Node Workshop

Node.js is a javascript runtime. Project will work to write a few RESTful endpoints running a server saving, retrieving, updating and deleting data.

Project Creation

- npm is the Node Package Manager (similar to nuget for .net or Linux apt-get)
- hapiJS is a node package (library) used to create a web server, including RESTful endpoints

Step Description	Command
Open command line	Open Git Bash On windows or terminal on Linux/Mac
Change to root directory	➢ **cd /c** on windows **cd /** on Linux/Mac
Make a new directory for project	➢ mkdir nodeproj
Change directory to new project directory	➢ cd nodeproj
Initialize a git repository	➢ git init
Initial npm	➢ npm init
answer npm questions	➢ defaults are fine
Add third-party libraries hapi.js that will be used in this project www.hapijs.com	➢ npm install --save hapi

Project Setup

Step Description	Command
Create a new file **.gitignore** (filename begins with a dot) Add files / directories git will ignore node_modules - packages installed by npm *.map - files that are for debugging *.bak - some editors keep original files with bak extension ~* - some temp files begin with ~	➢ add the following: node_modules *.map *.bak dist ~*

HelloWord

Simple test to make sure node is installed properly

Step Description	Command
Create a directory called src	➢ mkdir src
Create a new file called main.js in src directory	➢ add the following: console.log('Hello World!');
Setup npm to execute main.js	➢ edit package.json ➢ under scripts (line 6/7)add the following ➢ "start": "node src/main.js", ➢ The file should be: "main": "index.js", "scripts": { "start": "node src/main.js", "test": "echo \"Error: no test specified\" && exit 1" },
Execute the application	➢ npm start returns: Hello World!

Git Commit

Let's check in code to git

Step Description	Command
Add files to git from command line in the root project directory	➢ git add .gitignore ➢ git add package.json ➢ git add src
Verify what is ready to be committed	➢ git status
Commit	➢ git commit -m "initial project commit"

Hapi HelloWorld

Modify **main.js** to use hapi and return hello world with the following code:

```
'use strict';
var Hapi = require('hapi');

var server = new Hapi.Server();

server.connection({port: 3000});

server.route({
    method: 'GET',
    path: '/',
    handler: function (request, reply) {
        reply('Hello World! from Hapi');
    }
});

server.start(function (err) {
    if (err) {
        throw err;
    }
    console.log('Server running at ', server.info.port);
});
```

From command line start application **npm start**

Start Chrome or another browser

Type **http://localhost:3000**

Server will response with Hello, World! from Hapi

To stop the server use **ctrl-c**

First RESTful Route

Let's commit code changes to get

git status shows modified files

git add -u add modified files

git commit -m "hapi helloworld"

Create Some Data

Create a new file **games.json** & save it in the **src** directory with the following:

```
[
  {
    "id": 1,
    "name": "Tic-Tac-Toe"
  },
  {
    "id": 2,
    "name": "Checkers"
  },
  {
    "id": 3,
    "name": "Chess"
  }
]
```

Add this file to our main.js after the var server line (around line 3)

```
var games = require('./games.json');
console.log(games);
```

star the application **npm start** and the result should be the following:

```
[ { id: 1, name: 'Tic-Tac-Toe' },
  { id: 2, name: 'Checkers' },
  { id: 3, name: 'Chess' } ]
Server running at  3000
```

Check-in change

git status
notice a new untracked file games.json is listed **git add src/games.json** to add it
add modified files **git add -u**
commit change list **git commit -m "add games list"**

First RESTful Route

Add another route to the main js file. Add this just before server.start (about line 17). This route will return a full list of all games.

```
server.route( {
    method: 'GET',
    path: '/games',
    handler: function (request, reply) {
        reply(games);
    }
});
```

Start the server **npm start**

In the browser **http://localhost:3000/games**

[{"id":1,"name":"Tic-Tac-Toe"},{"id":2,"name":"Checkers"},{"id":3,"name":"Chess"}]

Check in changes

stage changes in modified files **git add -u**

commit **git commit -m "added games endpoint"**

Second Endpoint

The last route returned a complete list of games. Let's return just a game by its id. To do this we will use a library called lodash which we need to install first

npm install --save lodash

At the top of the main.js file using the _ (underscore symbol is common for the library lodash. There is also a library called underscore that is very similar)

var _ = require('lodash');

```
server.route( {
    method: 'GET',
    path: '/games/{id}',
    handler: function (request, reply) {
        var game = _.find(games, {'id': parseInt(request.params.id, 10)});
        reply(game);
    }
});
```

In the function .find the first parameter (games is the data being searched. The second parameter an object with of what to search for in games. In this case search the property 'id' for the value **request.params.id** which is what is sent in the path {id}. parseInt is converting it to a number.

Validation

The /games/{id} endpoint works, but we can validate that id is a number using a library called joi. Let's install this library

Install joi **npm install --save joi**

Add require statement to top of main.js **var Joi = require('joi');**

Modify the /games/{id} add the **config** object and remove the parseInt function:

```
server.route({
    method: 'GET',
    path: '/games/{id}',
    handler: function (request, reply) {
        // var game = _.find(games, {'id': parseInt(request.params.id, 10)});
        var game = _.find(games, {'id': request.params.id});
        reply(game);
    },
    config: {
        validate: {
            params: {
                id: Joi.number().integer().min(1).required()
            }
        }
    }
});
```

By adding the config object the param id is being converted to a number that must be an integer (no decimal) and a minimum value of 1. It is also required.

Start the server **npm start**

From the browser try the following:

http://localhost:3000/games/1 http://localhost:3000/games/2

http://localhost:3000/games/x http://localhost:3000/games/0 http://localhost:3000/games/-99

Check-in changes (Do you remember the steps?)

Boom – return html errors

Hapi has a library to return html error codes easily. For example 404 error if an a game is not found. For example, /games/4 doesn't exist.

Install boom **npm install –save boom**

Require boom **var Boom = require('boom');**

Change the endpoint **/games/{id}** add the **if block**

```
        var game = _.find(games, {'id': request.params.id});
        if (!game){
            return reply(Boom.notFound('game id not found'));
        }
        reply(game);
```

The if (!game) will be true if game is not found. The ! is a not operator. i.e. !true is false

Note: It is a good idea to always return reply(). This avoids an issue of replying twice.

Start the server

Try **http://localhost:3000/games/4**