@Ken Struys what is the difference? as i understand, if i make return, it stops the execution? isn't it? – Simon Jul 25 '10 at 17:23

3 Well here's the thing, using a return will just return to the context of the calling function. If you actually want the exit semantic you want to stop execution, you could do something like this: vikku.info/codesnippets/javascript/… – Ken Struys Jul 25 '10 at 17:34

1 @Ken - That link you provided deals with stopping execution of a `for` loop. Even then, I have no idea why the method suggested would be used, when you could just call `break;` . To use the example from the article: `if(i==5) break;` Using `return` will halt the execution of the function, whether or not you're in a `for` loop. – user113716 Jul 25 '10 at 18:09

Syom - Yes, `return` will stop the execution of the function, which seems to be what you asked. – user113716 Jul 25 '10 at 18:11

## 11 Answers

You can just use `return` .

583

```
function myfunction() {
     if(a == 'stop')
         return;
}
```

This will send a return value of `undefined` to whatever called the function.

```
var x = myfunction();

console.log( x );  // console shows undefined
```

Of course, you can specify a different return value. Whatever value is returned will be logged to the console using the above example.

```
return false;
return true;
return "some string";
return 12345;
```

edited Jul 25 '10 at 17:25

answered Jul 25 '10 at 17:20

user113716

**269k** 59 406 423

4 I know this is an old post, and this is common practice BUT I think this is a bad solution. If the function is SUPPOSED to return a value, you should use return. If you just blindly use return you might run into problems later. Especially if you start binding events that have a return in them. Check out this post for more info: fuelyourcoding.com/jquery-events-stop-misusing-return-false – user603284 Jul 14 '11 at 21:53

59 @dbme: Functions in JavaScript *always* return. The return statement is implicit if it hasn't been provided. The default return value is `undefined` , and that's what my primary solution provides. The article you referenced is talking about doing `return false` in a jQuery event handler. That's an entirely different issue. This is the proper way to exit a JavaScript function. Obviously if the caller relies on the value returned, the value needs to be defined appropriately. – user113716 Jul 14 '11 at 22:03

3 ...An implicit variation would be `if(a != 'stop') { /* run my code */ }` so that the code only runs when `a` doesn't equal `'stop'` without providing an explicit `return` . But the return value *is identical to my solution*. In both cases, `undefined` will be returned. – user113716 Jul 14 '11 at 22:06

3 Awesome response. I didn't realize there was any difference between returning undefined vs false vs a different value. I've been trying to find a definitive answer regarding this behavior for the past hour. So is it safe to say (to reiterate your point) that return is a 100% safe way to exit a method, even if the caller is bound to events etc? – user603284 Jul 14 '11 at 22:11

@dbme: Well, it all depends on what is expected by the method calling the function. If some code library defines an event system that will break if it receives `undefined` (or it doesn't receive some other value) as a return value, then you'll need to conform to the specification of that API, and return the correct value. Generally for an event handling system, it will expect `undefined` as a simple indication that the handler has finished, and there are

no further instructions. With jQuery, `return false;` has special meaning giving instruction to do a `preventDefault` and `stopPropagation`. – user113716 Jul 14 '11 at 22:19

47

Apparently you can do this:

```
function myFunction() {myFunction:{
    console.log('i get executed');
    break myFunction;
    console.log('i do not get executed');
}}
```

See block scopes through the use of a label: https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Statements/label

I can't see any downsides yet. But it doesn't seem like a common use.

Derived this answer: JavaScript equivalent of PHP's die

edited May 23 '17 at 12:03
Community ♦
1 1

answered Mar 8 '14 at 10:34

CMCDragonkai
3,273 6 39 69

is it possible to use this solution with nodejs exports module? when I try it I get "not used label" error. exports.MyFunction = function(data){myFunction: {break myFunction;}} – Yuri Almeida Feb 1 '17 at 16:12

17

This:

```
function myfunction()
{
     if (a == 'stop')  // How can I stop working of function here?
     {
         return;
     }
}
```

answered Jul 25 '10 at 17:20
Rob


38.3k 22 104 132

16

```
function myfunction() {
     if(a == 'stop')
         return false;
}
```

`return false;` is much better than just `return;`

edited Jan 31 '18 at 19:35

user664833
11.2k 17 73 114

answered Feb 1 '13 at 2:54

xlaok
351 2 8

12 Why is `false` better? I'd say the default `undefined` is better in the generic case. Either way, you're correct to say it's often better to return a meaningful value. – Brad Koch Apr 30 '13 at 15:06

It's up to the programmer and dependant on the use case. For example, a function might validate something, so if the validation fails it makes more sense to return `false` than `undefined` . – Adam McArthur Oct 1 '14 at 4:06

10

Using a little different approach, you can use `try catch` , with throw statement.

```
function name() {
    try {
        ...

        //get out of here
        if (a == 'stop')
            throw "exit";

        ...
    } catch (e) {
        // TODO: handle exception
    }
}
```

answered Feb 26 '16 at 13:58
Rodney Salcedo


1,046 16 20

3

if you are looking for a script to avoid submitting form when some errors found, this method should work

```
function verifyData(){
     if (document.MyForm.FormInput.value.length == "") {
          alert("Write something!");
     }
     else {
          document.MyForm.submit();
     }
}
```

change the **Submit Button** type to "button"

```
<input value="Save" type="button" onClick="verifyData()">
```

hope this help.

edited Oct 9 '12 at 9:41

answered Oct 9 '12 at 8:08


Awal Istirdja
47 4

3

Using a `return` will stop the function and return `undefined`, or the value that you specify with the return command.

```
function myfunction(){
    if(a=="stop"){
        //return undefined;
        return; /** Or return "Hello" or any other value */
    }
}
```

edited May 1 '16 at 3:43


4castle
23.2k 6 48 77

answered Apr 30 '16 at 17:47


Seizefire
81 7

0

I dislike answering things that aren't a real solution...

...but when I encountered this same problem, I made below workaround:

```
function doThis() {
  var err=0
  if (cond1) { alert('ret1'); err=1; }
  if (cond2) { alert('ret2'); err=1; }
  if (cond3) { alert('ret3'); err=1; }
  if (err < 1) {
    // do the rest (or have it skipped)
  }
}
```

Hope it can be useful for anyone.

answered Nov 5 '16 at 12:29

Leo
**2,225** 2 14 15

Duh. That's so simple I never thought of it!! Pretty useful to avoid big nested IF's. Thanks. – Debbie A Mar 27 at 19:20

This would be better with a switch. – Nazka Apr 30 at 12:57

Depends if you wish to have all error alerts at once, of be notified of just one. However each condition needs to be executed anyway. – Leo May 1 at 11:10

-3

exit(); can be use to go for the next validation.

answered Aug 9 '16 at 20:15

Chirag Dineshbhai Ponda
**5**

Unclear and faulty answer. – Frank Conijn Jun 5 '18 at 10:06

If you are using jquery. This should stop the function from bubbling up to so the parent function calling this should stop as well.

-4

```
function myfunction(e)
{
     e.stopImmediatePropagation();
     ..............
}
```

answered Oct 29 '12 at 21:02

silvster27

**1,375** 4 26 38

6 `stopImmediatePropagation()` is not a jQuery thing, and stopping propagation is not the same thing as exiting a function. – Brad Koch Apr 30 '13 at 15:01

type any random command that throws an error, for example:

-11

```
exit
```

or

```
die:-)
```

answered Sep 15 '16 at 2:12

amrali

**1**

This may stop all code from being executed, not only the rest of the function. Check fiddle: jsfiddle.net/b3k0xo7n/1 – treecon Jun 19 '17 at 21:13

1 What a terrible programming approach. How is this intuitive for another developer who works on the same codebase? – osullic Jun 26 '18 at 15:07