The University Of Sheffield.

Lab Class for Week 3

Json, Ajax and callbacks

Professor Fabio Ciravegna
f.ciravegna@shef.ac.uk

2 exercises

- Learning to pass JSON to a nodejs server using Ajax
- Understanding callbacks
  - we will see also how to communicate between two servers
- Server to server communication

- The exercise is per se very simple
  - however you are required to put together a number of concepts that are non trivial

Exercise 1

- An HTML form will allow to input name, surname and year of birth of a person
- The form data is sent via Ajax/Json to a nodejs server which will return the same Json object with added age

My Form

First name: Mickey

Last name: Mouse

Year of Birth: 1900

Submit

{"firstname":"Mickey","lastname":"Mouse","year":"1900","age":118}

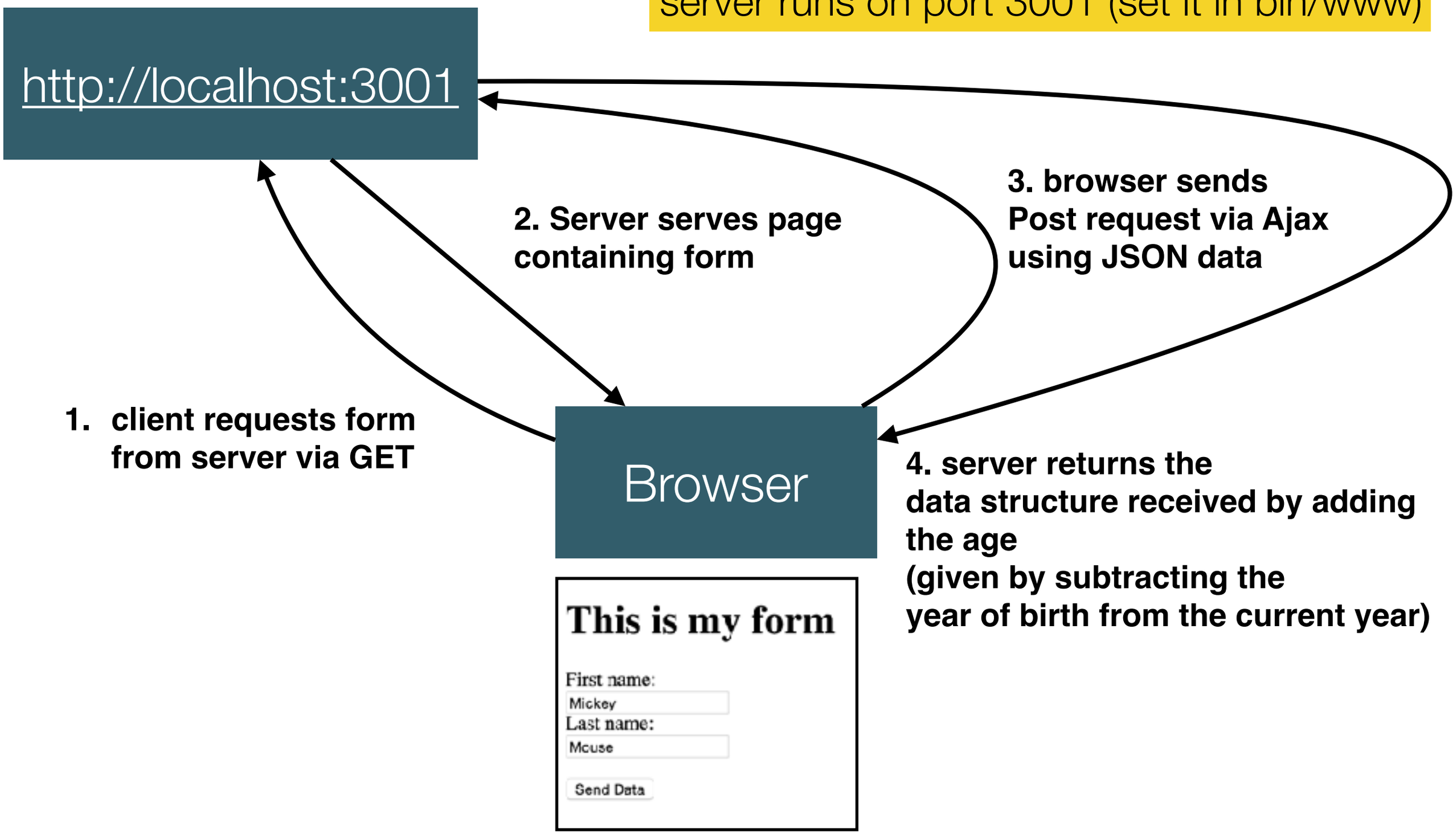
The University Of Sheffield.

P.S.

- Obviously you could do this exercise entirely on the client side without going to the server
  - but the point is to learn to send data to a server using Ajax and Json, so please do as asked

How would you do it?

Note: make sure that the server runs on port 3001 (set it in bin/www)



the client

- receives the filled form via javascript
  - remember that the onsubmit function should return false otherwise the browser will post to the server without going through Ajax
  - serialise the content of the form
    - i.e. you create an object containing the information
  - stringify it
  - send it via Ajax
  - receive it on the server side using router.post
  - access the parameter via body-parser

The form

- You should be able to write the form yourself, but here it is

```
<form id="xForm" onsubmit="onSubmit()">
      <label for='firstname'>First name:</label>
      <input type="text" name="firstname" id="firstname" value="Mickey">
      <p><label for='lastname'>Last name:</label>
              <input type="text" name="lastname" id="lastname" value="Mouse">
      </p>
      <p><label for='year'>Year of Birth:</label>
              <input type="number" name="year" id="year" value="1900">
      </p>
      <p><input type="submit" name="g" value="Submit" id="g"></p>
  </form>
<div id="results" style="margin-top:30px"></div>
```

- now implement the onSubmit function in javascripts/index.js

Remember

- This is how you serialise the information in a form:

```
var formArray= $("form").serializeArray();
    var data={};
    for (index in formArray){
        data[formArray[index].name]= formArray[index].value;
    }
// now data has a form like
// {name:"Mickey", surname: "Mouse", …}
```

- This is how you stringify a Javascript object
    - JSON.stringify(object);

The server side

- in routes/index.js you will define the post

```
router.post('/index', function(req, res, next) {
```

- this is how you access the object sent by the client

```
router.post('/index', function(req, res, next) {
    var userData = req.body;
   …
```

- no need do JSON.parse(req.body) as already done by body-parser
  - so in req.body we have a Javascript object rather than the JSon object the client sent

Useful to know

- This is how you get the current year in Javascript

```
const year = (new Date()).getFullYear()
```

- The age you have received from the form is likely to be a string (unless you declared the field as number in HTML 5)
  - this is how you turn it into a number on tree server side

```
parseInt(userData.year);
```

Expected result

My Form

First name: Mickey

Last name: Mouse

Year of Birth: 1900

Submit

{"firstname":"Mickey","lastname":"Mouse","year":"1900","age":118}

- the data is added to a div underneath the form

Exercise 2

!important

create a new project for this exercise as we will reuse the server built in exercise 1 as is
(i.e. do NOT modify exercise 1 to implement the new server)

Exercise 2

- The browser sends an array of the same Person structure we have used in exercise 1, e.g.

```
[
{"firstname":"Mickey","lastname":"Mouse","year":"1900"
},
{"firstname":"Minnie","lastname":"Mouse","year":"1908"
}
]
```

- and adds the age to all the Person objects sent by the client

My Form

Json Array of People:

[{"firstname":"Mickey","lastname":"Mouse","year":"1900"},
{"firstname":"Minnie","lastname":"Mouse","year":"1908"}]

Submit

How to do it

- To simplify, create a form that receives the JSON array in a big <textarea>

```
<form id="xForm" onsubmit="onSubmit()">
      <p><label for='array'>Json Array of People:</label></p>
      <p><textarea rows="4" cols="50" type="text" name="array_of_people"
             id="array_of_people" >

     </textarea></p>
      <input type="submit" name="g" value="Submit" id="g">
  </form>
<div id="results" style="margin-top:30px"></div>
.
```

- Insert into the textArea:

```
 [{"firstname":"Mickey","lastname":"Mouse","year":"1900"},
 {"firstname":"Minnie","lastname":"Mouse","year":"1908"}]
```

- send the content of the form (as we did in exercise 1) to a server via Ajax using JSon
  - this time the data must be sent to localhost:**3001**

- Then create a nodeJS server working **on port 3000**
  - (set the port in bin/www)
- The server will receive the array and
    - for each element it will send a request to the server developed in exercise 1
      - which must still be running on localhost:3000
- The server will collect all the responses given by localhost:3001
  - when the ages of all Person objects in the list have been collected
    - it will send back the modified array of Person objects to the client

Result on browser

My Form

Json Array of People:

```
[{"firstname":"Mickey","lastname":"Mouse","year":"1900"},
{"firstname":"Minnie","lastname":"Mouse","year":"1908"}]
```

Submit

[{"firstname":"Mickey","lastname":"Mouse","year":"1900","age":118},{"firstname":"Minnie","lastname":"Mouse","year":"1908","age":110}]

- the data is added to a div underneath the form

The University Of Sheffield.

2. server posts one Person object at a time

http://localhost:3001

[{"firstname":"Mickey","lastname":"M

http://localhost:3000 (exercise 1)

3. server 3000 returns the modified Person object with added age

1. client POSTS list of Person objects

4 server sends back the modified array of \ Person objects

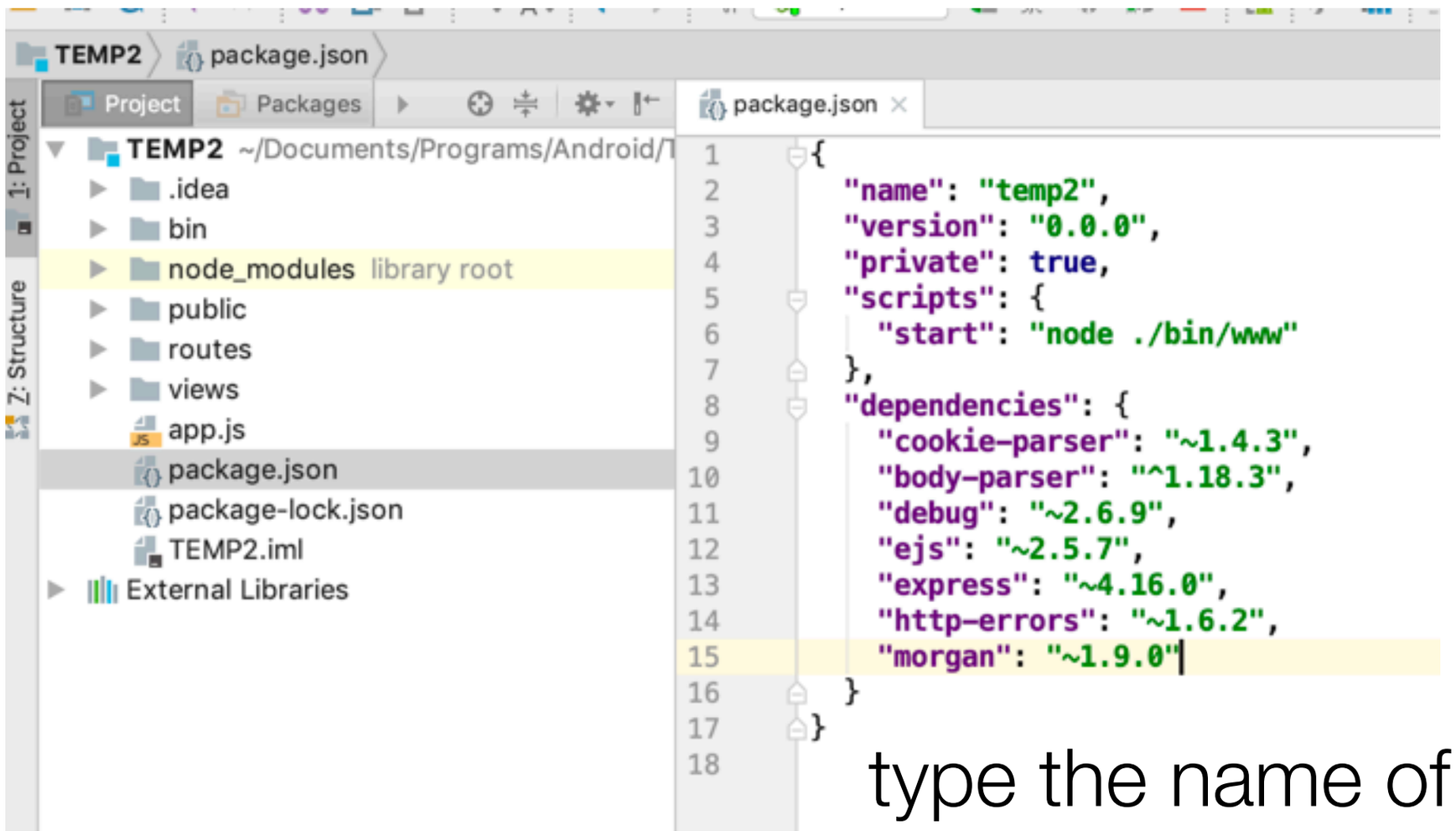
Browser

This is my form

[{"firstname":"Mickey","lastname":"M
{"firstname":"Minnie","lastname":"M

Installing modules

- IF you need to install some modules do this:
-



type the name of the module

The University Of Sheffield.

Accept the suggestions

```
    "body-parser": "^1.18.3",
    "debug": "~2.6.9",
    "ejs": "~2.5.7",
    "express": "~4.16.0",
    "http-errors": "~1.6.2",
    "morgan": "~1.9.0",
    "request
  }
}
```

request	Simplified HTTP request client
request-promise	The simplified HTTP request client 'reque…
request-progress	Tracks the download progress of a reques…
request__no_405	[tmp fork] Simplified HTTP request client.

Press ^. to choose the selected (or first) suggestion and insert a dot afterwards >>

dependencies

```
    "body-parser": "^1.18.3",
    "debug": "~2.6.9",
    "ejs": "~2.5.7",
    "express": "~4.16.0",
    "http-errors": "~1.6.2",
    "morgan": "~1.9.0",
    "request": ""
  }
}
```

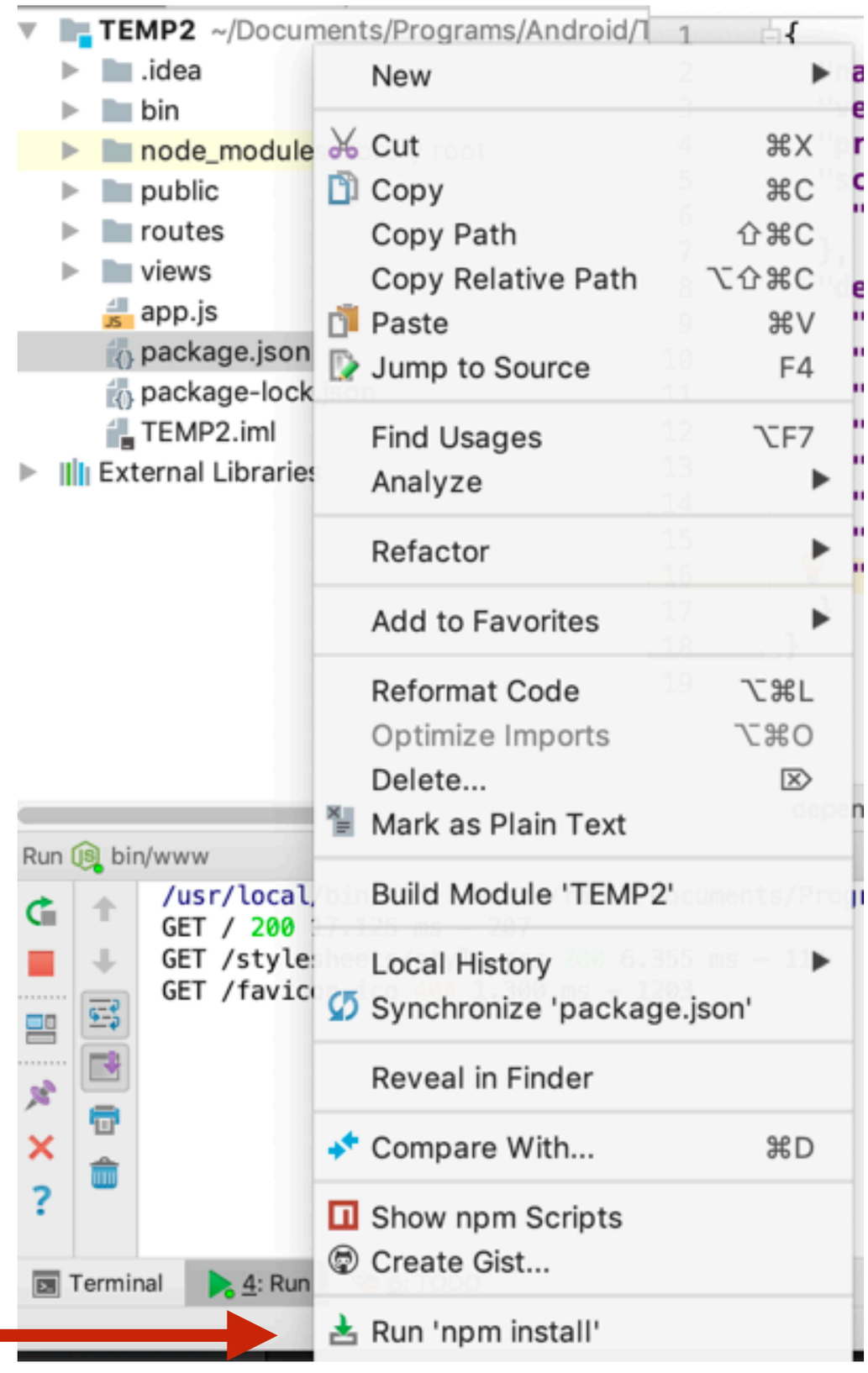
^2.88.0	latest
~2.88.0	latest
2.88.0	latest

^↓ and ^↑ will move caret down and up in the editor >>

dependencies › request

Run NPM Install

- right click on package.json



Run NPM Install

The Ajax part

```
function sendAjaxQuery(url, data) {
    const input= JSON.stringify(data);
    $.ajax({
        url: url ,
        data: input,
        contentType: 'application/json',
        type: 'POST',
        success: function (dataR) {
            // no need to JSON parse the result, as we are using
            // contentType:json, so JQuery knows it and unpacks the
            // object for us before returning it
            var ret = dataR;
            // in order to see the object
           // we need to JSON stringify it
            document.getElementById('results')
                                    .innerHTML= JSON.stringify(ret);
        },
        error: function (xhr, status, error) {
            alert('Error: ' + error.message);
        }
    });
}
```

Server on 3001

- Will receive an Array of Person objects

```
router.post('/index', function (req, res, next) {
        var userDataArray = req.body;
        …
```

- then it will cycle on the array and post to the server on 3000 each stringify-ed Person Object

Remember

- How to post to another server:
  - use the response node module

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

  - set the header and options

```
var headers = {
    'User-Agent': 'me me me',
    'Content-Type': 'application/json'
}
// Configure the request
var options = {
    url: ‘http://localhost:3000/index',
    method: 'POST',
    headers: headers,
}
```

  -

The University Of Sheffield.

- then add the Json data when cycling on all elements

```
for (index in userDataArray) {
   options.json = userDataArray[index];
   // Send the request
   request(options, function (error, response, body) {

      … add your callback body here …
```

- You do not need to stringily the data before sending because **`request`** does so automatically when you declare the data to be JSON in the options

```
options.json= …
```

Remember

- The point of the exercise is to make sure you learn to control the callbacks
  - remember the example of the fast food:
    - you must keep a list of items you need to receive
    - when the callback realises that all the element have been processed, then it will return the complete structure

So...

```
for (index in userDataArray) {
    options.json = userDataArray[index];
    // Start the request
    request(options,
        function (error, response, body) {
            if (!error && response.statusCode == 200) {
                userDataArray[counter] = body;
```

- here is where you have to put the control of the callbacks (i.e. decide how to return

```
res.setHeader('Content-Type', 'application/json');
res.send(JSON.stringify(userDataArray));
```

- when userDataArray is completed (hint: do not leave before the burger arrives)

Good luck

Solutions will be published on Mole in the next hour

but absolutely you should try to find it yourself