DotNet Society
Hour-of-Code

DOT NET SOCIETY

Build-a-Bot Workshop 2: JavaScript & Node.js

Hello!

I am Martius

I am here because I love to share about cool new technologies ☻

You can find me at @martiuslim on Telegram or GitHub

Join our Microsoft Student Community Telegram chat @ **http://tinyurl.com/smudotnet**

These slides can be found @ https://github.com/martiuslim/build-a-bot

Hour-of-Code: Build-a-Bot

- ◇ Understanding Chatbots
- ☞ ◇ **> JavaScript & Node.js <**
- ◇ Microsoft Bot Framework
- ◇ Git
- ◇ Azure

JavaScript & Node.js

- ◇Software Prerequisites
- ◇JavaScript
- ◇Node.js
- ◇Building a server using Express.js

2

Software Prerequisites

Getting started

Visual Studio Code

- https://code.visualstudio.com/
- Visit the extensions tab (**ctrl + shift + x**) and download the **Code Runner** extension by Jun Han
- This allows you to run code in your VSCode terminal using the command **ctrl + alt + n**

Homebrew
(Mac users only)

- https://brew.sh/

Node.js

- https://nodejs.org/en/
- Downloading the LTS version is enough
- For Mac users that installed Homebrew, you can install node using ‘**brew install node**’
- Check your installation using ‘**node –v**’

npm (Node Package Manager)

- https://docs.npmjs.com/getting-started/installing-node
- It should come installed with Node.js
- Check your installation using **‘npm –v’**
- Update your npm using **‘npm install npm@latest –g’**

2

JavaScript

A technical introduction

JavaScript (ECMAScript 6)

- Lightweight interpreted programming language
- Multi-paradigm scripting language
- Usually runs on the client side of the web

- https://developer.mozilla.org/en-US/docs/Web/JavaScript

Types

- **String** – ‘hello world’, ‘building a bot!’
- **Number** – 7, 12.34
- **Booleans** – true, false
- **Null** – represents ‘no value’
- **Undefined** – represents an undeclared value

console.log()

- Prints data to the console (your cmd/ powershell/ terminal window)
- Quick and useful tool for debugging
- Use it often when developing!

Operators

- Built-in operators – + - / *
- Compute numerical operations on numbers
- There are also += -= /= and *=
- This means to perform the mathematical operation on the left using the number on the right, then assign the new value to the variable.

Properties & Methods

- JavaScript associates certain properties with different data types
- Built-in methods for different data types such as **.length**

Libraries

- Collection of functions or methods
- Useful abstractions of code that others have written so that you don't have to write your own
- If there's a function you're looking for, chances are someone out there has already written it

Comments

- Write single-line comments with // and multi-line comments with /* */
- Use comments to document and explain your code

Variables

- ◇ Holds reusable data
- ◇ JavaScript traditionally used **var**
- ◇ ES6 introduced **const** and **let**
- ◇ Use **const** when you don’t need to reassign the value and **let** when you need to.

String Interpolation

- Insert variables into your Strings
- ES6: Use **backticks** and **${variableName}**
- Backtick is the key above your Tab key and beside the number 1

Control Flow

- Statements that allow JavaScript programs to make **decisions** by executing code based on **conditions**

If/ else and else if

```
if (condition) {
  // do something if condition is true
} else if (anotherCondition) {
  // do something else if anotherCondition is true
} else {
  // if all else fails do this
}
```

Truthy vs Falsy

- All conditions are evaluated to be **truthy** or **falsy**
- JavaScript can evaluate conditions on more than just 'true' or 'false'
- E.g. a non-empty String evaluates to be **truthy**, undefined or null evaluates to be **falsy**

Switch statements

- Another way of writing if/ else if/ else
- switch(condition) {

```
  case (condition):
    // do something
  case (anotherCondition):
    // do something else
}
```

Ternary Operator ?

- Allows us to refactor simple if/ else statements
- (condition) ? (result if condition is true) : (result if condition is false)
- let hungry = (hungerLevel > 5) ? ‘Yes’ : ‘No’

Comparison Operators

- Less than > or more than <
- Correspondingly >= and <= for less than equals and more than equals respectively
- Comparing the two values evaluates to true or false

Logical Operators

- === for strict equality
- Use **!** to reverse the truthiness or falsiness
- We may want to evaluate several conditions
- To say ‘both must be true’ we use **&&**
- To say ‘either can be true’ we use **||**

Assignment Operator

- = is used for assignment
- let myValue = 10
- Don’t confuse it with the equality operator

  === or !==

Functions

- Written to perform a task
- Take in data, perform a set of tasks on the data, then return a result

Functions

- Define **parameters** to be used when calling the function
- Pass in **arguments** when calling the function to set its parameters
- Use **return** to return the result of a function

Scope

- The idea in programming that some variables are accessible or inaccessible from other parts of the program
- **Global Scope** refers to variables that are accessible to every part of the program
- **Block Scope** refers to variables that are only accessible within the block they are defined

Arrays

- Arrays are lists and a way to store data in JavaScript
- Created with brackets **[ ]**
- Index starts from 0, not 1
- Access items using the index such as myArray[0]
- Has .length property to let you see the size of the array

Arrays

- Has its own methods such as .push() and .pop() which adds and removes items from the array respectively
- You can create an array of arrays!
- let countries = [[‘SG’, ‘MY’], [‘US, CA’]];

Loops

- Write less repetitive code using loops
- **for** loops allow you to repeat a block of code for a known amount of times
- **while** loops are for repeating a block of code for an unknown amount of times
- **Infinite loops** occur when stop conditions are never met

Iterators

- **.forEach()** – used to execute code on every element in an array, but does not change the array and **returns undefined**

Iterators

- **.map()** – executes the same code on every element in an array, but does not change the array and **returns a new array with the updated elements**

Iterators

- **.filter()** – checks every element in an array to see if it meets certain criteria and **returns a new array with the elements that return truthy for the criteria**

Objects

- **Objects** store **key-value pairs** and let us represent real-world things in JavaScript
- **Properties** in objects are separated by commas, **key-value pairs** are always separated by colons

Objects

- You can add or edit a property within an object with **dot notation**
- The **this** keyword helps us scope inside of object methods. **this** is a dynamic variable that can change depending on the object calling the method

3

Node.js

A JavaScript runtime for server-side implementations

What is Node.js

- Non-blocking, asynchronous event driven JavaScript runtime
- Designed to build scalable network applications

Blocking vs Non-blocking

```javascript
const fs = require('fs');
const data = fs.readFileSync('/file.md');
// blocks here until file is read
```

```javascript
const fs = require('fs');
fs.readFile('/file.md', (err, data) => {
    if (err) throw err;
});
```

Callbacks

- Asynchronous equivalent for a function
- A callback function is called at the completion of a given task

```
const fs = require('fs');
fs.readFile('/file.md', (err, data) => {
    if (err) throw err;
});
```

Modules

- You can consider them to be the same as JavaScript libraries
- To include a module, use the **require()** function

npm

- Node Package Manager
- npm Registry is a public collection of open-source packages or libraries
- Don’t reinvent the wheel! Use **npm install** to make use of code others have written!

4

Hands-on!

Let’s build our own server using Express.js!

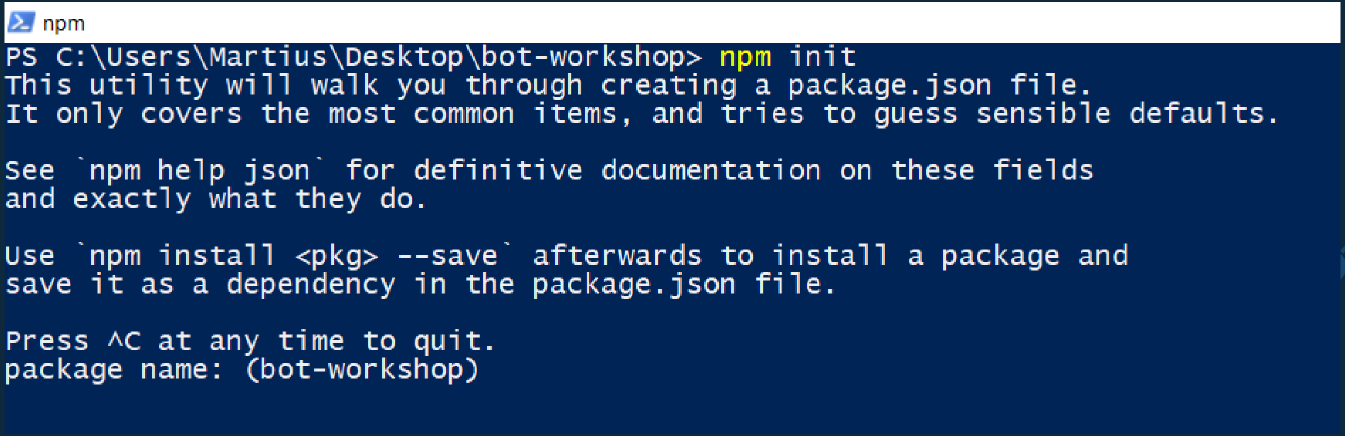
Express.js

- ◇ Web application framework for Node
- ◇ Minimal and flexible
- ◇ Great for building Web APIs (Application Programming Interface)

Initializing your Node.js project

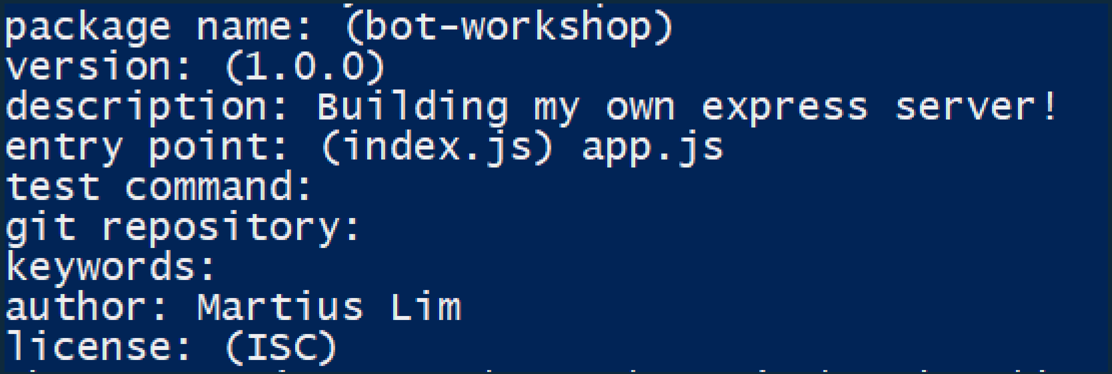
- Create a new folder on your Desktop
- Navigate to the folder and open a cmd or terminal window there.
- Type **npm init** into the cmd or terminal window

Initializing your Node.js project



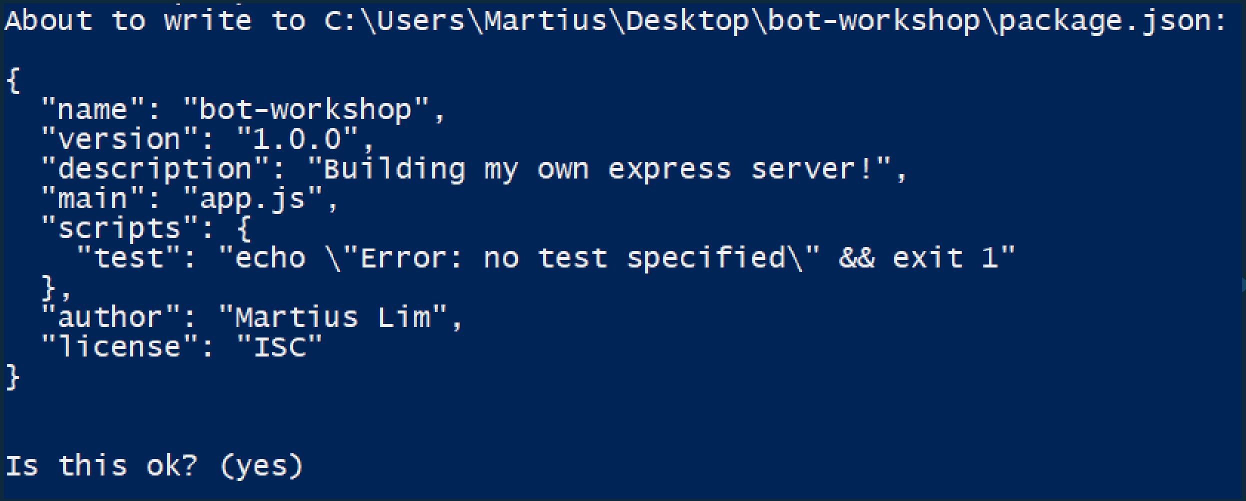
```
PS C:\Users\Martius\Desktop\bot-workshop> npm init
This utility will walk you through creating a package.json file.
It only covers the most common items, and tries to guess sensible defaults.

See `npm help json` for definitive documentation on these fields
and exactly what they do.

Use `npm install <pkg> --save` afterwards to install a package and
save it as a dependency in the package.json file.

Press ^C at any time to quit.
package name: (bot-workshop)
```

Initializing your Node.js project



```
package name: (bot-workshop)
version: (1.0.0)
description: Building my own express server!
entry point: (index.js) app.js
test command:
git repository:
keywords:
author: Martius Lim
license: (ISC)
```

Initializing your Node.js project



```
About to write to C:\Users\Martius\Desktop\bot-workshop\package.json:

{
  "name": "bot-workshop",
  "version": "1.0.0",
  "description": "Building my own express server!",
  "main": "app.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "author": "Martius Lim",
  "license": "ISC"
}


Is this ok? (yes)
```

package.json

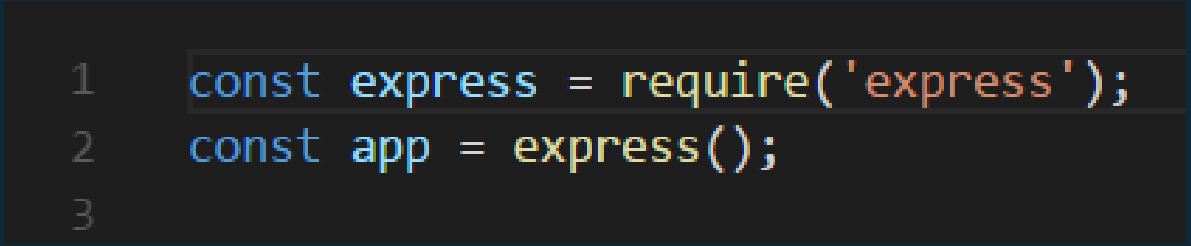
- Tells you important information about your project
- Tells npm what dependencies your project requires
- Allows you to specify **scripts** that you can run

Installing Express

- npm install express --save
- Note that there are **two dashes**
- --save tells npm to save this module into your package.json file as well

Building a server using Express

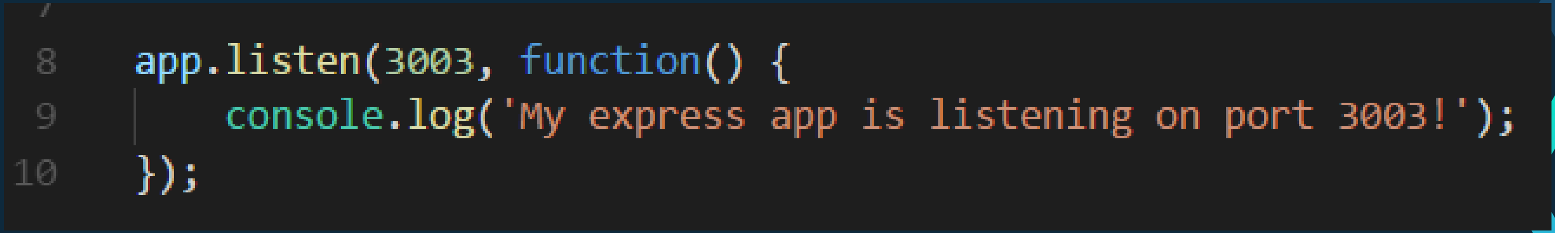
- Open your project in Visual Studio Code
- Create a file named **app.js**



```
const express = require('express');
const app = express();

```

Building a server using Express

- Write the code that starts your server listening on a port, in this case port 3003



```
app.listen(3003, function() {
    console.log('My express app is listening on port 3003!');
});
```

Building a server using Express

- ◇ Congrats! You've successfully built a server using Express on Node.js
- ◇ Now let's make it better ☺

HTTP Request Methods

- There are several HTTP Request Methods, namely POST, GET, PUT, DELETE for the Create, Read, Update, Delete functions
- We will only focus on GET and POST today

HTTP Request Methods

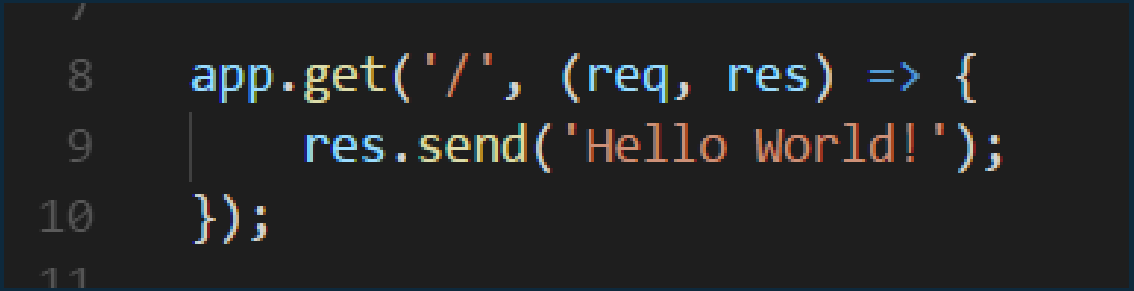
- GET requests a representation of the specified resource
- Requests using GET should only retrieve data

HTTP Request Methods

- POST is used to submit an entity to the specified resource
- Often causes a change in state or side effects on the server

HTTP Requests in Express

◇ Writing a GET request route handler



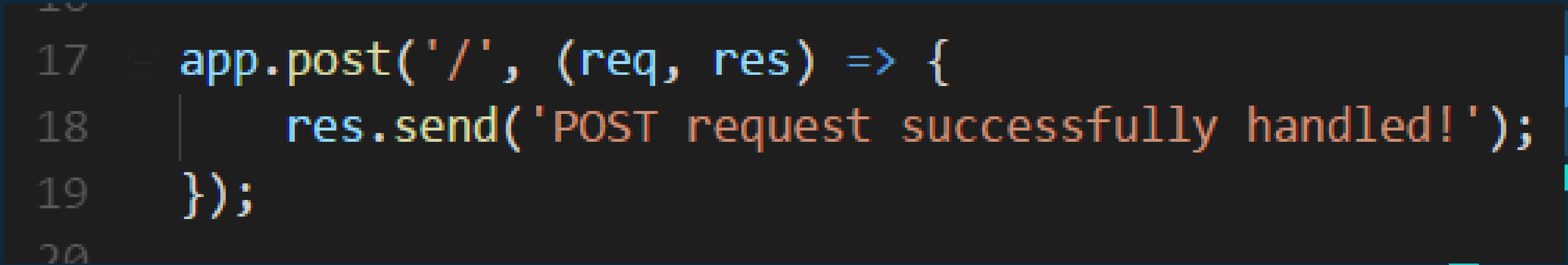
```
app.get('/', (req, res) => {
    res.send('Hello World!');
});
```

HTTP Requests in Express (GET)

- Pass in parameters using ?
- localhost:3003?hello=world
- Accessing GET parameters
- Use the query property in the request variable
- E.g. req.query

HTTP Requests in Express

- Writing a POST request route handler



```
app.post('/', (req, res) => {
    res.send('POST request successfully handled!');
});
```

HTTP Requests in Express (POST)

- Pass in parameters in the body
- localhost:3003
- Accessing POST parameters
- Use the body property in the request variable
- E.g. req.body

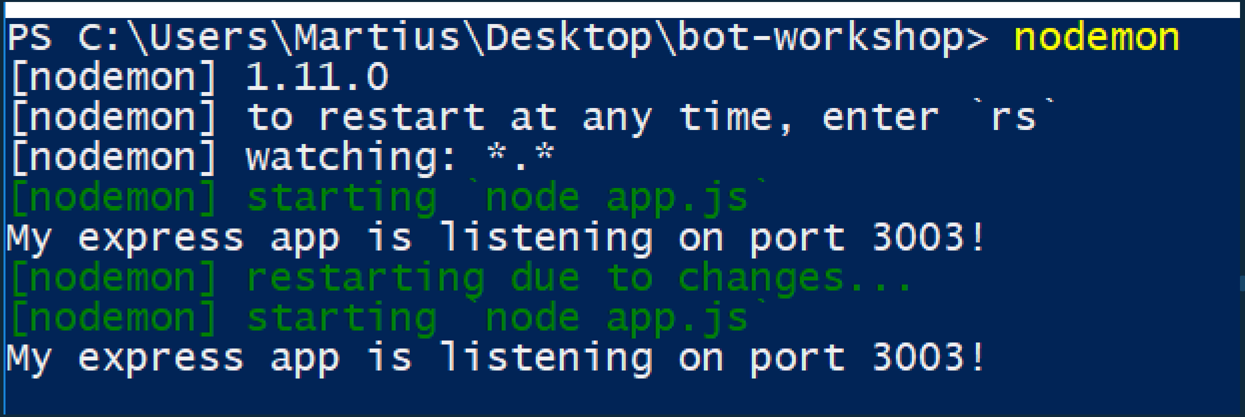
nodemon

- You may find it annoying to keep typing in node app.js whenever you make new changes
- nodemon is a tool that helps you reload your code automatically upon saving

nodemon

- You can install it using **npm install nodemon –g**
- Now you can run your code using **nodemon app.js** instead of **node app.js**

nodemon



```
PS C:\Users\Martius\Desktop\bot-workshop> nodemon
[nodemon] 1.11.0
[nodemon] to restart at any time, enter `rs`
[nodemon] watching: *.*
[nodemon] starting `node app.js`
My express app is listening on port 3003!
[nodemon] restarting due to changes...
[nodemon] starting `node app.js`
My express app is listening on port 3003!
```

Thanks!

Any questions?

You can find me at:

- ◇ @martiuslim (Telegram)
- ◇ martius.lim.2015@sis.smu.edu.sg

- ◇ Next week: Microsoft Bot Framework