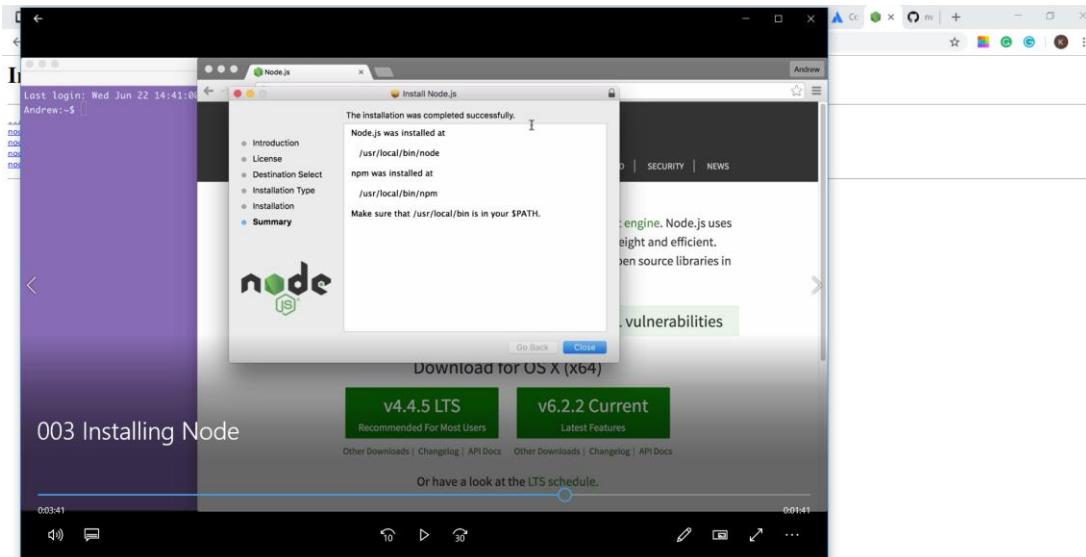


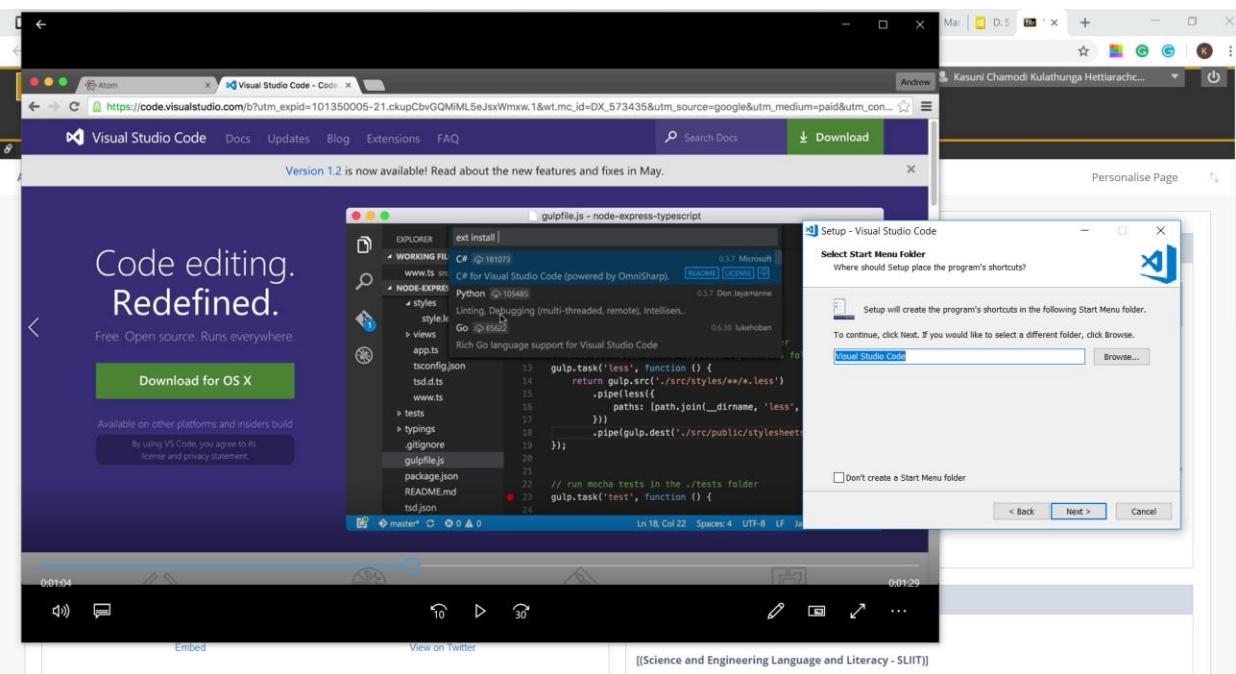
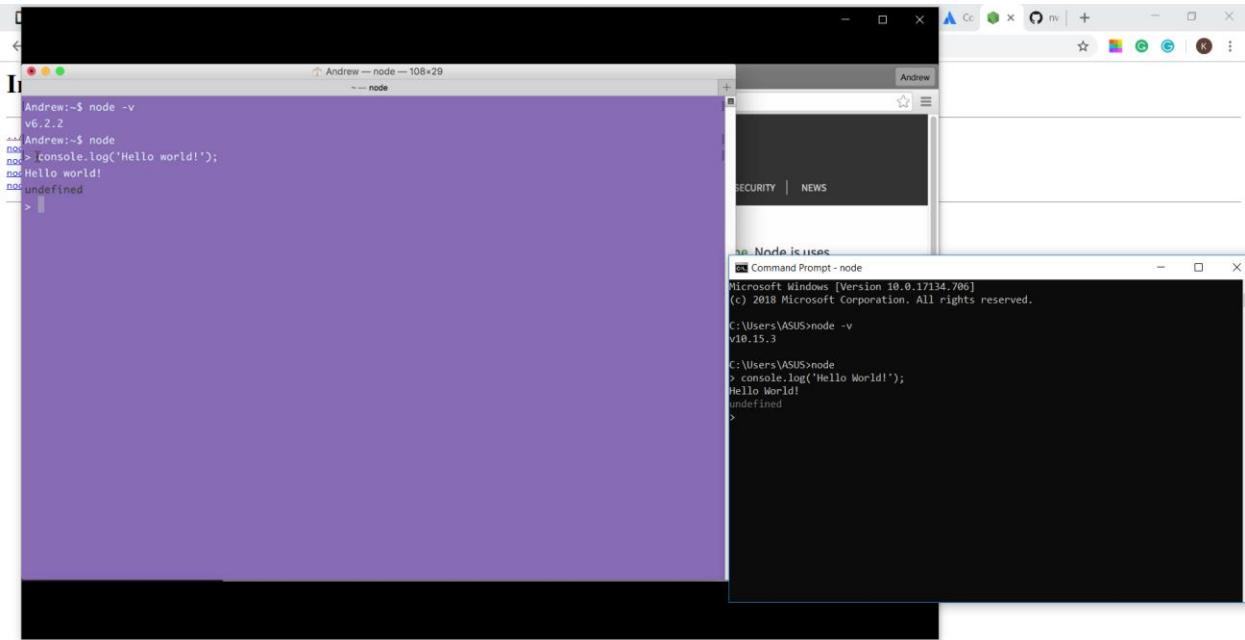
## 2) Installing Node

A screenshot of a Mac desktop showing three windows:

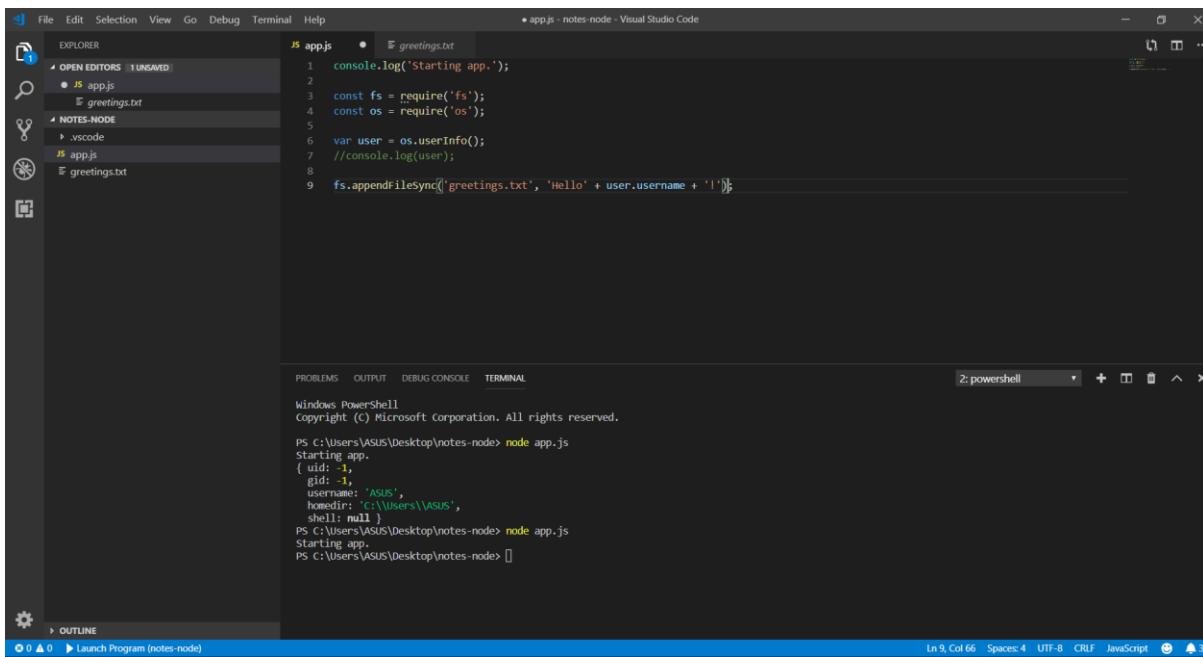
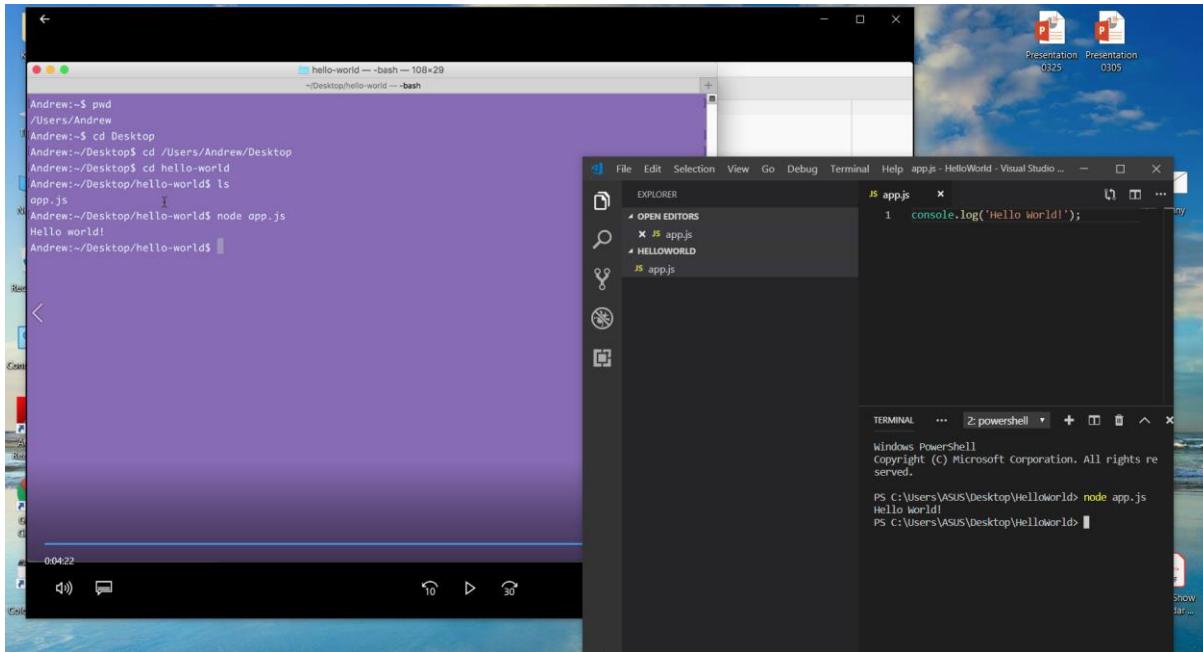
- Terminal:** Shows the command `node --version` being run, resulting in the output `v6.2.2`.
- Browser:** Shows a news article from BBC News about security.
- Command Prompt:** Shows the Node.js source code for the `process` module, specifically the `internal/promises` file.



## What is Node? & Installing VS Code



## Hello World! – first program



### 3) Using Require

The screenshot shows a Mac desktop environment. On the left, a terminal window displays the following log:

```
Last login: Tue Jun 28 09:35:45 on ttys000
Andrew:~ Desktop$ cd Desktop
Andrew:~/Desktop$ mkdir notes-node
Andrew:~/Desktop$ cd notes-node
Andrew:~/Desktop/notes-node$ node app.js
Starting app.
Andrew:~/Desktop/notes-node$ node app.js
Starting app.
{
  uid: 501,
  gid: 20,
  username: 'Andrew',
  homedir: '/Users/Andrew',
  shell: '/bin/bash'
}
Andrew:~/Desktop/notes-node$ node app.js
Starting app.
Andrew:~/Desktop/notes-node$
```

On the right, a Visual Studio Code window is open with two files:

- app.js**:

```
1 console.log('Starting app.');
2
3 const fs = require('fs');
4 const os = require('os');
5
6 var user = os.userInfo();
7
8 fs.appendFile('greetings.txt', `Hello ${user.username}!`);
```
- greetings.txt**:

```
HelloASUS!
```

The status bar at the bottom indicates "0:13:04" and "app.js 8:33".

The screenshot shows a Windows desktop environment. On the left, a terminal window displays the following log:

```
Last login: Tue Jun 28 09:35:45 on ttys000
Andrew:~ Desktop$ cd Desktop
Andrew:~/Desktop$ mkdir notes-node
Andrew:~/Desktop$ cd notes-node
Andrew:~/Desktop/notes-node$ node app.js
Starting app.
{
  uid: 501,
  gid: 20,
  username: 'Andrew',
  homedir: '/Users/Andrew',
  shell: '/bin/bash'
}
Andrew:~/Desktop/notes-node$ node app.js
Starting app.
Andrew:~/Desktop/notes-node$
```

On the right, a Visual Studio Code window is open with two files:

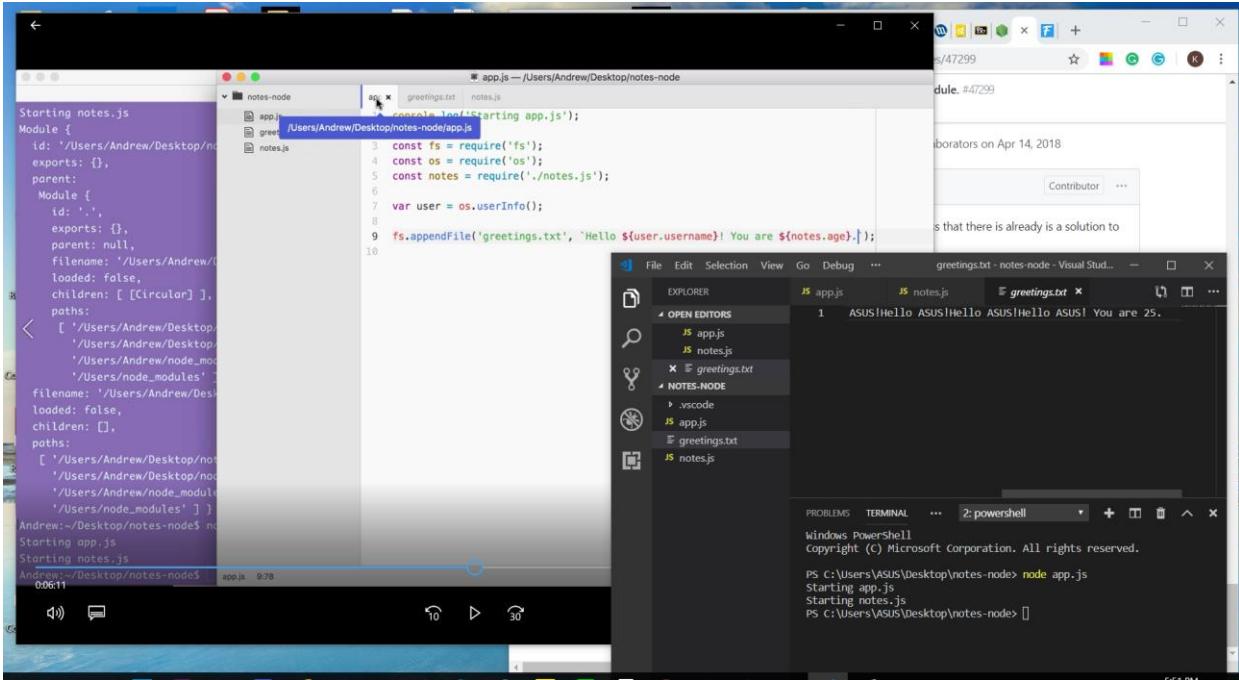
- app.js**:

```
1 console.log('Starting app.');
2
3 const fs = require('fs');
4 const os = require('os');
5
6 var user = os.userInfo();
7
8 fs.appendFile('greetings.txt', `Hello ${user.username}!`);
```
- greetings.txt**:

```
HelloASUS!
```

The status bar at the bottom indicates "0:14:07" and "app.js 3:16".

## Requiring your own files

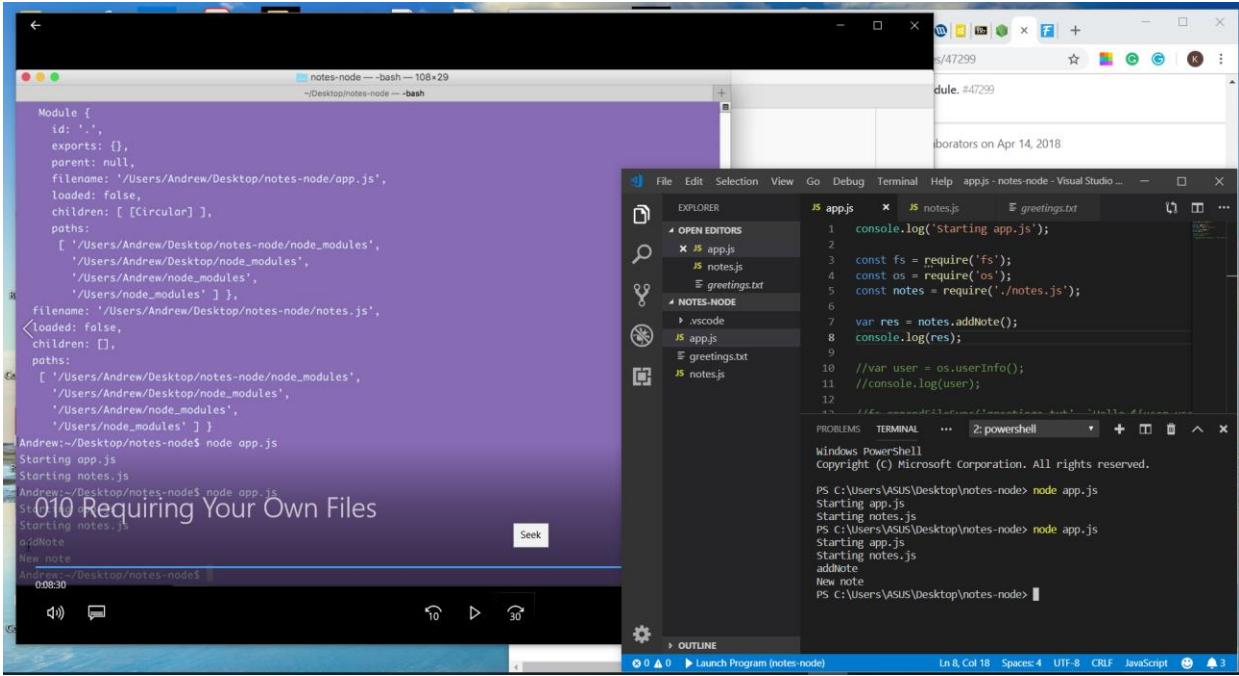


```
Starting notes.js
Module {
  id: '/Users/Andrew/Desktop/notes-node/app.js',
  exports: {},
  parent: null,
  filename: '/Users/Andrew/Desktop/notes-node/app.js',
  loaded: false,
  children: [ [Circular] ],
  paths: [
    '/Users/Andrew/Desktop/notes-node/app.js',
    '/Users/Andrew/Desktop/notes-node/app.js',
    '/Users/Andrew/node_modules',
    '/Users/node_modules'
  ],
  filename: '/Users/Andrew/Desktop/notes-node/app.js',
  loaded: false,
  children: [],
  paths: [
    '/Users/Andrew/Desktop/notes-node/app.js',
    '/Users/Andrew/Desktop/notes-node/app.js',
    '/Users/Andrew/node_modules',
    '/Users/node_modules'
  ]
}
Andrew:~/Desktop/notes-node$ node app.js
Starting app.js
Starting notes.js
Andrew:~/Desktop/notes-node$ 008:11
```

```
const fs = require('fs');
const os = require('os');
const notes = require('./notes.js');

var user = os.userInfo();
fs.appendFile('greetings.txt', `Hello ${user.username}! You are ${notes.age}.`);

ASUSHello ASUSHello ASUSHello ASUSHello ASUS! You are 25.
```



```
notes-node -- bash -- 108x29
Module {
  id: '/',
  exports: {},
  parent: null,
  filename: '/Users/Andrew/Desktop/notes-node/app.js',
  loaded: false,
  children: [ [Circular] ],
  paths: [
    '/Users/Andrew/Desktop/notes-node/node_modules',
    '/Users/Andrew/Desktop/node_modules',
    '/Users/Andrew/node_modules',
    '/Users/node_modules'
  ],
  filename: '/Users/Andrew/Desktop/notes-node/notes.js',
  loaded: false,
  children: [],
  paths: [
    '/Users/Andrew/Desktop/notes-node/node_modules',
    '/Users/Andrew/Desktop/node_modules',
    '/Users/Andrew/node_modules',
    '/Users/node_modules'
  ]
}
Andrew:~/Desktop/notes-node$ node app.js
Starting app.js
Starting notes.js
Andrew:~/Desktop/notes-node$ 008:30
```

```
console.log('Starting app.js');

const fs = require('fs');
const os = require('os');
const notes = require('./notes.js');

var res = notes.addnote();
console.log(res);

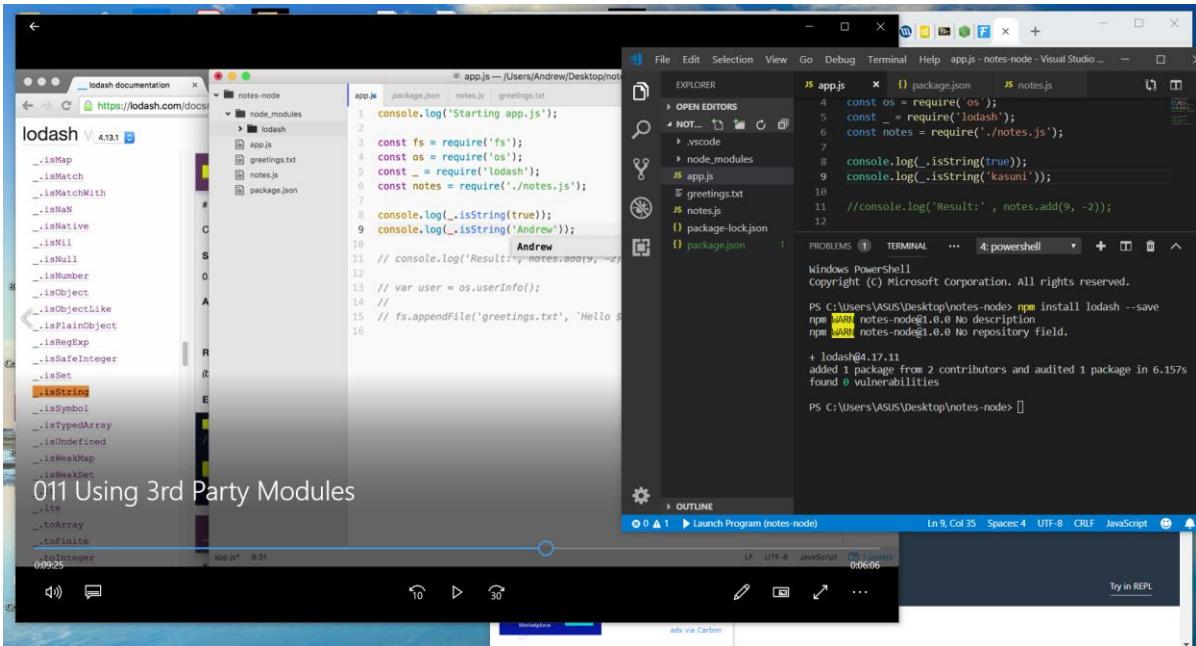
//var user = os.userInfo();
//console.log(user);

010 Requiring Your Own Files
```

```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

PS C:\Users\ASUS\Desktop\notes-node> node app.js
Starting app.js
Starting notes.js
PS C:\Users\ASUS\Desktop\notes-node> node app.js
Starting app.js
Starting notes.js
addnote
New note
New note
PS C:\Users\ASUS\Desktop\notes-node>
```

## Using 3<sup>rd</sup> party modules

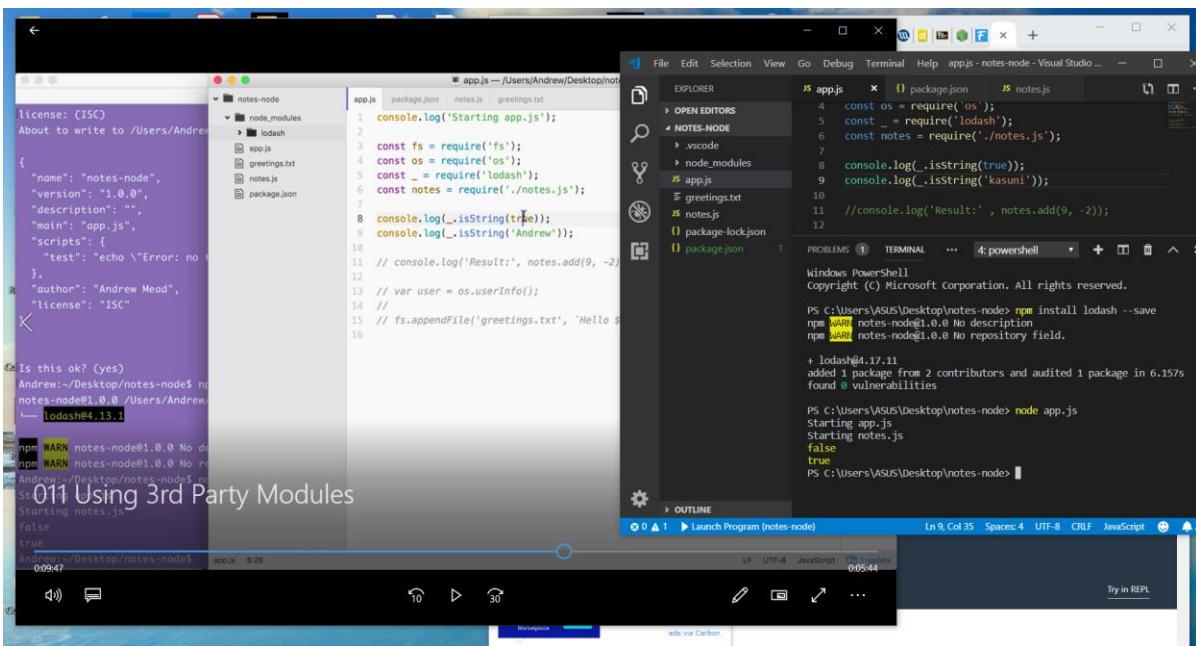


This screenshot shows a dual-monitor setup. The left monitor displays a browser window for the lodash documentation (<https://lodash.com/docs>) with the search term 'notes'. The right monitor shows Visual Studio Code with an open project named 'notes-node'. The Explorer sidebar shows files like package.json, notes.js, and greetings.txt. The app.js file contains code demonstrating lodash's \_.isString function. The terminal shows the command `npm install lodash` being run.

```
const os = require('os');
const _ = require('lodash');
const fs = require('fs');
const os = require('os');
const _ = require('lodash');
const notes = require('./notes.js');

console.log(_.isString(true));
console.log(_.isString('kasumi'));
notes.add('Hello', -2);
notes.add('World', -1);
notes.appendFile('greetings.txt', 'Hello World\n');
notes.appendFile('greetings.txt', 'World Hello\n');
```

```
+ lodash@4.17.11
added 1 package from 2 contributors and audited 1 package in 6.157s
found 0 vulnerabilities
```



This screenshot shows the same dual-monitor setup. The left monitor now displays a terminal window with the output of running `node app.js`. The right monitor shows Visual Studio Code with the same project structure and code as the previous screenshot. The terminal output shows the application running and writing to 'greetings.txt'.

```
Starting app.js
Starting notes.js
false
true
PS C:\Users\ASUS\Desktop\notes-node>
```

## Restarting App with nodemon

The screenshot shows a Visual Studio Code interface with two tabs open: 'app.js' and 'notes.js'. The terminal window at the bottom displays the following log:

```
Andrew:~/Desktop/notes-node$ nodemon app.js
[nodemon] 1.9.2
[nodemon] to restart at any time, enter `rs`
[nodemon] watching: ***!
[nodemon] starting 'node app.js'
Starting app.js
Starting notes.js
[ 'Andrew', 1, 2, 3, 4 ]
[nodemon] clean exit - waiting
[nodemon] restarting due to changes...
[nodemon] starting 'node app.js'
Starting app.js
Starting notes.js
[ 'kasumi', 1, 2, 3, 4 ]
[nodemon] clean exit - waiting for changes before restart
```

The screenshot shows a Visual Studio Code interface with two tabs open: 'app.js' and 'notes.js'. The terminal window at the bottom displays the following log:

```
Andrew:~/Desktop/notes-node$ nodemon app.js
[nodemon] 1.9.2
[nodemon] to restart at any time, enter `rs`
[nodemon] watching: ***!
[nodemon] starting 'node app.js'
Starting app.js
Starting notes.js
[ 'Andrew', 1, 2, 3, 4 ]
[nodemon] clean exit - waiting
[nodemon] restarting due to changes...
[nodemon] starting 'node app.js'
Starting app.js
Starting notes.js
[ 'Mike' ]
[nodemon] clean exit - waiting
00:56
```

## Getting input from user

The screenshot shows the Visual Studio Code interface. The left sidebar displays the project structure with files like `app.js`, `notes.js`, and `package.json`. The main editor window shows the `app.js` code. The terminal at the bottom shows the execution of the application and its responses to different commands.

```
File Edit Selection View Go Debug Terminal Help app.js - new-notes-node - Visual Studio Code

OPEN EDITORS JS app.js x JS notes.js
1 console.log('Starting app.js');
2
3 const fs = require('fs');
4 const _ = require('lodash');
5
6 const notes = require('./notes.js');
7
8 var command = process.argv[2];
9 console.log(`Command: ${command}`);
10
11 if (command === 'add'){
12     console.log('Adding new note');
13 } else if (command === 'list'){
14     console.log('Listing all notes');
15 }
16 else {
17     console.log("error");
18 }

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL 1: powershell + - x

at Function.Module.runMain (internal/modules/cjs/loader.js:754:1)
2) at startup (internal/bootstrap/node.js:283:19)
PS C:\Users\ASUS\Desktop\new-notes-node> node app.js list
Starting app.js
Starting notes.js
Command: list
Listing all notes
PS C:\Users\ASUS\Desktop\new-notes-node>
PS C:\Users\ASUS\Desktop\new-notes-node> node app.js list
Starting app.js
Starting notes.js
Command: list
Listing all notes
PS C:\Users\ASUS\Desktop\new-notes-node> node app.js add
Starting app.js
Starting notes.js

Ln 18, Col 2  Spaces: 4  UTF-8  CRLF  JavaScript  ⚡  🌐
```

This screenshot shows a dual-terminal setup in Visual Studio Code. The left terminal window shows the application's response to various commands. The right terminal window shows the actual command inputs being typed. Both windows show the same code as the previous screenshot.

```
File Edit Selection View Go Debug Terminal Help app.js - new-notes-node - Visual Studio Code
OPEN EDITORS JS app.js x JS notes.js
1 console.log('Starting app.js');
2
3 const fs = require('fs');
4 const _ = require('lodash');
5
6 const notes = require('./notes.js');
7
8 var command = process.argv[2];
9 console.log(`Command: ${command}`);
10
11 if (command === 'add'){
12     console.log('Adding new note');
13 } else if (command === 'list'){
14     console.log('Listing all notes');
15 }
16 else if (command === 'read'){
17     console.log('Reading note');
18 } else if (command === 'remove'){
19     console.log('Removing note');
20 }
21 else {
22     console.log('Command not recognized');
23 }

PROBLEMS TERMINAL ... 1: powershell + - x

Starting notes.js
Command: list
Listing all notes
Andrew:~/Desktop/notes-node$ node app.js
Starting app.js
Starting notes.js
Command: add
Adding new note
Andrew:~/Desktop/notes-node$ node app.js
Starting app.js
Starting notes.js
Command: list
Listing all notes
Andrew:~/Desktop/notes-node$ node app.js
Starting app.js
Starting notes.js
Command: read
Command not recognized
Andrew:~/Desktop/notes-node$ node app.js
Starting app.js
Starting notes.js
Command: remove
Removing note
Andrew:~/Desktop/notes-node$ node app.js
Starting app.js
Starting notes.js
Command: asdf
Command not recognized
Andrew:~/Desktop/notes-node$ node app.js
Starting app.js
Starting notes.js
Command: list
Listing all notes
Andrew:~/Desktop/notes-node$ node app.js
Starting app.js
Starting notes.js
Command: read
Reading note
Andrew:~/Desktop/notes-node$ node app.js
Starting app.js
Starting notes.js
Command: remove
Removing note
Andrew:~/Desktop/notes-node$ node app.js
Starting app.js
Starting notes.js
Command: rhd
Starting notes.js

Ln 24, Col 1  Spaces: 4  UTF-8  CRLF  JavaScript  ⚡  🌐
```

## Simplified input with yargs

The screenshot shows two terminal windows in Visual Studio Code. The top terminal window displays the output of running node app.js add multiple times with different arguments. The bottom terminal window shows the output of running node app.js read --title accounts, which triggers a TypeScript error because notes.getNote is not a function.

```

PS C:\Users\ASUS\Desktop\new-notes-node> node app.js add
Starting app.js
Starting notes.js
Command: add
process [ 'C:\Program Files\nodejs\node.exe',
  'C:\Users\ASUS\Desktop\new-notes-node\app.js',
  'add' ]
yargs { _: [ 'add' ], '$0': 'app.js' }
Adding new note
PS C:\Users\ASUS\Desktop\new-notes-node> node app.js add encrypted
Starting app.js
Starting notes.js
Command: add
process [ 'C:\Program Files\nodejs\node.exe',
  'C:\Users\ASUS\Desktop\new-notes-node\app.js',
  'add',
  'encrypted' ]
yargs { _: [ 'add', 'encrypted' ], '$0': 'app.js' }
Adding new note
PS C:\Users\ASUS\Desktop\new-notes-node> node app.js add --title=secrets
Starting app.js
Starting notes.js
Command: add
process [ 'C:\Program Files\nodejs\node.exe',
  'C:\Users\ASUS\Desktop\new-notes-node\app.js',
  'add',
  '--title=secrets' ]
yargs { _: [ 'add' ], title: 'secrets', '$0': 'app.js' }
Adding new note
PS C:\Users\ASUS\Desktop\new-notes-node>

PS C:\Users\ASUS\Desktop\new-notes-node>

```

```

notes-node -- bash - 108x29
~/Desktop/notes-node -- bash

Yargs { _: [ 'list' ], '$0': 'app.js' }
Getting all notes
Andrew:~/Desktop/notes-node$ node app.js read --title accounts
Starting app.js
Starting notes.js
Command: read
Yargs { _: [ 'read' ], title: 'accounts', '$0': 'app.js' }
/Users/Andrew/Desktop/notes-node/app.js:19
  notes.getNote(argv.title);
  ^
TypeError: notes.getNote is not a function
  at Object.<anonymous> (/Users/Andrew/Desktop/notes-node/app.js:19:32)
  at Module._compile (module.js:541:32)
  at Object.Module._extensions..js (module.js:550:10)
  at Module.load (module.js:458:32)
  at tryModuleLoad (module.js:417:12)
  at Function.Module._load (module.js:409:3)
  at Module.runMain (module.js:575:10)
  at run (node.js:348:7)
  at startup (node.js:140:9)
  at node.js:63:3
Andrew:~/Desktop/notes-node$ node app.js read --title accounts
Starting app.js
Starting notes.js
Command: read
Yargs { _: [ 'read' ], title: 'accounts', '$0': 'app.js' }
Getting note accounts
Andrew:~/Desktop/notes-node$ 018 Removing a Note-subtitle-en.vtt

```

## Working with JSON

The screenshot shows a Mac desktop environment. In the top-left corner, there are two terminal windows. The left one is titled "notes-node" and shows a Node.js script running, outputting logs about restarting and watching files. The right terminal window is titled "json.js" and shows a Node.js script reading and parsing a JSON object.

In the center, a Visual Studio Code instance is open. The Explorer sidebar shows a file tree with "OPEN EDITORS" containing "app.js", "json.js", and "notes.js". The "notes.js" editor tab is active, displaying the following code:

```
// var obj = {  
//   name: 'Andrew'  
// };  
// var stringObj = JSON.stringify(obj);  
// console.log(typeof stringObj);  
// console.log(stringObj);  
  
var personString = '{"name": "Andrew", "age": 25}';  
var person = JSON.parse(personString);  
console.log(typeof person);  
console.log(person);
```

The bottom-right terminal window in VS Code also shows the same logs as the first terminal window, indicating the script is running.

The screenshot shows a Windows desktop environment. A terminal window in the bottom-left corner displays logs from nodemon, indicating it is watching files and starting a node.js process.

A Visual Studio Code instance is open in the center. The Explorer sidebar shows a file tree with "OPEN EDITORS" containing "app.js", "json.js", "notes.js", and "notes-data.json". The "notes.js" editor tab is active, displaying the following code:

```
const fs = require('fs');  
  
var addNote = (title, body) => {  
  //console.log('Adding note', title, body);  
  var notes = [];  
  var note = {  
    title,  
    body  
  };  
  
  try{  
    var notesString = fs.readFileSync('notes-data.json');  
    notes = JSON.parse(notesString);  
  } catch (e) {}  
  
  var duplicateNotes = notes.filter((note) => note.title === title);  
  
  if (duplicateNotes.length === 0){  
    notes.push(note);  
    fs.writeFileSync('notes-data.json', JSON.stringify(notes));  
  }  
};  
  
var getAll = () => {  
  console.log('Getting all notes');  
};
```

The bottom-right terminal window in VS Code shows the logs from nodemon, confirming the script is running.

## Refactoring for reusability

```
File Edit Selection View Go Debug Terminal Help app.js - new-notes-node - Visual Studio Code

OPEN EDITORS
  app.js
  json.js
  notes.js
  notes-data.json

  NEW-NOTES-NODE
    .vscode
    node_modules
      playground
        notes.js
        notes-data.json
      json.js
      package-lock.json
      package.json

  app.js
  json.js
  notes.js
  notes-data.json
  package-lock.json
  package.json

EXPLORER
  OPEN EDITORS
    app.js
    json.js
    notes.js
    notes-data.json

  NEW-NOTES-NODE
    .vscode
    node_modules
      playground
        notes.js
        notes-data.json
      json.js
      package-lock.json
      package.json

  app.js
  json.js
  notes.js
  notes-data.json
  package-lock.json
  package.json

app.js
  const _ = require('lodash');
  const yargs = require('yargs');

  const notes = require('./notes.js');

  const argv = yargs.argv;
  var command = argv[0];
  //var command = process.argv[2];
  console.log(`Command: ${command}`);
  //console.log(`Process: ${process}`);
  console.log(`Yargs: ${argv}`);
  console.log(`Notes: ${notes}`);

  if (command === 'add') {
    //console.log(`Adding new note`);
    var note = notes.addNote(argv.title, argv.body);
    if (note) {
      console.log(`Note created`);
      console.log(`Title: ${note.title}`);
      console.log(`Body: ${note.body}`);
    } else {
  }

  Starting app.js
  Starting notes.js
  Command: add
  yargs: [ '_ add' ]
  title: 'to buy'
  body: 'someday'
  $0: 'app.js'
  PS C:\Users\VSUS\Desktop\new-notes-node> node app.js add --title="to buy" --body="f
  ood"
  Starting app.js
  Starting notes.js
  Command: add
  yargs: [ '_ add' ]
  title: 'to buy'
  body: 'food'
  $0: 'app.js'
  Note created
  Title: to buy
  Body: food
  PS C:\Users\VSUS\Desktop\new-notes-node>
```

## Removing a note

```
File Edit Selection View Go Debug Terminal Help notes.js - new-notes-node - Visual Studio Code

OPEN EDITORS
  app.js
  json.js
  notes.js
  notes-data.json

  NEW-NOTES-NODE
    .vscode
    node_modules
      playground
        notes.js
        notes-data.json
      json.js
      package-lock.json
      package.json

  app.js
  json.js
  notes.js
  notes-data.json
  package-lock.json
  package.json

EXPLORER
  OPEN EDITORS
    app.js
    json.js
    notes.js
    notes-data.json

  NEW-NOTES-NODE
    .vscode
    node_modules
      playground
        notes.js
        notes-data.json
      json.js
      package-lock.json
      package.json

  app.js
  json.js
  notes.js
  notes-data.json
  package-lock.json
  package.json

notes.js
  41;
  42;
  43;
  44;
  45;
  46 var getAll = () => (
  47   | console.log('Getting all notes');
  48 );
  49;
  50;
  51 var getNote = (title) => [
  52   | console.log(`Getting note: ${title}`);
  53 ];
  54;
  55 var removeNote = (title) => {
  56   //console.log(`Removing Note: ${title}`);
  57   var notes = fetchNotes();
  58   var filteredNotes = notes.filter((note) => note.title !== title);
  59   saveNotes(filteredNotes);
  60   return notes.length -= filteredNotes.length;
  61 };
  62;
  63 module.exports = {
  64   addNote,
  65   getAll,
  66   getNote,
  67   removeNote
  68 };

  at module.load (internal/modules/cjs/loader.js:600:32)
  at tryModuleLoad (internal/modules/cjs/loader.js:539:12)
  at Function.Module._load (internal/modules/cjs/loader.js:531:3)
  at Module.require (internal/modules/cjs/loader.js:637:17)
  at require (internal/modules/cjs/helpers.js:22:18)
  PS C:\Users\VSUS\Desktop\new-notes-node> node app.js remove --title=secret
  Starting app.js
  Command: remove
  yargs: [ '_ remove' ]
  title: 'secret'
  $0: 'app.js'
  PS C:\Users\VSUS\Desktop\new-notes-node>
```

## Reading notes and reusability

```
File Edit Selection View Go Debug Terminal Help app.js - new-notes-node - Visual Studio Code

OPEN EDITORS
  app.js
  json.js
  notes.js
  notes-data.json

  NEW-NOTES-NODE
    .vscode
    node_modules
      playground
        notes.js
        notes-data.json
      json.js
      package-lock.json
      package.json

  app.js
  json.js
  notes.js
  notes-data.json
  package-lock.json
  package.json

app.js
  23   console.log(`Title: ${note.title}`);
  24   console.log(`Body: ${note.body}`);
  25 } else {
  26   | console.log(`Note title taken`);
  27 }
  28 } else if (command === 'list'){
  29   //console.log(`Listing all notes`);
  30   notes.getAll();
  31 } else if (command === 'read'){
  32   //console.log(`Reading note`);
  33   var note = notes.getNote(argv.title);
  34   if (note) {
  35     console.log(`Note found`);
  36     notes.logNote(note);
  37   } else {
  38     console.log(`Note not found`);
  39   }
  40 } else if (command === 'remove'){
  41   //console.log(`Removing note`);
  42   var noteRemoved = notes.removeNote(argv.title);

  body: 'go to movies',
  $0: 'app.js'
  Note title taken
  PS C:\Users\VSUS\Desktop\new-notes-node>
  PS C:\Users\VSUS\Desktop\new-notes-node> node app.js add --title="things to do"
  dog" --body="go to movies"
  Starting app.js
  Starting notes.js
  Command: add
  yargs: [ '_ add' ]
  title: 'things to do'
  body: 'go to movies'
  $0: 'app.js'
  Note created
  --
  title: things to do
  Body: go to movies
  PS C:\Users\VSUS\Desktop\new-notes-node>
  PS C:\Users\VSUS\Desktop\new-notes-node>
```

## Listing notes

The screenshot shows the Visual Studio Code interface with the terminal tab active. The terminal output shows the execution of `node app.js list`. The application logs the creation of a note titled 'things to do2' with body 'go to movies'. It then lists all notes: 'secret' (body 'somebody'), 'secret3' (body 'somebody'), 'to buy' (body 'food'), 'things to do' (body 'undefined'), and 'things to do2' (body 'go to movies').

```
app.js - new-notes-node - Visual Studio Code
File Edit Selection View Go Debug Terminal Help
OPEN EDITORS JS app.js x JS json.js JS notes.js
EXPLORER .vscode playground notes.js
NEW-NOTES-NODE .vscode node_modules playground debugging.js json.js notes.json app.js
notes-data.json notes.js package-lock.json package.json
Note created
Title: things to do2
Body: go to movies
PS C:\Users\ASUS\Desktop\new-notes-node>
PS C:\Users\ASUS\Desktop\new-notes-node> node app.js list
Starting app.js
Starting notes.js
Command: list
args [ '_': [ 'list' ], '$0': 'app.js' ]
Printing 5 Note(s).
Title: secret
Body: somebody
Title: secret3
Body: somebody
Title: to buy
Body: food
Title: things to do
Body: undefined
Title: things to do2
Body: go to movies
PS C:\Users\ASUS\Desktop\new-notes-node>
```

## Requiring arguments and advanced yargs

The screenshot shows the Visual Studio Code interface with the terminal tab active. The terminal output shows the execution of `node app.js remove`. The application prompts for a note title and body. When 'title' is entered but 'body' is missing, it displays usage information for the 'remove' command.

```
app.js - new-notes-node - Visual Studio Code
File Edit Selection View Go Debug Terminal Help
OPEN EDITORS JS app.js x JS json.js JS notes.js
EXPLORER .vscode playground notes.js
NEW-NOTES-NODE .vscode node_modules playground debugging.js json.js notes.json app.js
notes-data.json notes.js package-lock.json package.json
describe: 'title or note',
demand: true,
alias: 't'
const bodyOptions = {
  describe: 'Body of note',
  demand: true,
  alias: 'b'
}
//const argv = yargs.argv;
const argv = yargs
  .command('add', 'Add a new note', {
    title: titleOptions,
    body: bodyOptions
  })
  .command('list', 'List all notes')
  .command('read', 'Read a note', {
    title: titleOptions,
  })
  .command('remove', 'Remove a note', {
    title: titleOptions
  })
  .help()
  .argv;
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
Starting app.js
Starting notes.js
app.js remove
Remove a note
Options:
  --version Show version number [boolean] --help Show help [boolean] --title Title of note
Missing required argument: title
PS C:\Users\ASUS\Desktop\new-notes-node>
```

## Arrow functions

The screenshot shows the Visual Studio Code interface with the following details:

- File Structure (EXPLORER):** Shows files like `app.js`, `arrow-function.js`, and `notes.js`.
- Open Editors:** `arrow-function.js` is the active editor.
- Code Content:**

```

4    }; */
5
6    var square = x => x * x;
7    console.log(square(9));
8
9    var user = {
10      name: 'kasuni',
11      sayHi: () => {
12        console.log(arguments);
13        console.log(`Hi. I'm ${this.name}`);
14      },
15      sayHiAlt () {
16        console.log(arguments);
17        console.log(`Hi. I'm ${this.name}`);
18      }
19    };
20
21 user.sayHi(1,2,3);

```
- Terminal:** Shows the output of running the code in Node.js, including the path to the file and the resulting log message: "Hi. I'm undefined".
- Status Bar:** Shows the current file is `arrow-function.js`, line 14, column 7, with 140% zoom and 2 items in the status bar.

## 4) Async Basics, Call stack & event loop, Callback functions & APIs and Pretty printing objects

The screenshot shows the Visual Studio Code interface with the following details:

- File Structure (EXPLORER):** Shows files like `async-basics.js`, `callbacks.js`, and `app.js`.
- Open Editors:** `app.js` is the active editor.
- Code Content:**

```

3 request({
4   url: 'https://maps.googleapis.com/maps/api/geocode/json?address=1
5   json: true
6 }, (error, response, body) => {
7   console.log(JSON.stringify(body, undefined, 2));
8 });

```
- Terminal:** Shows the command `node app.js` being run, and an error message indicating it cannot find the module `c:\Users\ASUS\Desktop\weather-app\playground\app.js`.
- Status Bar:** Shows the current file is `app.js`, line 8, column 4, with 140% zoom and 2 items in the status bar.

## What's makes up an HTTP request & Encoding user input

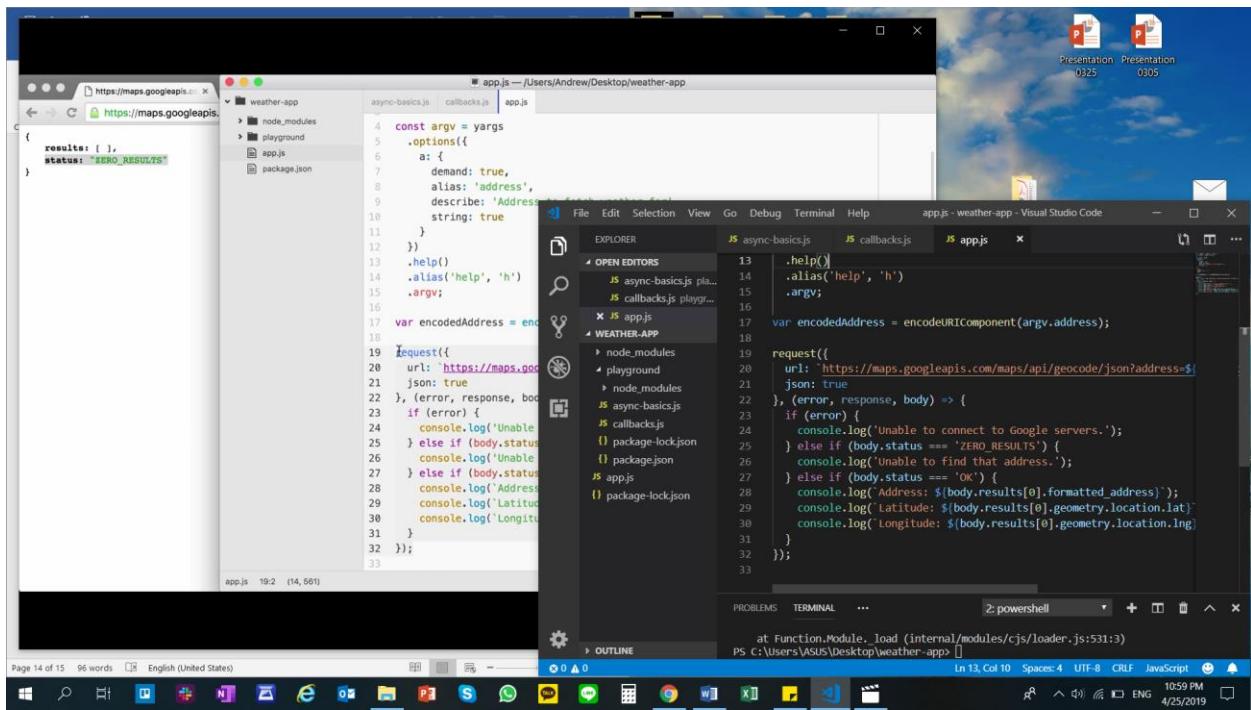
The screenshot shows the Visual Studio Code interface with the following details:

- File Explorer:** Shows the project structure for "weather-app" with files like app.js, callbacks.js, and package.json.
- Code Editor:** Displays the contents of app.js, which uses the "request" module to make an HTTP GET request to the Google Geocoding API to encode a user-specified address.
- Terminal:** Shows command-line output indicating npm warnings about missing dependencies.
- Taskbar:** Shows the Windows taskbar with various pinned icons.
- Status Bar:** Shows the current file is "app.js - weather-app - Visual Studio Code", the file path is "C:\Users\ASUS\Desktop\weather-app", and the status "14 1055 PM 4/25/2019".

The screenshot shows the Visual Studio Code interface with the following details:

- File Explorer:** Shows the project structure for "weather-app" with files like app.js, callbacks.js, and package.json.
- Code Editor:** Displays the contents of app.js, which now includes encoding of the user-specified address before sending it to the Google Geocoding API.
- Terminal:** Shows the application output displaying the encoded address and coordinates for the specified location.
- Taskbar:** Shows the Windows taskbar with various pinned icons.
- Status Bar:** Shows the current file is "app.js - weather-app - Visual Studio Code", the file path is "C:\Users\ASUS\Desktop\weather-app", and the status "24 1057 PM 4/25/2019".

## Callback errors



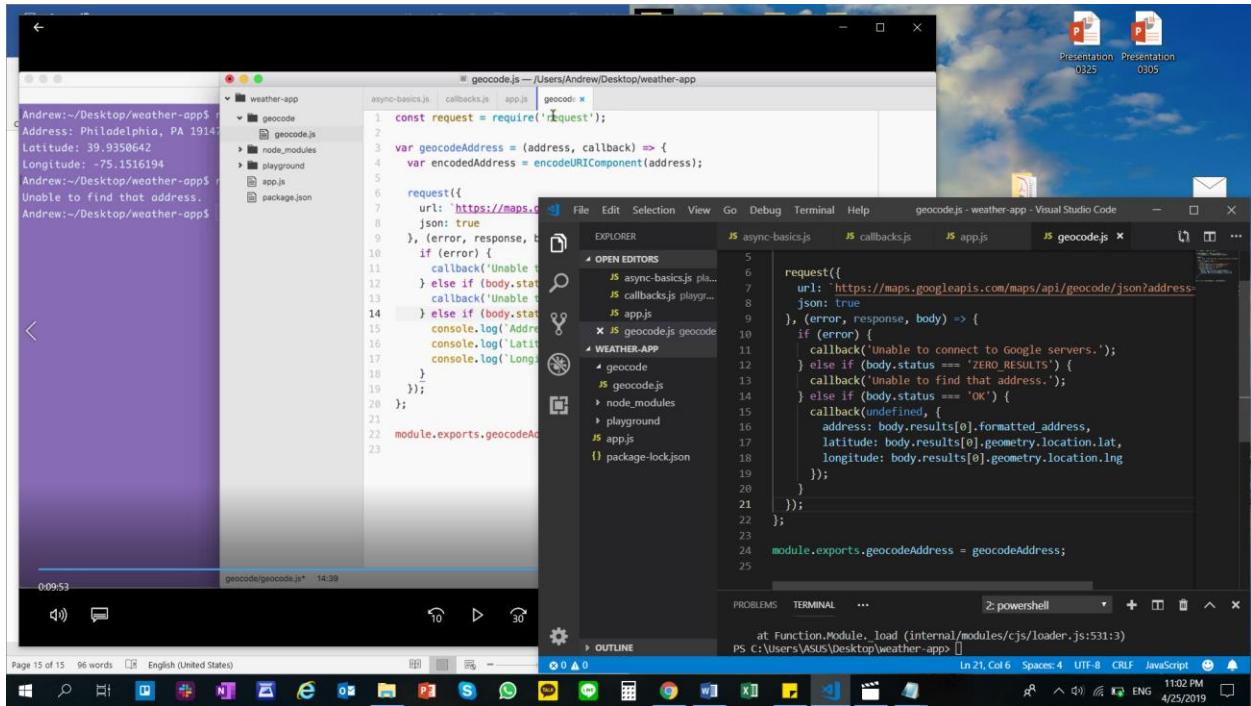
The screenshot shows a Windows desktop environment with Visual Studio Code open. A browser window in the background displays the URL <https://maps.googleapis.com/maps/api/geocode/json?address=Philadelphia+PA>. The response shows an error message: "results: [], status: 'ZERO\_RESULTS'". In the foreground, the Visual Studio Code editor has three tabs: "async-basics.js", "callbacks.js", and "app.js". The "app.js" tab contains the following code:

```
const argv = yargs
    .options({
        a: {
            demand: true,
            alias: 'address',
            describe: 'Address to search for',
            string: true
        }
    })
    .help()
    .alias('help', 'h')
    .argv;

var encodedAddress = encodeURI(argv.address);

request({
    url: 'https://maps.googleapis.com/maps/api/geocode/json?address=' + encodedAddress,
    json: true
}, (error, response, body) => {
    if (error) {
        console.log('Unable to connect to Google servers.');
    } else if (body.status === 'ZERO_RESULTS') {
        console.log('Unable to find that address.');
    } else if (body.status === 'OK') {
        console.log(`Address: ${body.results[0].formatted_address}`);
        console.log(`Latitude: ${body.results[0].geometry.location.lat}`);
        console.log(`Longitude: ${body.results[0].geometry.location.lng}`);
    }
});
```

## Abstracting callbacks



The screenshot shows a Windows desktop environment with Visual Studio Code open. A terminal window in the background shows the command `node geocode.js Philadelphia PA` and its output: "Address: Philadelphia, PA 19147", "Latitude: 39.9350642", and "Longitude: -75.1516194". In the foreground, the Visual Studio Code editor has four tabs: "async-basics.js", "callbacks.js", "app.js", and "geocode.js". The "geocode.js" tab contains the following code:

```
const request = require('request');

var geocodeAddress = (address, callback) => {
    var encodedAddress = encodeURIComponent(address);

    request({
        url: 'https://maps.googleapis.com/maps/api/geocode/json?address=' + encodedAddress,
        json: true
    }, (error, response, body) => {
        if (error) {
            callback('Unable to connect to Google servers.');
        } else if (body.status === 'ZERO_RESULTS') {
            callback('Unable to find that address.');
        } else if (body.status === 'OK') {
            callback(undefined, {
                address: body.results[0].formatted_address,
                latitude: body.results[0].geometry.location.lat,
                longitude: body.results[0].geometry.location.lng
            });
        }
    });
};

module.exports.geocodeAddress = geocodeAddress;
```

## Wiring up weather search

```

Andrew:~/Desktop/weather-app$ node app.js
{
  "address": "Philadelphia, PA",
  "latitude": 39.9396284,
  "longitude": -75.18663959999999
}
Andrew:~/Desktop/weather-app$ node app.js
85:27
Andrew:~/Desktop/weather-app$ node app.js
85:28
Andrew:~/Desktop/weather-app$ node app.js
Unable to fetch weather.
Andrew:~/Desktop/weather-app$ node app.js
Unable to connect to Forecast.io server.
Andrew:~/Desktop/weather-app$ node app.js

```

The code in `app.js` attempts to fetch weather data from Forecast.io. It uses the `request` module to make a GET request to `'https://api.forecast.io/forecast/4a04d1c42fd9d3c97a2c291a32'`. If successful, it logs the current temperature. If there's an error or the status code is 400, it logs an error message.

The code in `geocode.js` defines a function `geocodeAddress` that takes an address and an optional JSON object. It uses `request` to make a GET request to `'https://maps.googleapis.com/maps/api/geocode/json?address=Philadelphia%2C+PA'`. If successful, it logs the results. If there's an error or the status code is 400, it logs an error message.

## Chaining callbacks together

```

Andrew:~/Desktop/weather-app$ node app.js
80:52
Andrew:~/Desktop/weather-app$ node app.js
{
  "temperature": 81.61,
  "apparentTemperature": 87.02
}
Andrew:~/Desktop/weather-app$ node app.js
Flemington, NJ 08822, USA
It's currently 81.56. It feels like 87.02.
Andrew:~/Desktop/weather-app$ node app.js
Key West, FL 33040, USA
It's currently 84.17. It feels like 84.17.
Andrew:~/Desktop/weather-app$ node app.js

```

The code in `weather.js` uses the `weather` module to get the current weather for a given address. It then uses the `geocodeAddress` function from `geocode.js` to geocode the address. The output shows the current temperature and apparent temperature for two different locations.

## Intro to ES6 promises

The screenshot shows a Windows desktop environment with Visual Studio Code open. There are two terminal windows visible:

- Terminal 1 (Left):** Shows node.js logs from a 'weather-app' project. It includes several 'nodemon' log entries indicating restarts and successes, along with some error messages about promises.
- Terminal 2 (Right):** Shows a Node.js application interacting with Google's Geocoding API to find coordinates for an address.

Two code editors are open in the main workspace:

- promise.js:** A file containing a Promise example. It defines a promise that resolves after 2500ms with the message 'Hey, It worked!'.
- geocodejs:** A file containing a geocoding function. It uses the promise defined in promise.js to make an API request and handle the response.

The Visual Studio Code interface includes standard tools like Explorer, Outline, and Problems.

## 5) Web Servers and application Deployment

The screenshot shows a Windows desktop environment with Visual Studio Code open. A terminal window is active, showing the following git workflow:

- Initial commit: `git add .` and `git commit -m "Initial commit"
- Pushing to origin: `git push`
- Checking status: `git status`
- Adding a new file: `git add views/projects.hbs`
- Committing the change: `git commit -m "Add project page"
- Pushing again: `git push`

A code editor window shows the `server.js` file, which contains Express.js routes for home, about, and projects pages. The file uses EJS templates for rendering.

## 6) Testing your application

The screenshot shows a Windows desktop environment with Visual Studio Code open. The terminal window on the left displays the output of running tests:

```
node-tests └── node └── npm MANIFEST
> node-tests@1.0.0 test /Users/Andrew/mymyver...
> mocha **/*.test.js

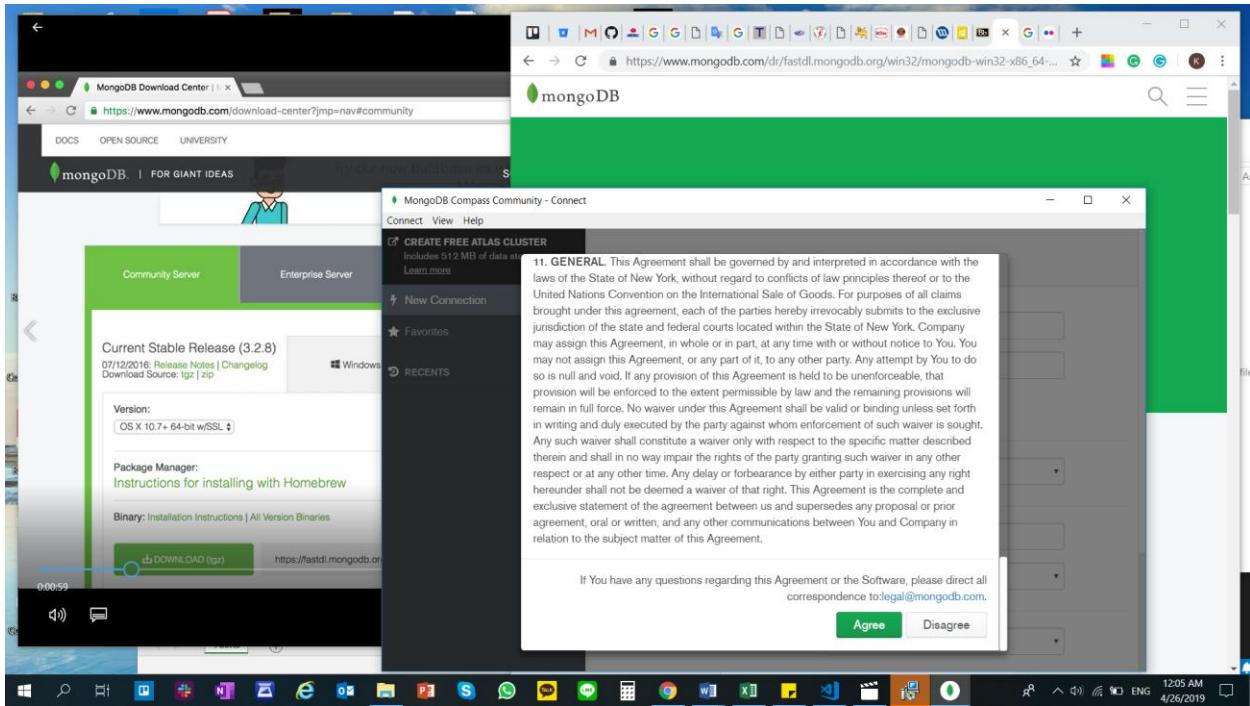
  ✓ should set firstName and lastName
  ✓ should return hello world
  ✓ should return my user object
  ✓ should call the spy correctly
  ✓ should call saveUser with user object
  ✓ should sync add two numbers
  ✓ should square a number
  ✓ should sync square a number
#add
  ✓ should add two numbers

  9 passing (2s)

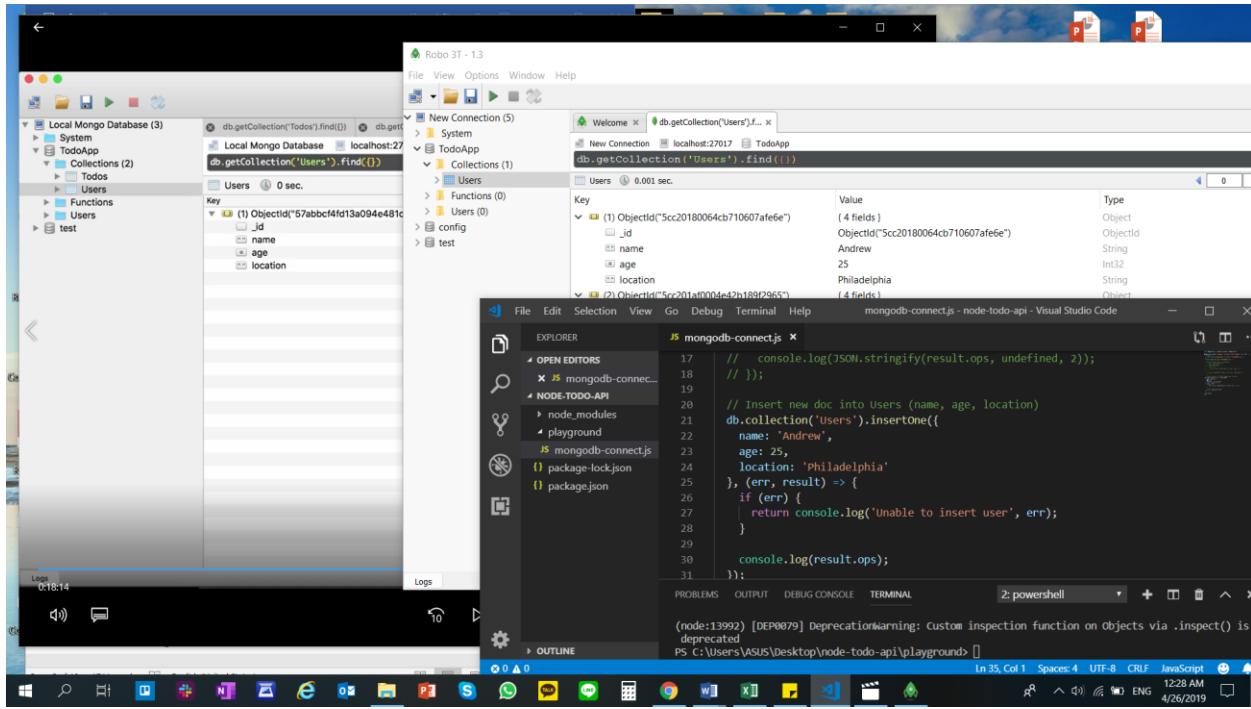
[nodemon] clean exit - waiting for changes...
```

The code editor on the right contains the file `app.test.js`, which includes Jest test cases for the application's logic. The file `db.js` is also visible in the Explorer sidebar.

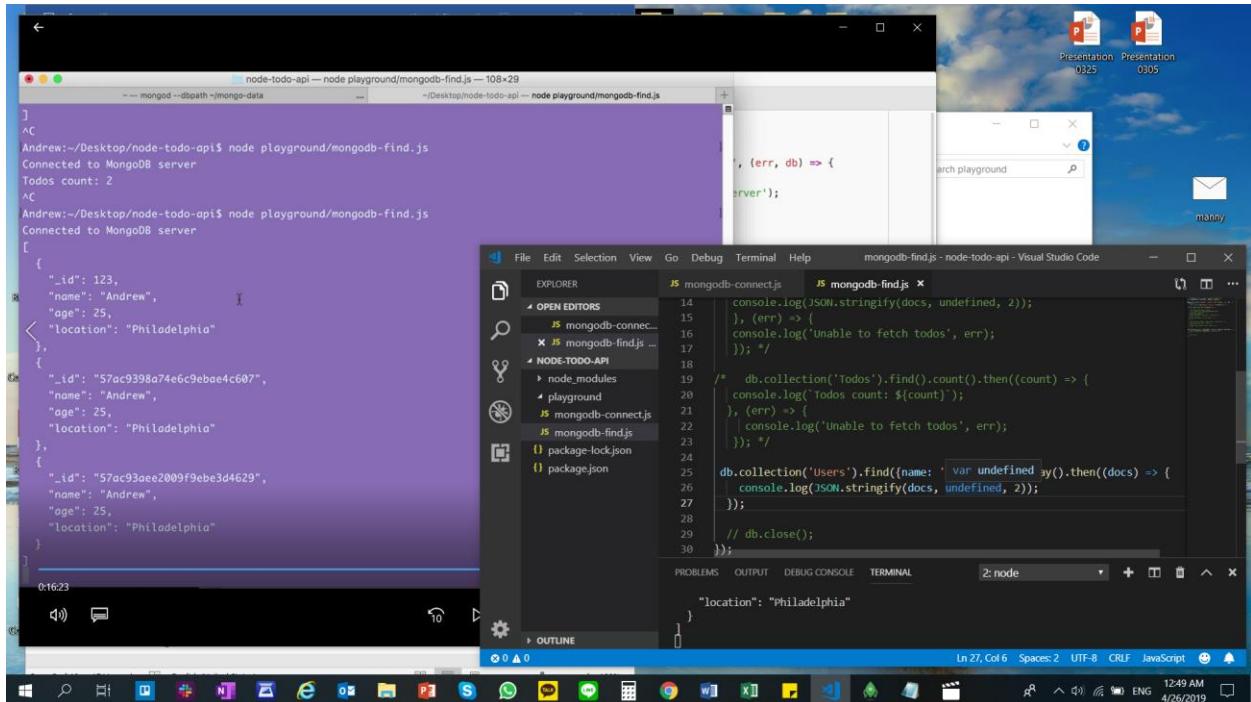
## 7) MongoDB and REST APIs



## Connecting to Mongo and Writing Data



## The ObjectId & Fetching data



## Deleting Document & Updating Document

The screenshot shows two code editors in Visual Studio Code. The left editor contains `mongodb-update.js` which connects to a MongoDB server and finds one document in the 'Todos' collection. The right editor contains `mongodb-delete.js` which finds a document by ID and updates its name to 'Andrew'. Both scripts use the Mongoose library.

```
// const MongoClient = require('mongodb').MongoClient;
const {MongoClient, ObjectID} = require('mongodb');

MongoClient.connect('mongodb://localhost:27017/TodoApp', (err, db) => {
  if (err) {
    return console.log('Unable to connect to MongoDB server');
  }
  console.log('Connected to MongoDB server');

  db.collection('Todos').findOneAndUpdate(
    { _id: new ObjectID('57abbcf4fd13a094e481cf2c') },
    {
      $set: {
        name: 'Andrew'
      },
      $inc: {
        age: 1
      }
    },
    {
      returnOriginal: false
    }).then(result => {
    console.log(result);
  });
  // db.close();
});

db.close();
```

```
db.collection("Users").findOneAndUpdate({
  _id: new ObjectID('57abbcf4fd13a094e481cf2c')
}, {
  $set: {
    name: 'Andrew'
  },
  $inc: {
    age: 1
  }
}, {
  returnOriginal: false
}).then(result => {
  console.log(result);
});
```

## Creating a Test Database

The screenshot shows a code editor in Visual Studio Code with `server.test.js` open. This file uses the Supertest library to test a Node.js application. It defines a 'todos' collection and performs various operations like creating, updating, and deleting documents. The application uses the Express framework and Mongoose for MongoDB integration.

```
const _ = require('lodash');
const express = require('express');
const bodyParser = require('body-parser');
const {ObjectID} = require('mongodb');

var mongoose = require('mongoose');
var {Todo} = require('../models/todo');
const bodyparser = require('body-parser');
const expect = require('expect');
const request = require('supertest');
const app = express();
const port = process.env.PORT || 3001;

app.use(bodyParser.json());

app.post('/todos', (req, res) => {
  var todo = new Todo({
    text: req.body.text
  });

  todo.save().then((doc) => {
    res.send(doc);
  }, (e) => {
    res.status(400).send(e);
  });
});

app.get('/todos', (req, res) => {
  Todo.find().then((todos) => {
    res.send(todos);
  });
});

beforeEach((done) => {
  Promise.all([
    Todo.deleteMany({}),
    User.deleteMany({})
  ]).then(() => done());
});
```