Best Practices

Felix Geisendörfer

Munich Node.js User Group 01.12.2011 (v1)

(@)felixge(.de)

Twitter / GitHub / IRC

Felix Geisendörfer

(Berlin, Germany)

Callbacks

A terrible Example:

```
var fs = require('fs');

function readJSON(path, cb) {
  fs.readFile(path, 'utf8', function(err, data) {
    cb(JSON.parse(data));
  });
}
```

If you’re function works like this, I will not use it

Error Delegation

```
var fs = require('fs');

function readJSON(path, cb) {
  fs.readFile(path, 'utf8', function(err, data) {
    if (err) return cb(err);

    cb(JSON.parse(data));
  });
}
```

Otherwise JSON.parse(undefined) -> SyntaxError: Unexpected token ILLEGAL

Exception Handling

```
var fs = require('fs');

function readJSON(path, cb) {
  fs.readFile(path, 'utf8', function(err, data) {
    if (err) return cb(err);

    try {
      cb(JSON.parse(data));
    } catch (err) {
      cb(err);
    }
  });
}
```

Otherwise invalid JSON -> SyntaxError: Unexpected token ILLEGAL

Error parameter first

```
var fs = require('fs');

function readJSON(path, cb) {
  fs.readFile(path, 'utf8', function(err, data) {
    if (err) return cb(err);

    try {
      cb(null, JSON.parse(data));
    } catch (err) {
      cb(err);
    }
  });
}
```

Otherwise I always have to -> if (json instanceof Error) ...

Not your error

```
var fs = require('fs');

function readJSON(path, cb) {
  fs.readFile(path, 'utf8', function(err, data) {
    if (err) return cb(err);

    try {
      var json = JSON.parse(data);
    } catch (err) {
      return cb(err);
    }

    cb(null, json);
  });
}
```

Subtle: Do not catch errors inside your callback -> very unexpected results

Another common mistake

```
function readJSONFiles(files, cb) {
  var results   = {};
  var remaining = files.length;

  files.forEach(function(file) {
    readJSON(file, function(err, json) {
      if (err) return cb(err);

      results[file] = json;
      if (!--remaining) cb(null, results);
    });
  });
}
```

Only call back once

```
function readJSONFiles(files, cb) {
  var results   = {};
  var remaining = files.length;

  files.forEach(function(file) {
    readJSON(file, function(err, json) {
      if (err) {
        cb(err);
        cb = function() {};
        return;
      }

      results[file] = json;
      if (!--remaining) cb(null, results);
    });
  });
}
```

Nested Callbacks

```
db.query('SELECT A ...', function() {
  db.query('SELECT B ...', function() {
    db.query('SELECT C ...', function() {
      db.query('SELECT D ...', function() {
        db.query('SELECT E ...', function() {
          db.query('SELECT F ...', function() {
            db.query('SELECT G ...', function() {
              db.query('SELECT H ...', function() {
                db.query('SELECT I ...', function() {

                });
              });
            });
          });
        });
      });
    });
  });
});
```

iPhone 4 owners?

You are holding it wrong!

Fake photograph btw.: http://www.snopes.com/photos/politics/bushbook.asp

Just kidding

Nested callbacks are an actual concern in the node community

Control Flow Libs

```
var async = require('async');

async.series({
  queryA: function(next) {
    db.query('SELECT A ...', next);
  },
  queryB: function(next) {
    db.query('SELECT A ...', next);
  },
  queryC: function(next) {
    db.query('SELECT A ...', next);
  }
  // ...
}, function(err, results) {

});
```

Split code into more methods

Maybe node.js is not a good tool for your problem?

Exceptions

(Dealing with Death)

Exceptions

```
throw new Error('oh no');
```

Explicit, useful to crash your program

Please DO NOT EVER use exceptions for control flow.

Like so many things in JS, JSON.parse() throwing exceptions is bullshit.

Exceptions

```
node.js:134
        throw e; // process.nextTick error, or 'error' event on first tick
        ^
Error: oh no
    at Object.<anonymous> (/Users/felix/explicit.js:1:69)
    at Module._compile (module.js:411:26)
    at Object..js (module.js:417:10)
    at Module.load (module.js:343:31)
    at Function._load (module.js:302:12)
    at Array.<anonymous> (module.js:430:10)
    at EventEmitter._tickCallback (node.js:126:26)
```

Output on stderr

Exceptions

```
function MyClass() {}

MyClass.prototype.myMethod = function() {
  setTimeout(function() {
    this.myOtherMethod();
  }, 10);
};

MyClass.prototype.myOtherMethod = function() {};

(new MyClass).myMethod();
```

Implicit, usually undesirable / bugs

Exceptions

```
/Users/felix/implicit.js:5
    this.myOtherMethod();
         ^
TypeError: Object #<Object> has no method 'myOtherMethod'
    at Object._onTimeout (/Users/felix/implicit.js:5:10)
    at Timer.callback (timers.js:83:39)
```

Incomplete stack trace : (

Global Catch

```
process.on('uncaughtException', function(err) {
  console.error('uncaught exception: ' + err.stack);
});
```

Also known as Stage 3 / Bargaining stage in Kübler-Ross model of “The Five Stages of Grief”

Please be careful with this!

Global catch gone wrong

```
// Deep inside the database driver you are using
cb(null, rows);

this._executeNextQueuedQuery();

// In your code
db.query('SELECT ...', function(err, rows) {
  console.log('First row: ' + row[0].id);
});
```

If rows.length === 0 the exception triggered from accessing rows[0].id will STALL your database driver. Not good!

If you have to:

```
process.on('uncaughtException', function(err) {
  // You could use node-airbake for this
  sendErrorToLog(function() {
    // Once the error was logged, kill the process
    console.error(err.stack);
    process.exit(1);
  });
});
```

Even Better

Processes die.

Accept it.

Deal with it on a higher level.

Higher level could be upstart / monit / etc.

Deployment

Novice

```
$ node server.js
```

Advanced Beginner

```
$ node server.js &
```

Competent

```
$ screen
$ node server.js
```

Proficient

```
#!/bin/bash

while :
do
  node server.js
  echo "Server crashed!"
  sleep 1
done
```

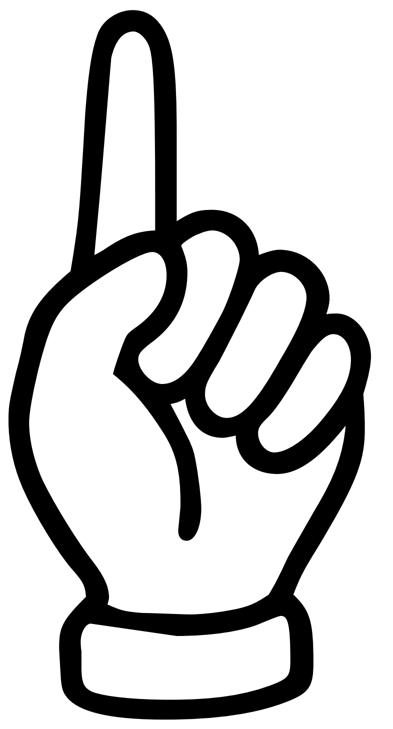
Expert

```
#!upstart

description "myapp"
author      "felix"

start on (local-filesystems and net-device-up
IFACE=eth0)
stop  on shutdown

respawn                    # restart when job dies
respawn limit 5 60         # give up restart after 5
respawns in 60 seconds

script
  exec sudo -u www-data /path/to/server.js >> /
var/log/myapp.log 2>&1
end script
```

Innovation

```
$ git push joyent master
$ git push nodejitsu master
$ git push heroku master
```

Questions?



@felixge

Thank you!