



The
University
Of
Sheffield.

Week 2 Lab Class

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

Plan for today

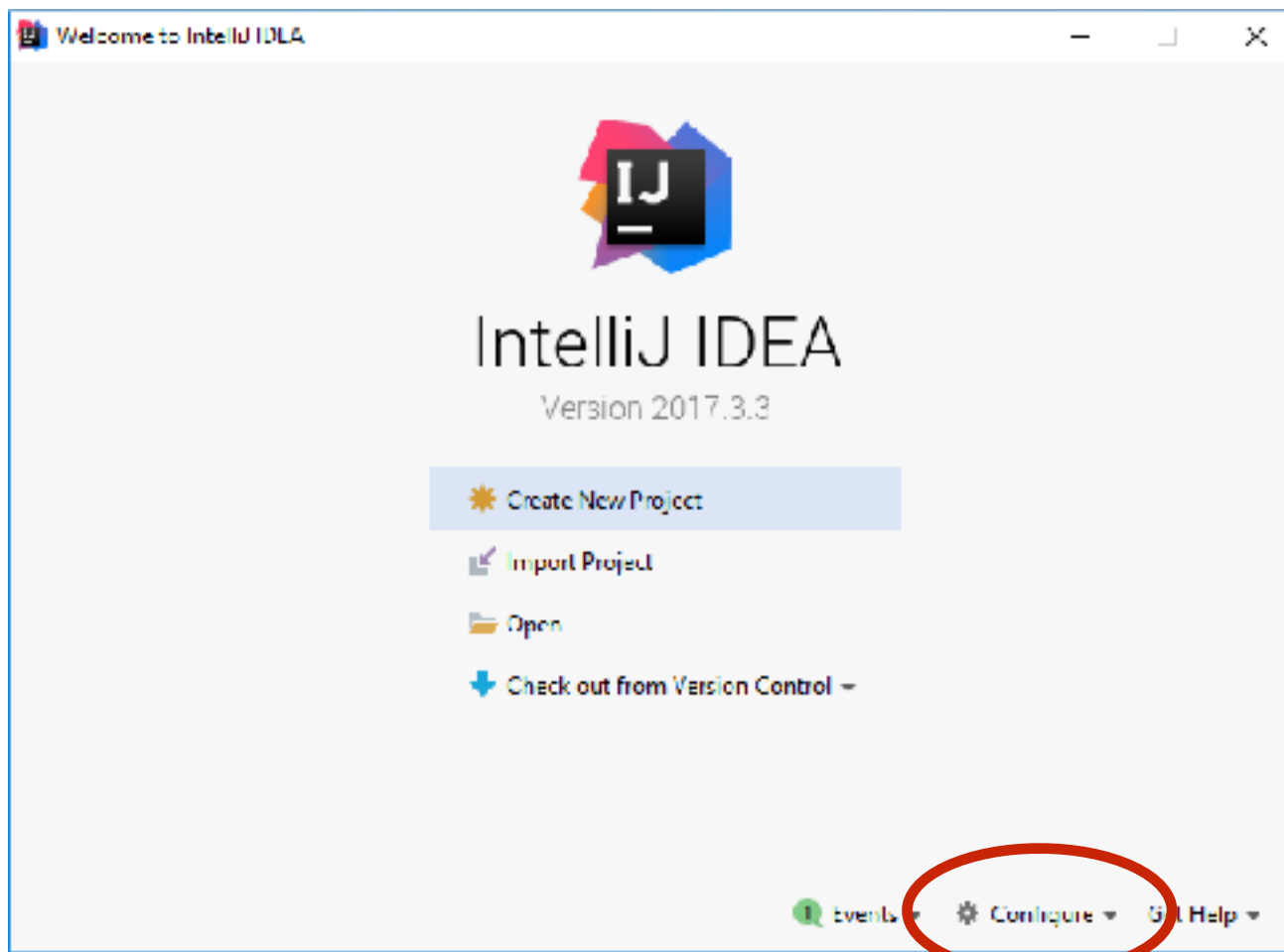
- We have just one hour and there is a lot to go through
- Plan
 - Learn to use IntelliJ (our development environment)
 - Create your first nodeJs server
 - Learn to get a file
 - Learn to post a form
- You are expected to know HTML and Javascript as a starting point

Important!

- Very important:
 - if you do not finish the exercise today,
 - make sure to finish it over the coming week
 - from next week we will build on this
 - if you have not completed the exercises you will struggle
- Also
 - use today as a test of your Javascript and HTML knowledge

Using IntelliJ

- Go to the Window menu and select
 - JetBrains>IntelliJ
 - I had to click a number of times before it worked
- This is what you will see:
-

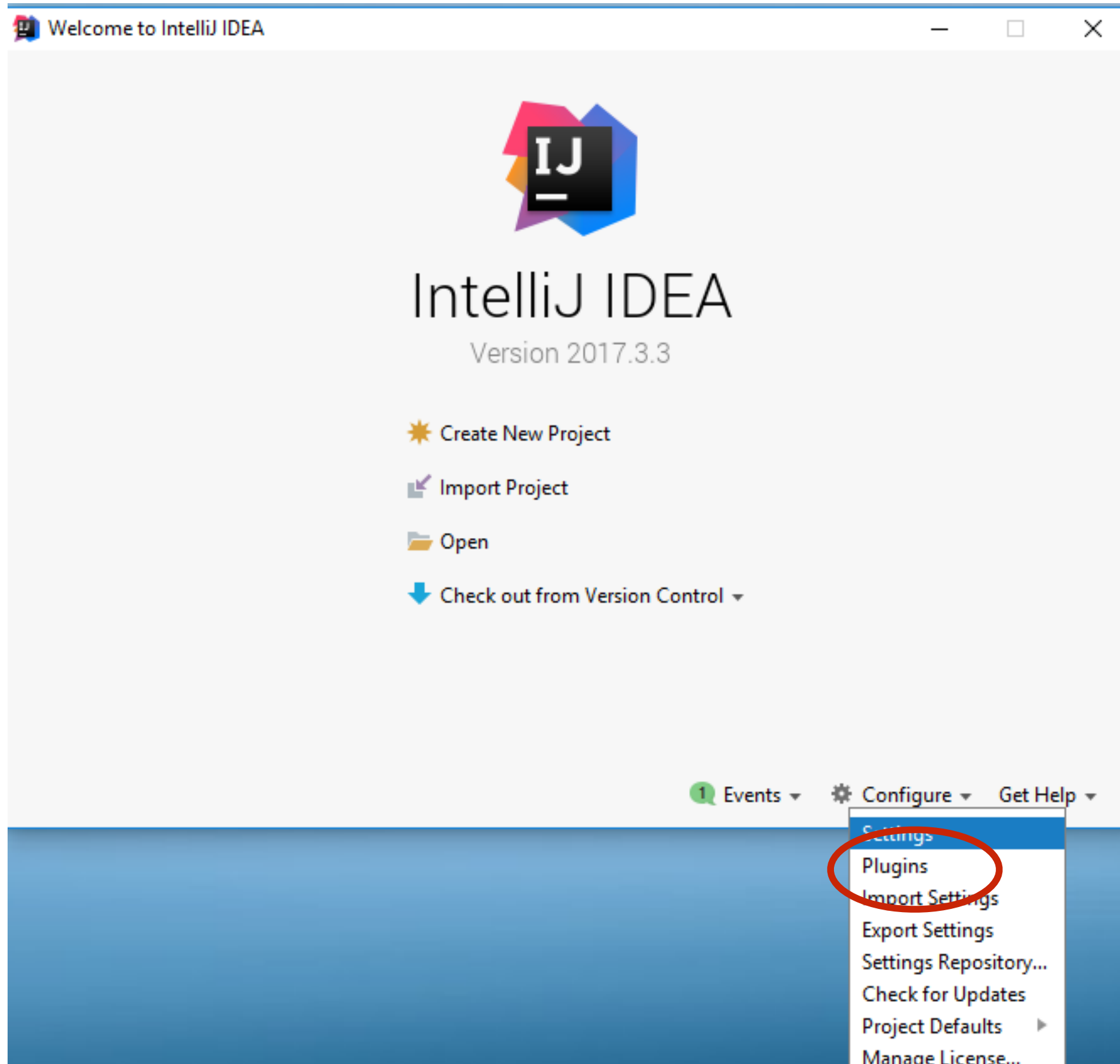


The first time you use that computer, click on configure



The
University
Of
Sheffield.

Install Node Plugin





The

Plugins

Marketplace Installed 2

Search: node

Search Results (55) Sort By: Relevance

NodeJS (highlighted)
↓ 6.8M ☆ 4.2 JetBrains (highlighted)

NodeConfig (highlighted)
↓ 6.5K Flageolet

Node.js Remote Interpreter (highlighted)
↓ 144.2K ☆ 2.4 JetBrains

Node Security (highlighted)
↓ 8.4K hsz

xStructure (highlighted)
↓ 29.8K ☆ 4.6 Sylvain FRANCOIS

Live Edit (highlighted)
↓ 5M ☆ 3.9 JetBrains

Zoolytic - Zookeeper tool (highlighted)
↓ 1.8K ☆ 4.5 Danila Ermakov

NodeJS (highlighted)
↓ 6.8M ☆ 4.2 JetBrains
JavaScript 193.5662.65 Feb 13, 2020

Installed

Plugin homepage ↗
Support for Node.js projects.

Features

screen

You can also run and debug Node.js applications in the remote environments such as Docker containers, Vagrant machines and remote servers right from the IDE. For that, please install the Node.js remote interpreters plugin. Here you can find more information on working with Node.js in the IDE.

Size: 934.4K

Cancel OK

Install it

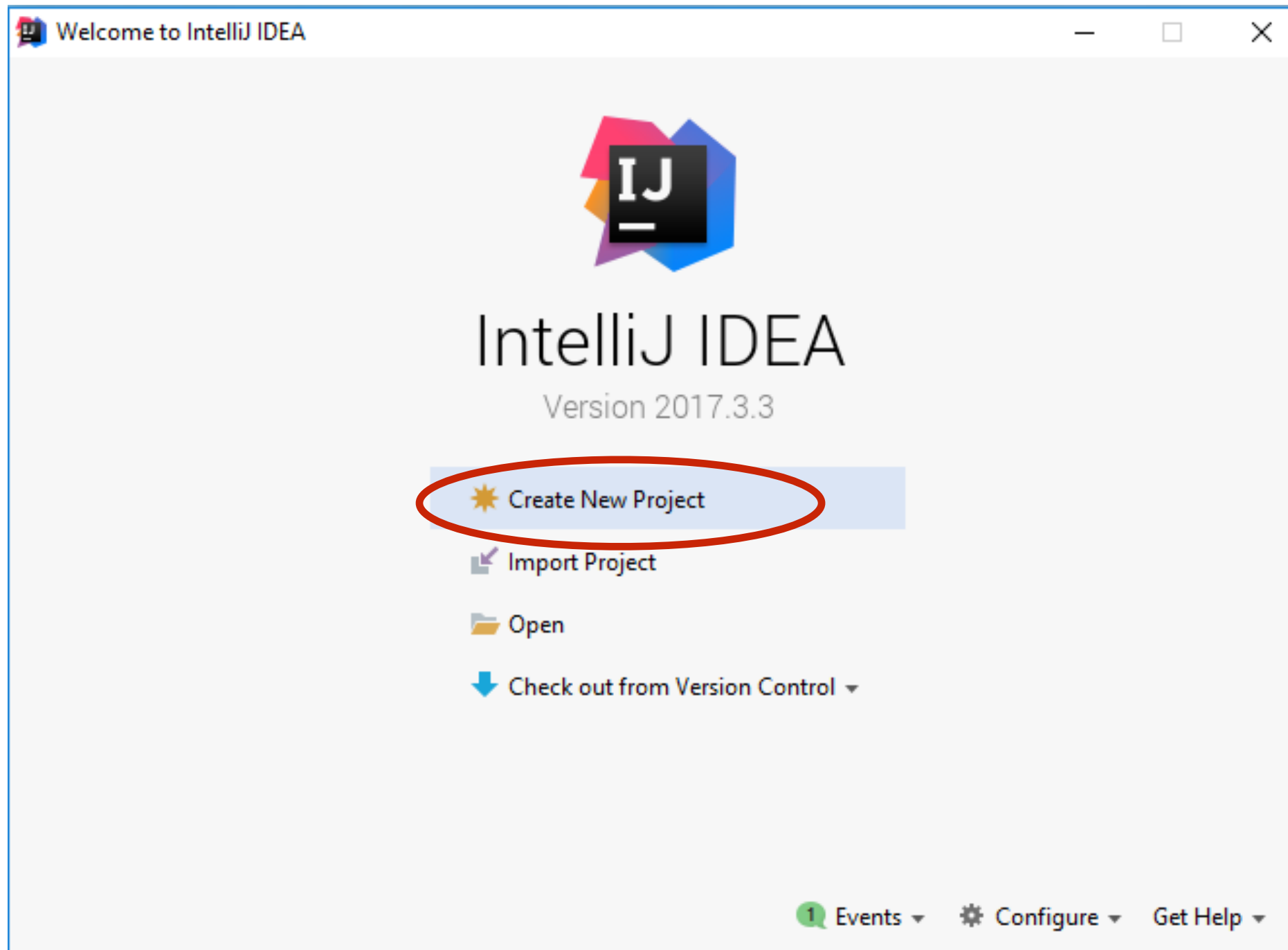
It may look different on the actual IntelliJ version (e.g. background could be white)

then restart IntelliJ



The
University
Of
Sheffield.

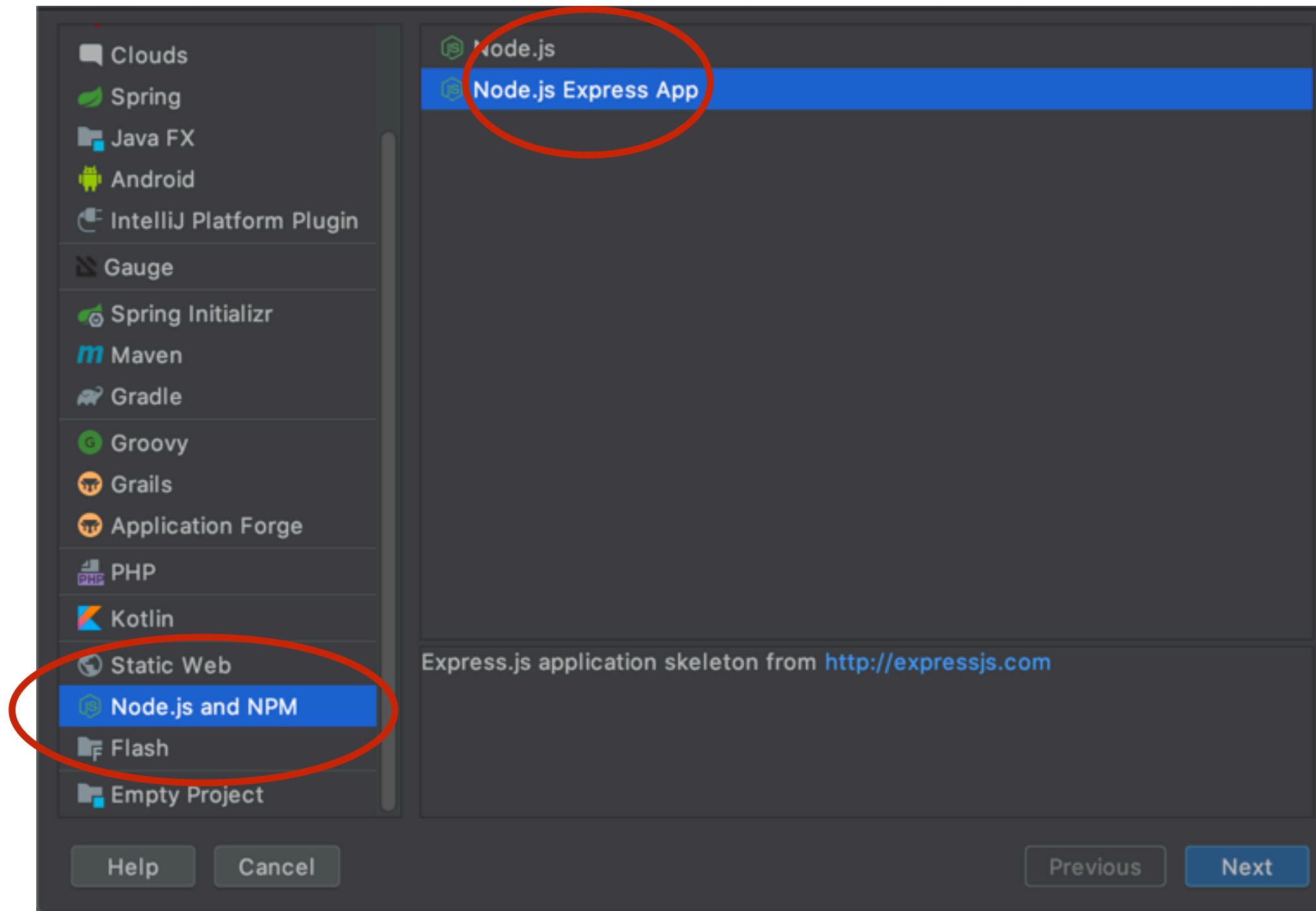
After restarting





The
University
Of
Sheffield.

Select NodeJs





The
University
Of
Sheffield.

Project name:

Project location:

Node interpreter: 10.16.3 ...

Package manager: 6.13.4 ...

Application will be created by [express-generator](#)

Version: ↺

Options

View Engine:

Stylesheet Engine:

▼ More Settings

Module name:

Content root: 📁

Module file location: 📁

Project format:

choose a folder in your U drive

choose EJS

Here is your project

Week1 [U:\Week1] - IntelliJ IDEA

File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help Beame

Week1

Project

Week1 U:\Week1

- > .idea
- > bin
- > node_modules library root
- public
 - images
 - javascripts
 - > stylesheets
- routes
 - index.js
 - users.js
- views
 - error.ejs
 - index.ejs
- app.js
- package.json
- package-lock.json
- Week1.iml

External Libraries

expand routes and views

Search Everywhere Double Shift

Go to File Ctrl+Shift+N

Recent Files Ctrl+E

Navigation Bar Alt+Home

Drop files here to open



Week1 [U:\Week1] - ...views\index.ejs [Week1] - IntelliJ IDEA

File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help Beame

Week1 > views > index.ejs

Project

Week1 U:\Week1

> .idea

> bin

> node_modules library root

> public

images

javascripts

> stylesheets

> routes

index.js

users.js

> views

error.ejs

index.ejs

app.js

package.json

package-lock.json

Week1.iml

External Libraries

index.ejs x

app.js x

index.js x

Plugins supporting *.ejs files found.

1 <!DOCTYPE html>

2 <html>

3 <head>

4 <title><%=

5 <link rel='stylesheet' href='/stylesheets/style.css' />

6 </head>

7 <body>

8 <h1>

9 <p>

10 </body>

note any path starting with /
refers to the folder public

Except for the ejs files which
are in the folder views

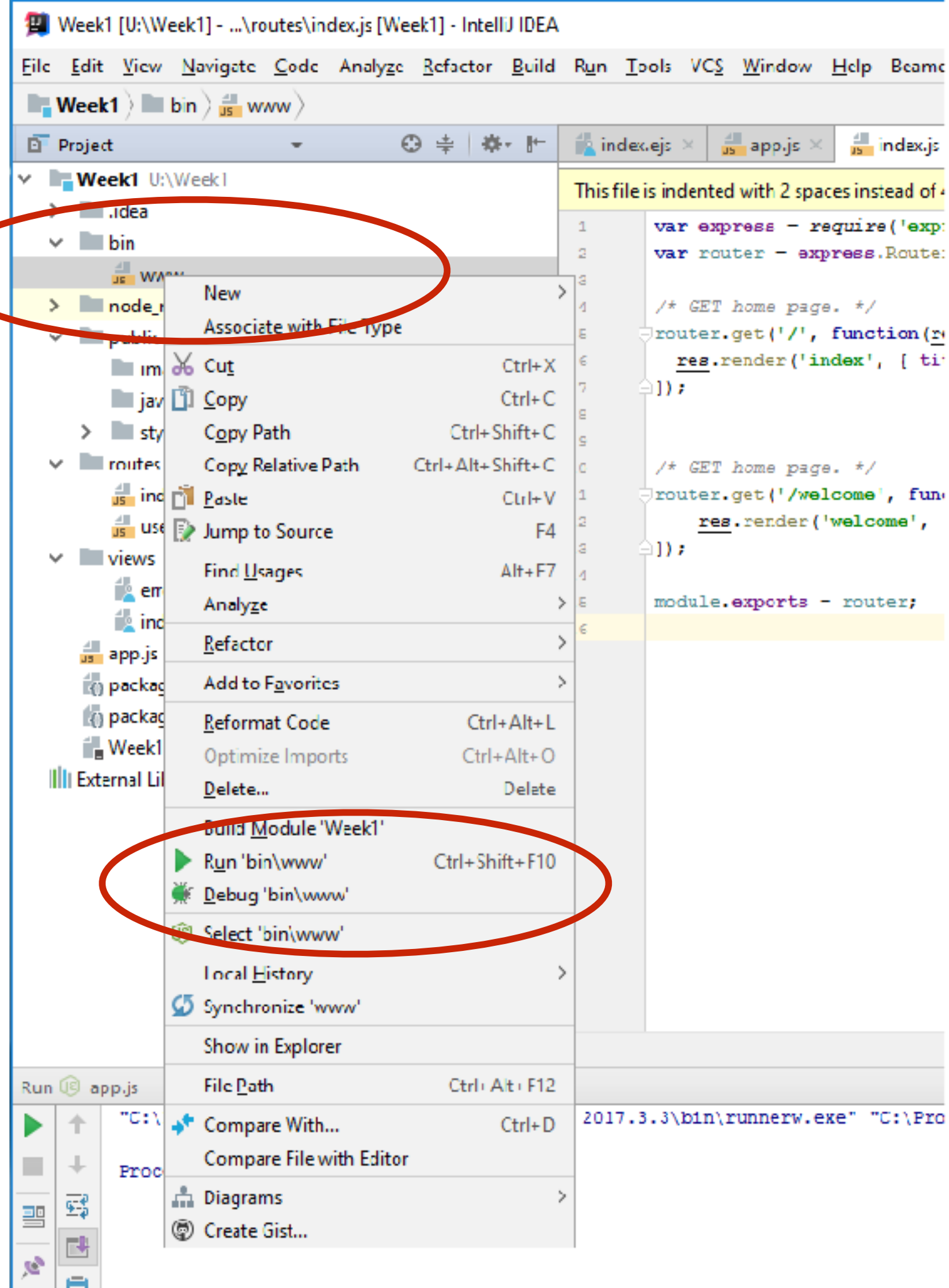
open views/index.ejs,
this is your home file
(what normally would be index.html)

remember: ejs files are html files with
parameters passed by the server!



The
University
Of
Sheffield.

- To run the server
- Right Click on bin>www
- Choose Run





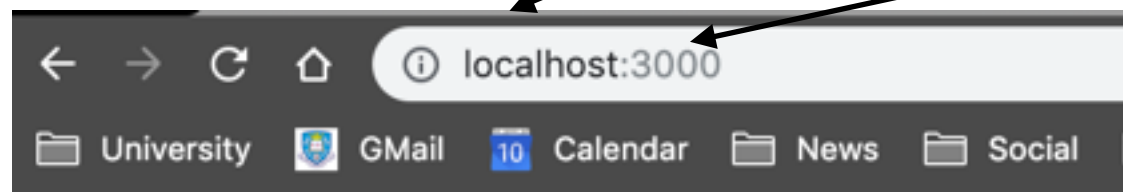
Running the client

open your browser

```
1  #!/usr/bin/env node
2
3  /**
4   * Module dependencies.
5   */
6
7  var app = require('../app');
8  var fs = require('fs-extra');
9  var debug = require('debug')('assignment:server');
10 var https = require('https');
11
12 /**
13  * Get port from environment and store in Express.
14  */
15
16 var port = normalizePort(process.env.PORT || '3000');
```

your server is on localhost
or 127.1.1

the port



what is a port?

- Ports are an old concept from when servers had physical cables entering ports
- you could contact a hardware server through a specific entry point, i.e. a port
 - Nowadays computers have just fibre optic entering them but the concept of ports has been kept
 - Ports are entry points to the physical server
 - You can only have one process (e.g. your node server) running on one port
 - If you try to run a server when another one is running you will get an error telling you that the port is taken
 - If so, either stop the server on that port or run your process on a different port by changing the value 3000 in bin/www

```
• var port = normalizePort(process.env.PORT || '3000');
```

Ports (ctd)

- Ports have values 1–65535 are available, and ports in range 1–1023 are the privileged ones: an application needs to be run as root in order to listen to these ports
 - Suggestion: use ports 3000–3004 or 8080 (standard port) 8090–8092
- If you use 8080 you can omit the port. i.e. http://localhost defaults to http://localhost:8080



The
University
Of
Sheffield.

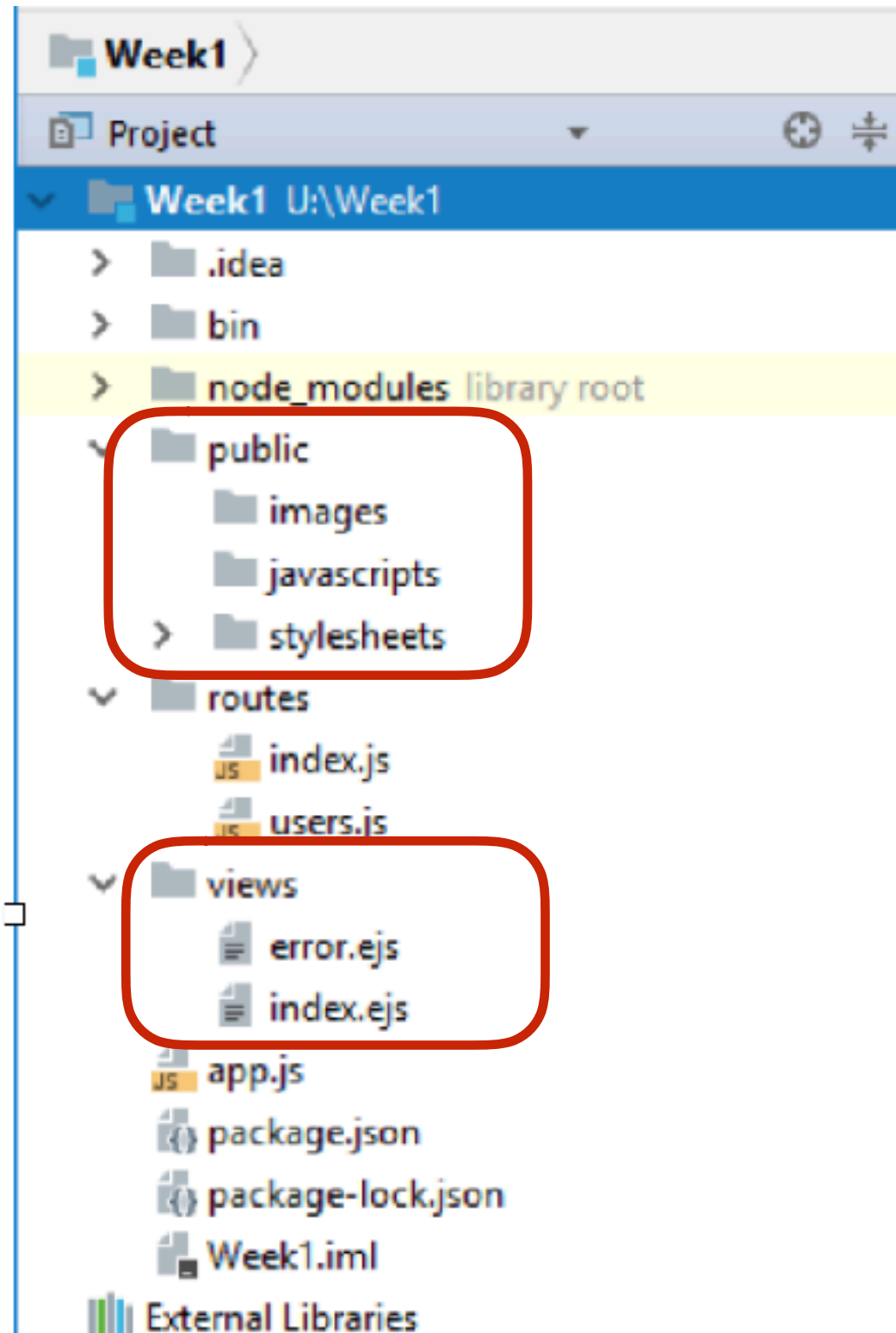
Exercise 1

Exercise one


- Familiarise yourself with the Express structure also using the lecture slides
- then start working on the exercises in the next slides




Exercise 1



- This exercise is designed to make sure that you understand how to use
 - routes and
 - ejs files
- in a GET using Express
- Open the project you have created
 - it will have the standard structure shown on the left side

- 

- 
- The screenshot shows a web browser window. The address bar at the top displays 'localhost:3000'. Below the address bar, there are several tabs or bookmarks labeled 'University', 'Mobilise-D', and 'NewCo'. The main content area of the browser shows the Express.js logo, which is the word 'Express' in a large, bold, black font. Below the logo, the text 'Welcome to Express' is displayed in a smaller, black font.

Learning to manipulate .ejs files

- Your first exercise is to modify views/index.ejs so that it looks like

Welcome to My Class

Please click [getting the other file](#)

- Please note: the term **My Class** should be passed as parameter to the ejs file, so using the notation
`<h1> Welcome to <%= title %> </h1>`
- and having the route in routes/index.js as (as it is already)

```
router.get('/', function(req, res, next) {  
  res.render('index', { title: 'Express' });  
})
```

Following the link

Welcome to My Class

Please click [getting the other file](#)

- insert a link into the html of the eps file so to provide a link to the path

- /welcome

- i.e. the html should be

Please click

```
<a href="/welcome">getting the other file</a>
```

Responding to /welcome

- Create a new route path in **routes/index.js**
 - so that when you click the link, a file called `welcome.ejs` is rendered which will be shown as

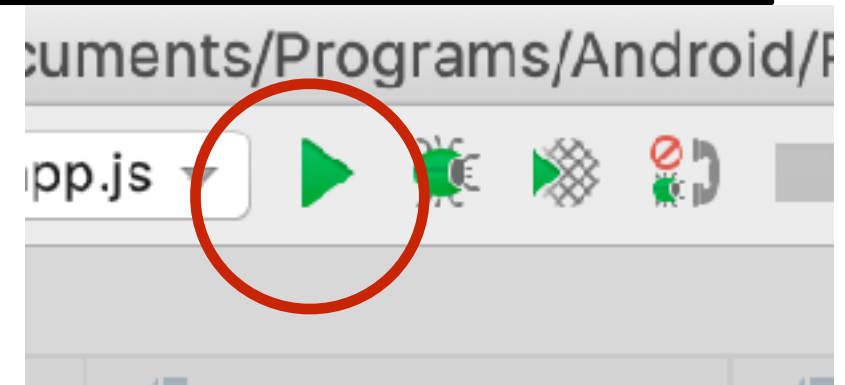
Welcome to COM3504

COM3504 is a cool module

- Note: the string `COM3504` should be changeable from server side to `COM6504` without changing the HTML of the file `welcome.ejs` (i.e. it is a parameter as well)

Note! every time you change the routes files you must reload the server

How would you do it?



Try to think: solution is in the next slides but try to come up with a solution before checking it

Hint 1: create an ejs view called `welcome.ejs` under `views`

Hint 2: you must add a new route path for a GET which returns the rendering of `welcome.ejs` with 'COM3504' as parameter

Solution

- This is the route path that you introduce in `routes/index.js`:

```
router.get('/welcome', function (req, res) {  
    res.render('welcome', { title: 'COM3504' });  
});
```




The
University
Of
Sheffield.

Exercise 2

Posting to a server
and Using the debugger

- In this exercise the path `/` will render an ejs file containing a form
- the form will request login and password

Welcome to My Class

Please fill the form

Login:

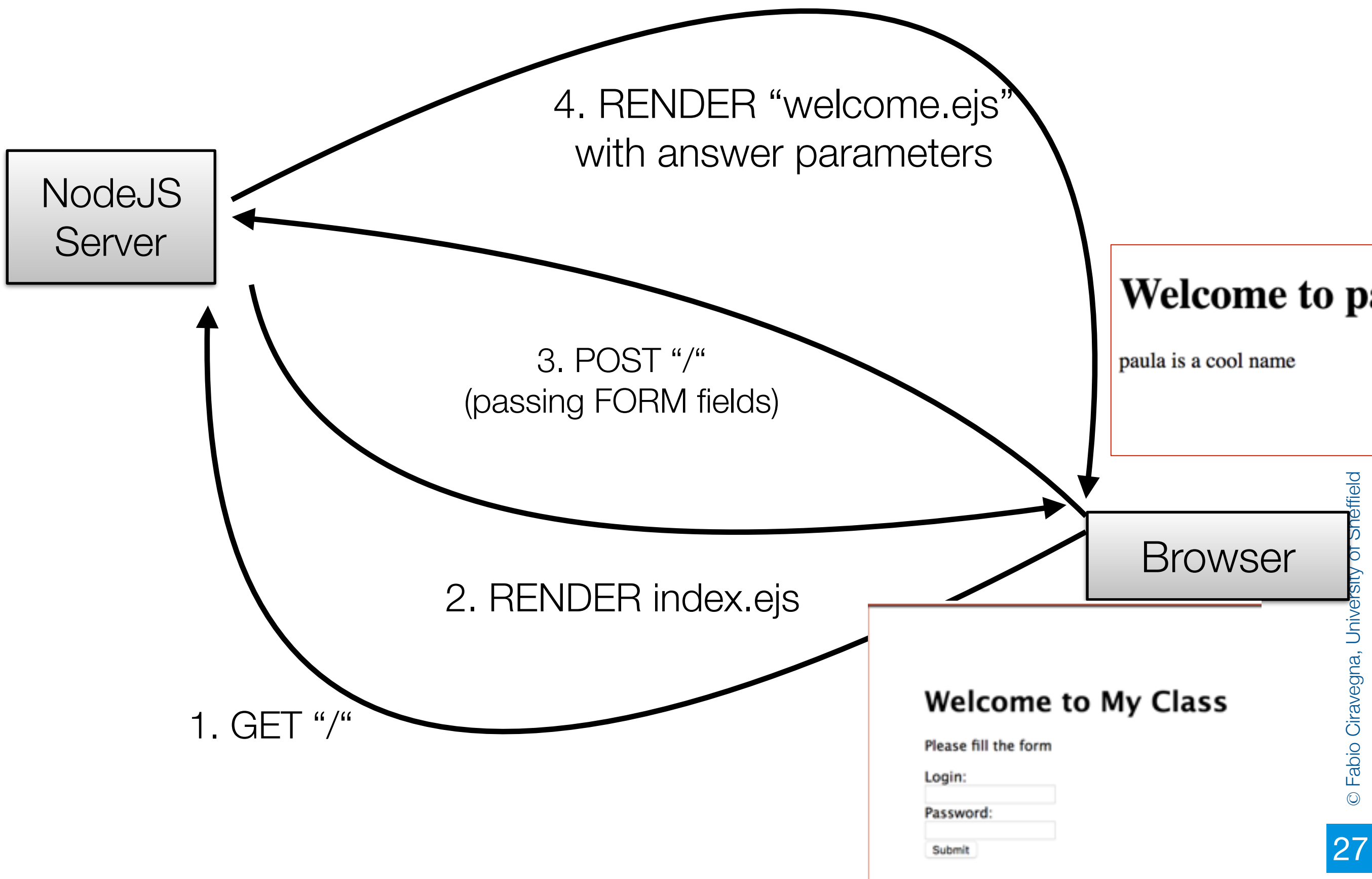
Password:

Submit

- you have to create a route in `routes/index.js` so that it responds to the `login/password`



Client/server interplay



Step 4

- The server will respond differently depending on the value of the field login
- If login value is Paula
- If login value is different from Paula it will return an error

Welcome to paula

paula is a cool name

Welcome to My Class

Please fill the form

Login:

Password:

Submit

localhost:3002 Says

login or password is incorrect

OK

How to

- Modify `views/index.ejs` to include an HTML form asking for login and password.
 - Make sure to use an `html5` form!
- This should post to the server as `POST` on the route `"/welcome"`
 - the form should be:
 - `<form action="/welcome" method="POST">`
- if you do not remember how to create a `FORM` in `HTML`
 - look it up

Welcome to My Class

Please fill the form

Login:

Password:

Submit

Server side

- As the form posts to the route `/welcome`. we must modify the file `routes/index.js` by adding:

```
app.post('/welcome', function(req, res){  
    ...}
```
- replace the three dots with code that will check the parameters passed by the module)
 - hint: use `body-parser` to access `response.body`
- your code should check if the login passed by the user is `'paula'`. If so it should render the file `welcome.ejs` which should look like the one on the right.
 - note: `Paula` should be a parameter

Welcome to paula

paula is a cool name

- otherwise render `index.ejs` again with an alert

Welcome to My Class

Please fill the form

Login:

Password:

localhost:3002 Says

login or password is incorrect

Hint

- Change the “/” route by adding a parameter:

```
router.get('/', function(req, res, next) {  
  res.render('index', { title: 'My Class', login_is_correct: true });  
});
```

- in the POST route for ‘/welcome’ you will

if the login is not ‘paula’ it will render index again but the parameter login_is_correct will be false

```
router.get('/welcome', function(req, res, next) {  
  (...check if login is not paula...)  
  res.render('index', { title: 'My Class', login_is_correct: false });  
});
```

- OR

- render a file named welcome.ejs that will say hello to Paula

```
res.render('welcome', { title: 'Paula' })
```


interpreting the second parameter

- On the client side we should interpret the parameter `login_is_correct`
 - if it is true we just show the page
 - if it is false, we have to show the page and show an alert
- in `views/index.ejs` modify the html code:
`<body onload="checkCredentials(<%= login_is_incorrect %>)">`
- then create `public/javascripts/index.js` and insert the following code

```
function checkCredentials(isLoginIncorrect){  
    if (isLoginCorrect){  
        alert('login or password is incorrect');  
    }  
}
```



The
University
Of
Sheffield.

Debugging

Debugging client & server

- While you develop, try to debug the client server architecture so to identify errors
 - 1. put a break into Chrome to check that the function *checkForErrors* works appropriately

Break points in client side

- open Chrome
- open the page *http://localhost:3000/*
- open the developers tools
- insert break point
- reload page

Welcome to My Class

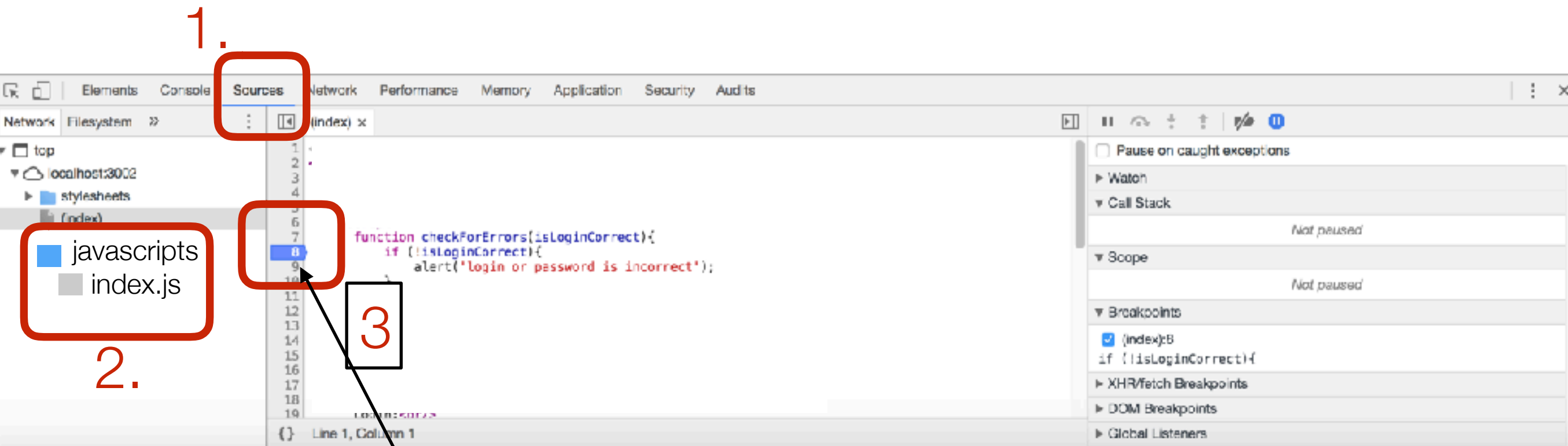
Please fill the form

Login:

ww

Password:

Submit



Click on the line number to set up a break point

2. Click debug

```
1  var express = require('express');
2  var router = express.Router();
3
4  /* GET home page. */
5  router.get('/', function(req, res, next) {
6    res.render('index', { title: 'My Class', login_is_correct: true });
7  });
8
9
10 /* POST from form. */
11 router.post('/welcome', function(req, res, next) {
12   var login= req.body.login;
13   var password= req.body.password;
14
15   if (login=='paula'){
16     res.render('welcome', { title: login, login_is_correct: true });
17   }
18 }
19 );
20
21
22 module.exports = router;
23
24
25
```

1. Click to set break point

Run bin/www

```
/usr/local/bin/node "/Users/fabio/Documents/Teaching/COM3504-6504 Intelligent Web/Lab Classes/Week 2/Week1b/bin/www"
GET / 304 14.619 ms - -
GET /stylesheets/style.css 304 1.428 ms - -
POST /welcome 200 42.057 ms - 192
```

Server (routes) debugging - use IntelliJ's debugger
you cannot use Chrome as inly the client runs
in the browser!



The
University
Of
Sheffield.

Questions?