the NODE FIRM

CALLBACKS

WHAT ARE CALLBACKS

functions that are called once a unit of work is complete

```
function callback() {
  console.log('tick');
}

setTimeout(callback, 1000);
```

NODE CALLBACKS

```
someAsyncFunction('arg1', 2, function(err, result) {
  // handle error
  // if no error, process result
});
```

- callback last
- error as the first argument

READ A FILE

01_read_file.js

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

function fileReadFinished(err, contents) {
  if (err) {
    console.log('File read error:', err);
  } else {
    console.log(contents);
  }
}

fs.readFile(__filename, 'utf8', fileReadFinished);
```

RESOLVE A HOSTNAME

02_dns_lookup.js

```
var dns = require('dns');
var hostname = process.argv[2] || 'google.com';

dns.lookup(hostname, function(err, address, family) {
  if (err) {
    console.log('error looking up', err.message);
  } else {
    console.log(hostname, 'resolved to %s (IPv%s)', address, family);
  }
});
```

FLOW CONTROL

Asynchronous functions can be run in serial or in parallel

SERIAL EXECUTION

perform I/O operations, each depending on the previous

03_serial.js

```
var fs = require('fs');
var maxFileSize = parseInt(process.argv[2], 10) || 400;

fs.stat(__filename, function(err, stats) {
  if (err) throw err;

  if (stats.size < maxFileSize) {
    fs.readFile(__filename, 'utf8', function(err, contents) {
      if (err) throw err;

      console.log(contents);
    });
  } else {
    console.log("Not reading the file, it's too big");
  }
});
```

You can try to run this:

```
$ node 03_serial.js 1000
```

and

```
$ node 03_serial.js 100
```

SERIAL EXECUTION: FLATTENED

04_serial_flat.js

```javascript
var fs = require('fs');
var maxFileSize = parseInt(process.argv[2], 10) || 400;

function statDone(err, stats) {
  if (err) throw err;

  if (stats.size < maxFileSize) {
    fs.readFile(__filename, 'utf8', readFileDone);
  } else {
    console.log("Not reading the file, it's too big");
  }
}

function readFileDone(err, contents) {
  if (err) throw err;

  console.log(contents);
}

fs.stat(__filename, statDone);
```

Minimizes the indentation level, but makes it harder to follow

PARALLEL EXECUTION

perform multiple I/O operations, at the same time

05_parallel.js

```
var get = require('http').get;
var terms = ['MSFT', 'AAPL', 'RHT'];
var count = 0;

terms.forEach(function(term) {
  get('http://finance.google.com/finance/info?client=ig&q=' + term, func

    var responseBody = '';

    // Collect response body
    res.on('data', function(d) {
      responseBody += d.toString();
    });

    res.once('end', function() {
      var obj = JSON.parse(responseBody.replace('//',''));
      console.log(term, '-', obj[0].l);
      if (++ count === terms.length) {
        console.log('done!');
      }
    });
  });
});
```

SUMMARY

- Callback is a JavaScript function
- Callbacks are only called once
- Callback is the last argument in an async call
- Error is the first callback argument