Department of Computing
COVENTRY UNIVERSITY

304CEM

Writing Robust Code

Coventry University

Outcomes

Modules
Linters
Debuggers
Documentation

Introduction

Lots of important tools and techniques
Need to apply these to successfully complete your work
Increase your chances of gaining a much higher grade.

Modules

Modules

Beginners often put all their code into a single file.

This:

- Creates a huge file!
- Makes it difficult to find stuff
- Makes it impossible to write unit tests...

Modules in NodeJS

NodeJS implements the CommonJS format
This is part of the ECMA6 standard
You should use these to organise your code

System Architecture

Implement the MVC pattern

- The route file is your controller
- The modules will be your model
- There is no view...

Organising Modules

Identify your API collections
One module per collection
Minimise module dependencies
Identify shared functionality (e.g. auth, settings)
Put these in their own modules

CommonJS

By default all functions and vars are private
Uses an object called `exports`
Any property assigned to this object is exported
These can then be imported into another script

CommonJS Module Example

```
// shopping.js
var data = new Map()

exports.count = () => {
    return data.size
}
```

Importing a CommonJS Module

```
const list = require('./shopping')

const size = list.count()
console.log(`there are ${size} items`)
```

Linters

The LINT tool

An old program that flagged suspicious constructs in C language source code.
Would go through the C code
Identify problems before it was compiled, linked, and run
It was a static checker (didn't run the code)

Linters

Programs that can be run by a developer
Reads the source code
Checks it for programmatic and stylistic errors.
Helps to identify many common mistakes
Ensures:

- You avoid using some 'buggy' language constructs
- The style of your code is consistent.
- Forces you to use a safe 'subset' of the language

Linting JavaScript

Very important to lint JavaScript code!
Many different language features, styles and techniques
Many should be avoided to prevent potential bugs

JavaScript Linters

Many different JavaScript linters
Original one by Doug Crockford (JSLint)
Very opinionated about what was 'good' JS
Forced programmers to use his style!
In this module we use ESLint
More powerful and completely configurable
Plug-ins available for most IDEs (and Cloud9)

Debuggers

What is a Debugger?

Software used to test and debug your programs.
Contains some key features to support you:

- Pausing the program at a defined line (breakpoint)
- Executing the code line by line (single-stepping)
- Tracking the values of variables

Stepping

Step Over

Execute the next line of code and move to the line directly below

Step In

Execute the next line by dropping into the function

Step Out

Run to the end of the current function then pass control to its caller

Continue Execution

Run the program to the next breakpoint (or the end of the program)

Types of Debugger

Debuggers fall into two main categories:

- Terminal debuggers
  - Run through the terminal
  - Need to learn the debugger commands
- Visual Debuggers
  - Integrated into the IDE
  - Mouse-driven interface

Terminal Debugger

NodeJS offers a terminal debugger
Breakpoints are added directly to the code
Script is run in debug mode
Can add watchers to monitor variables
Commands for stepping through code.

Visual Debugger

Cloud9 offers a visual debugger
Breakpoints added in editor by clicking in left gutter
Buttons for step-over, step-in, step-out, continue
Editor provides pane to view variables

Documentation Generators

Documentation Generators

Programming tools
Generate software documentation from a source code files.

JSDoc

A markup language
Based on Javadoc
Used to annotate JavaScript code.
Batch processed by a special tool
Generates a HTML website
Contains the human-readable documentation

Example

```
/**
 * Returns details for the named item.
 * @param   {string} item - The item name
 * @returns {string} The name of the item
 * @throws  {InvalidItem} item not in list
 */
```

Generated HTML Documentation

Home

Home

Modules

shopping

(static) getItem(item) → {string}

Returns details for the named item.

Parameters:

Name	Type	Description
item	string	The item name to retrieve.

Source: shopping.js, line 54

Throws:

item not in list.

Type
InvalidItem

Returns:

The name of the item

Type
string

JSDoc Reference

http://usejsdoc.org

NodeJS Modules

Coventry University

NodeJS Modules

Follows CommonJS standard
This forms part of the latest ECMA6 standard

Simple approach to writing and importing functionality
Keeps private data private

Using Modules

```
exports.command = function(data) {
  // do stuff...
}

var todo = require('./todo.js')
todo.command('Hello World!')
```

Modules

Like any good language, JavaScript has lots of useful libraries and frameworks available
Many of these are specifically designed to work with Node on a server:
To install, manage, and uninstall any packages that you would like to use, Node ships with NPM: the Node Package Manager.

Node Package Manager

Node has a large package/library ecosystem
Useful for common functionality - just use a pre-written module
Modules are managed by the "Node Package Manager" NPM

https://www.npmjs.com

Installing Modules

To install a module, use the npm command

```
npm install module_name --save
```

`--save` adds details to `package.json`

Developer Modules

Some packages are to support development
Eslint, jsdoc, etc.
Not needed to run the API on the server
Added using the `--save-dev` flag

```
npm install module_name --save-dev
```

Different section in `package.json`

Saved Dependencies

```
"dependencies": {
  "request": "^2.67.0"
},
"devDependencies": {
  "eslint": "^3.1.0"
}
```

Publishing NodeJS Modules

Coventry University

The Problem

Importing modules you have created has a problem
Correct path needs to be provided
Can get messy:

```
const async = require('../../mod/async')
```

We want:

```
const async = require('async')
```

Local Module Dependencies

You have already imported modules from npmjs
You can also define and import local modules

```
npm install --save ./mod/async
```

This will copy your module into `node_modules`

Package File

```
{
  "name": "bookshop",
  "dependencies": {
    "request": "^2.67.0",
    "async": "file:local_modules/async",
  }
}
```

What is a Module?

Any directory that has a `package.json` file
Everything in the directory will be included
Files in the `.gitignore` or `.npmignore` file will not be published

Reloading a Module

There are several approaches:

- Use the `rm-local-modules` command
- Increment the version number in its `package.json` then `npm install`
- Delete and reload

```
rm -rf node_modules/async
npm install
```

Logging in to NPMJS

Need an account and to be logged in

```
npm adduser
npm login
```

Check account details

```
npm config ls
https://www.npmjs.com/~
```

Publishing to NPMJS

Package name must not exist in npmjs

```
npm publish
```