# signal-exit

When you want to fire an event no matter how a process exits:

* reaching the end of execution.
* explicitly having process.exit(code) called.
* having process.kill(pid, sig) called.
* receiving a fatal signal from outside the process

Use signal-exit.

// Hybrid module, either works

import { onExit } from 'signal-exit'

// or:

// const { onExit } = require('signal-exit')

onExit((code, signal) => {

console.log('process exited!', code, signal)

})

## API

remove = onExit((code, signal) => {}, options)

The return value of the function is a function that will remove the handler.

Note that the function *only* fires for signals if the signal would cause the process to exit. That is, there are no other listeners, and it is a fatal signal.

If the global process object is not suitable for this purpose (ie, it's unset, or doesn't have an emit method, etc.) then the onExit function is a no-op that returns a no-op remove method.

### Options

* alwaysLast: Run this handler after any other signal or exit handlers. This causes process.emit to be monkeypatched.

### Capturing Signal Exits

If the handler returns an exact boolean true, and the exit is a due to signal, then the signal will be considered handled, and will *not* trigger a synthetic process.kill(process.pid, signal) after firing the onExit handlers.

In this case, it your responsibility as the caller to exit with a signal (for example, by calling process.kill()) if you wish to preserve the same exit status that would otherwise have occurred. If you do not, then the process will likely exit gracefully with status 0 at some point, assuming that no other terminating signal or other exit trigger occurs.

Prior to calling handlers, the onExit machinery is unloaded, so any subsequent exits or signals will not be handled, even if the signal is captured and the exit is thus prevented.

Note that numeric code exits may indicate that the process is already committed to exiting, for example due to a fatal exception or unhandled promise rejection, and so there is no way to prevent it safely.

### Browser Fallback

The 'signal-exit/browser' module is the same fallback shim that just doesn't do anything, but presents the same function interface.

Patches welcome to add something that hooks onto window.onbeforeunload or similar, but it might just not be a thing that makes sense there.