Lab Class WebStorm

Prof. Fabio Ciravegna

Dipartimento di Informatica

Università di Torino

fabio.ciravegna@unito.it

© Prof. Fabio Ciravegna, Università di Torino

Plan for today

- We have two hours and there is a lot to go through
- Learning Objectives
  - Learn to use WebStorm (our development environment)
  - Create your first nodeJs server
  - Learn to get a file
  - Learn to get an EJS file with parameters
  - Learn to post a form
- You are expected to know HTML and Javascript as a starting point

© Prof. Fabio Ciravegna, Università di Torino

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

© Prof. Fabio Ciravegna, Università di Torino

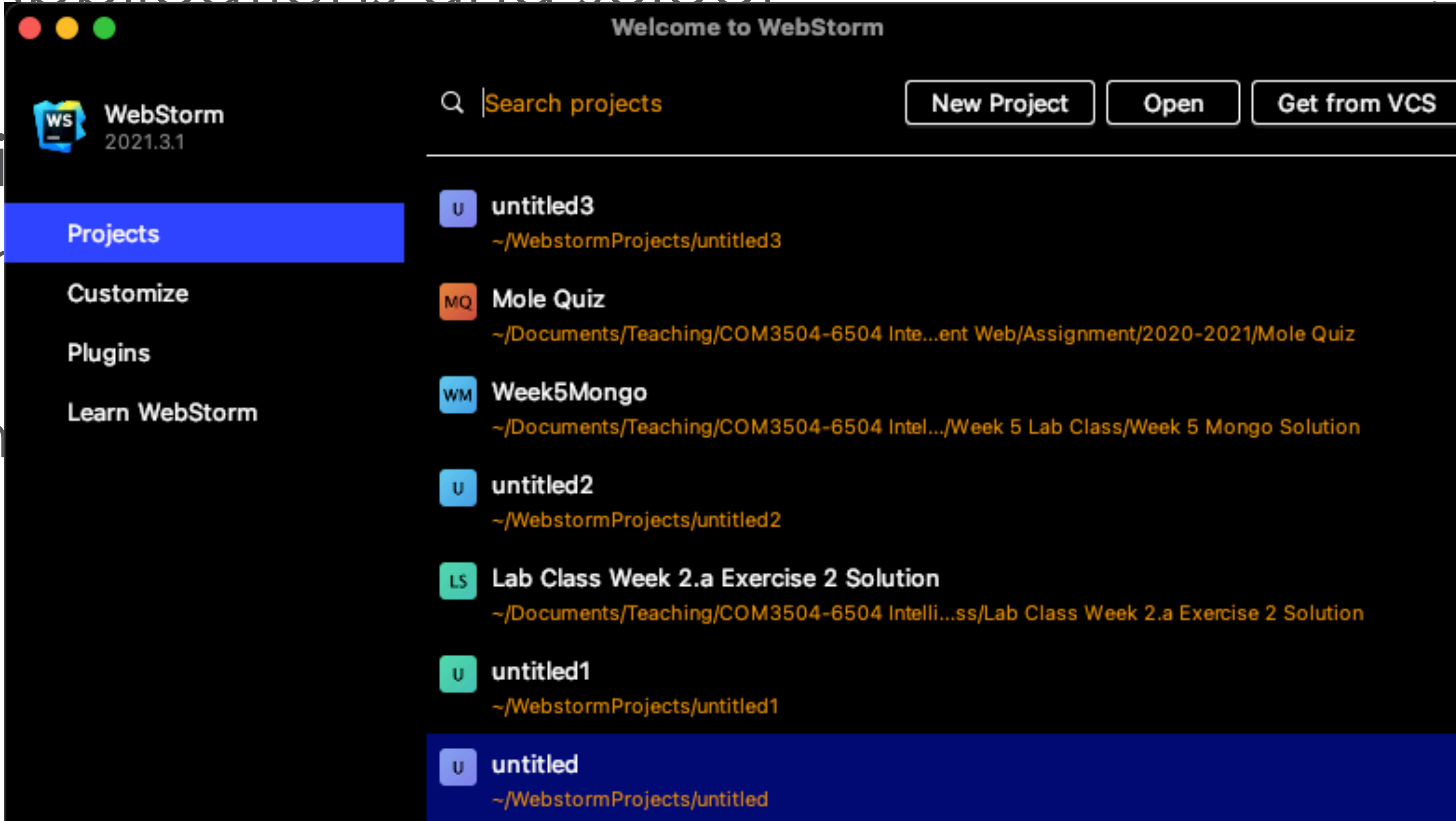
Creating an Express Project in WebStorm

© Prof. Fabio Cravegna, Università di Torino

UNIVERSITÀ DI TORINO

Using WebStorm

- In Windows Go to the Window menu and search
  - WebStorm
- On a Mac go to Applications and select WebStorm
- This is what you wi
- you should have r already installed
  - to be sure: click on


Welcome to WebStorm

WebStorm 2021.3.1

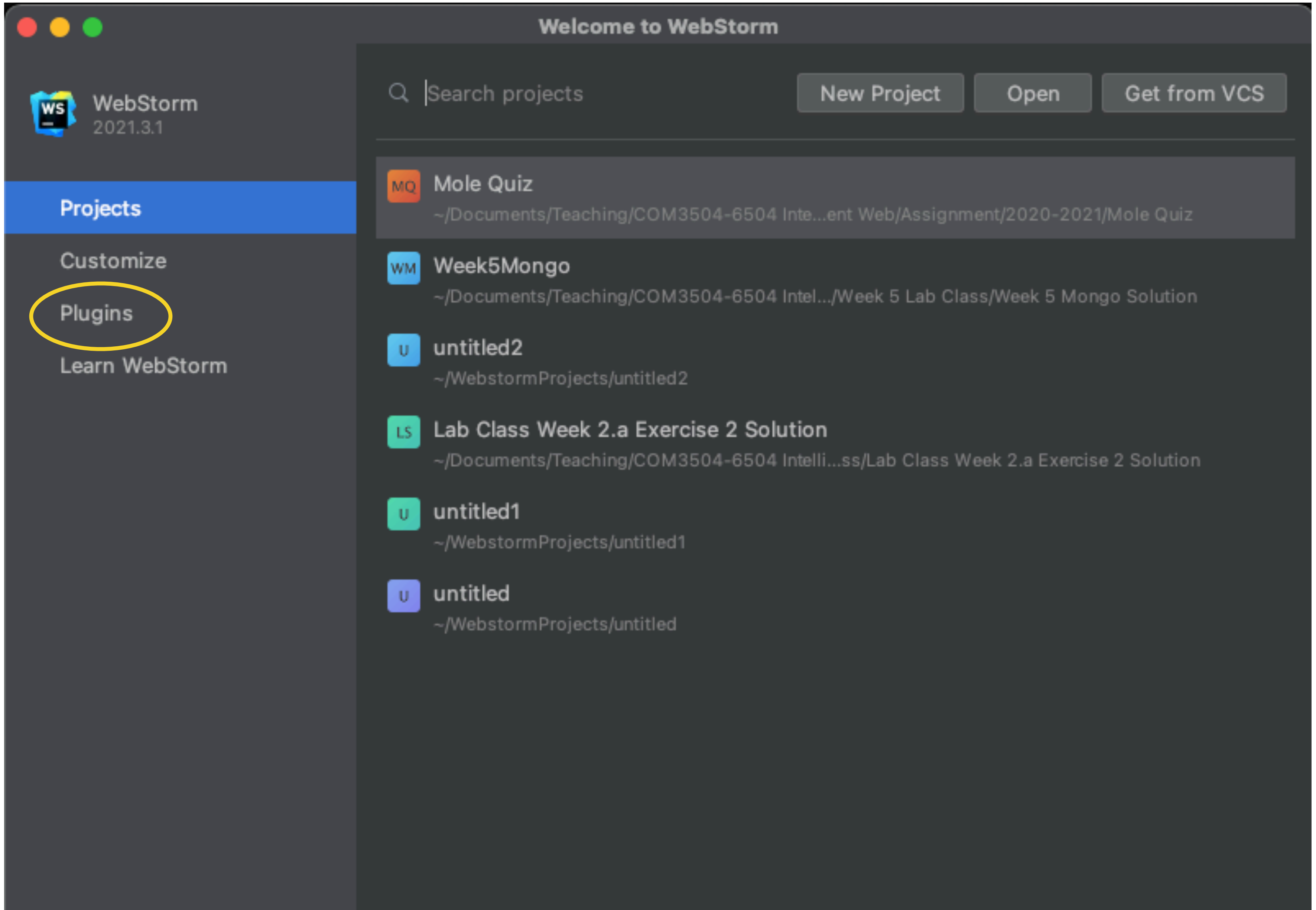
Projects
Customize
Plugins
Learn WebStorm

Search projects
New Project
Open
Get from VCS

untitled3
~/WebstormProjects/untitled3
Mole Quiz
~/Documents/Teaching/COM3504-6504 Inte...ent Web/Assignment/2020-2021/Mole Quiz
Week5Mongo
~/Documents/Teaching/COM3504-6504 Intel.../Week 5 Lab Class/Week 5 Mongo Solution
untitled2
~/WebstormProjects/untitled2
Lab Class Week 2.a Exercise 2 Solution
~/Documents/Teaching/COM3504-6504 Intelli...ss/Lab Class Week 2.a Exercise 2 Solution
untitled1
~/WebstormProjects/untitled1
untitled
~/WebstormProjects/untitled

UNIVERSITÀ DI TORINO

Not on Lab computers: Install Node Plugin



Mole Quiz
~/Documents/Teaching/COM3504-6504 Inte...ent Web/Assignment/2020-2021/Mole Quiz

Week5Mongo
~/Documents/Teaching/COM3504-6504 Intel.../Week 5 Lab Class/Week 5 Mongo Solution

untitled2
~/WebstormProjects/untitled2

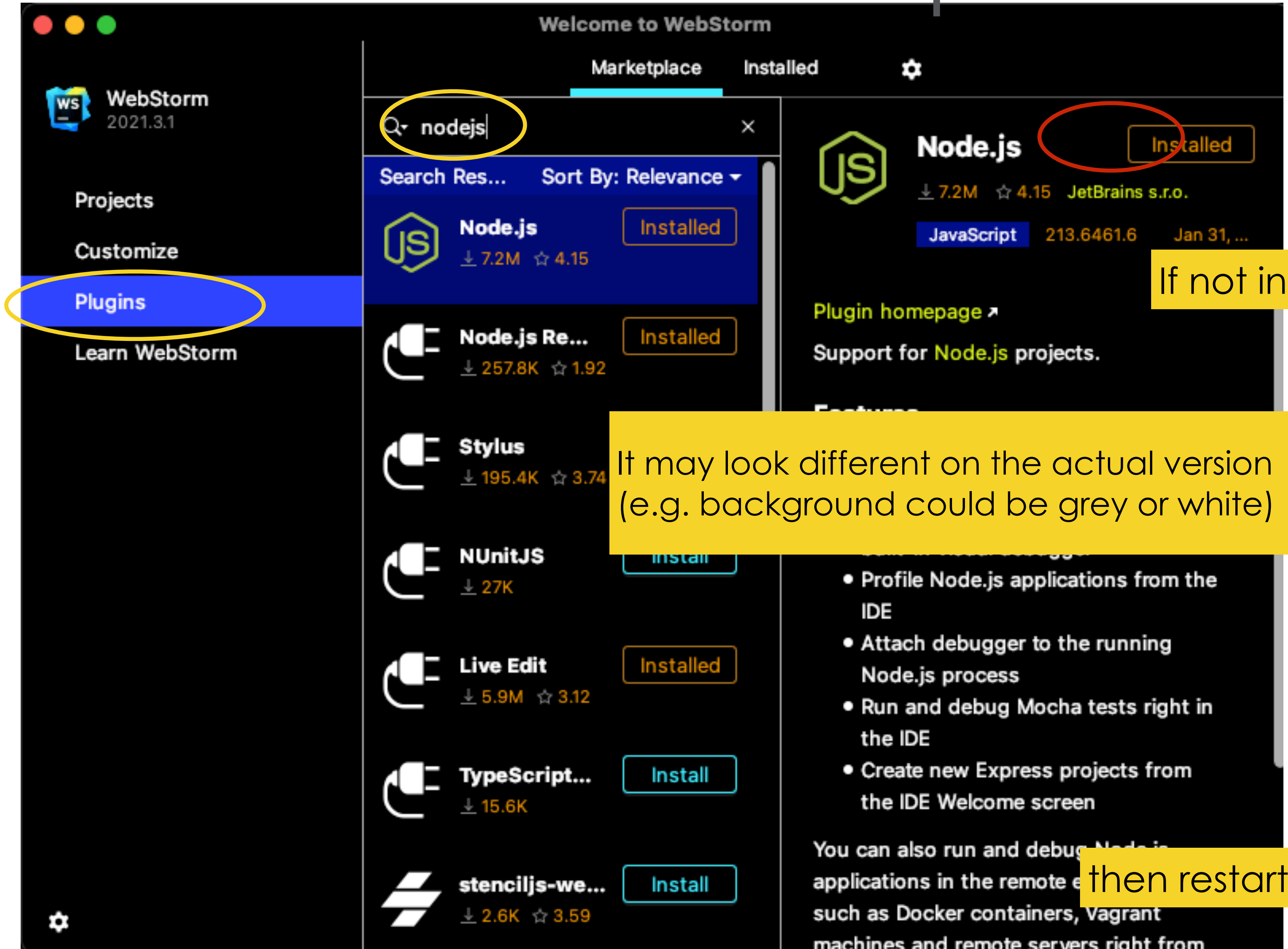
Lab Class Week 2.a Exercise 2 Solution
~/Documents/Teaching/COM3504-6504 Intelli...ss/Lab Class Week 2.a Exercise 2 Solution

untitled1
~/WebstormProjects/untitled1

untitled
~/WebstormProjects/untitled

© Prof. Fabio Ciravegna, Università di Torino

Not needed on lab computers!!


Welcome to WebStorm

WebStorm 2021.3.1

Projects
Customize
Plugins
Learn WebStorm

Marketplace | Installed

nodejs

Search Res... Sort By: Relevance

Node.js	7.2M ☆ 4.15	Installed
Node.js Re...	257.8K ☆ 1.92	Installed
Stylus	195.4K ☆ 3.74	
NUnitJS	27K	Install
Live Edit	5.9M ☆ 3.12	Installed
TypeScript...	15.6K	Install
stenciljs-we...	2.6K ☆ 3.59	Install

Node.js Installed
7.2M ☆ 4.15 JetBrains s.r.o.
JavaScript 213.6461.6 Jan 31, ...

Plugin homepage ↗

Support for Node.js projects.

Features

- Profile Node.js applications from the IDE
- Attach debugger to the running Node.js process
- Run and debug Mocha tests right in the IDE
- Create new Express projects from the IDE Welcome screen

You can also run and debug Node.js applications in the remote e... such as Docker containers, Vagrant machines and remote servers right from

If not installed, Ins

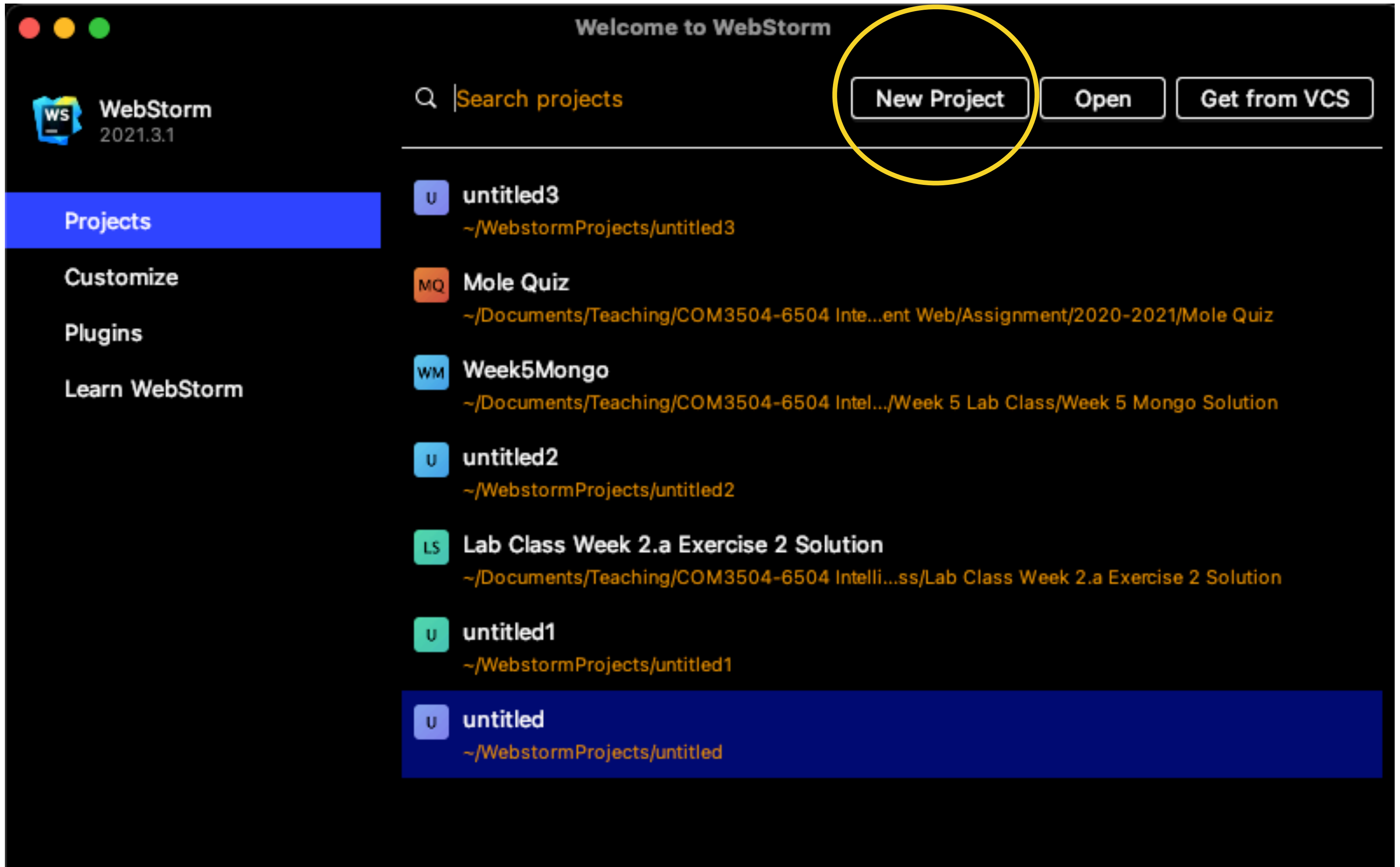
It may look different on the actual version (e.g. background could be grey or white)

then restart WebStorm

© Prof. Fabio Ciravegna, Università di Tor

UNIVERSITÀ DI TORINO

- After installing and restarting, create a new project



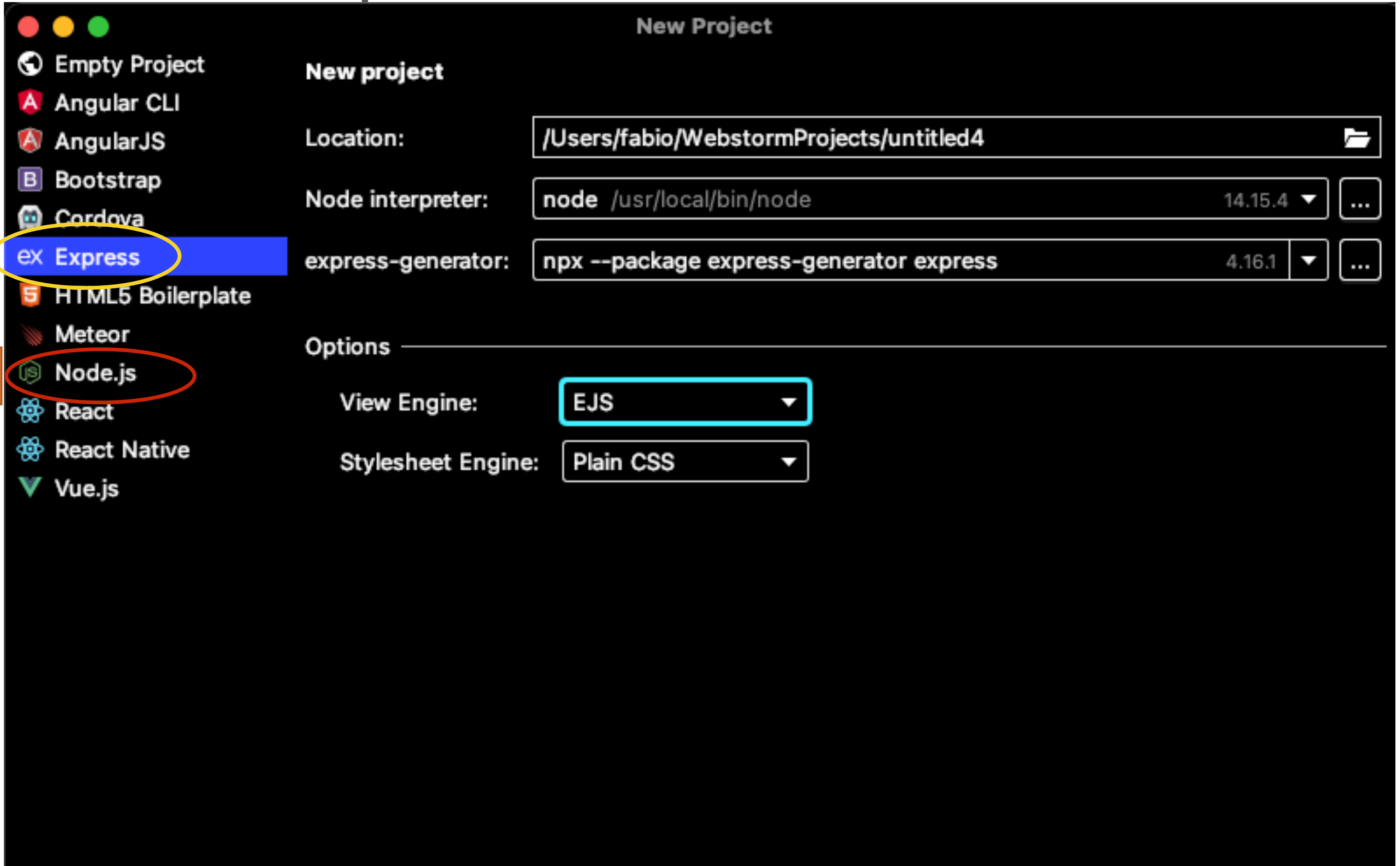
untitled3
~/WebstormProjects/untitled3
Mole Quiz
~/Documents/Teaching/COM3504-6504 Inte...ent Web/Assignment/2020-2021/Mole Quiz
Week5Mongo
~/Documents/Teaching/COM3504-6504 Intel.../Week 5 Lab Class/Week 5 Mongo Solution
untitled2
~/WebstormProjects/untitled2
Lab Class Week 2.a Exercise 2 Solution
~/Documents/Teaching/COM3504-6504 Intelli...ss/Lab Class Week 2.a Exercise 2 Solution
untitled1
~/WebstormProjects/untitled1
untitled
~/WebstormProjects/untitled

© Prof. Fabio Ciravegna, Università di Torino

Select Express

Yes

No!!!



New Project

Empty Project
Angular CLI
AngularJS
Bootstrap
Cordova
Express
HTML5 Boilerplate
Meteor
Node.js
React
React Native
Vue.js

New project

Location: /Users/fabio/WebstormProjects/untitled4

Node interpreter: node /usr/local/bin/node 14.15.4

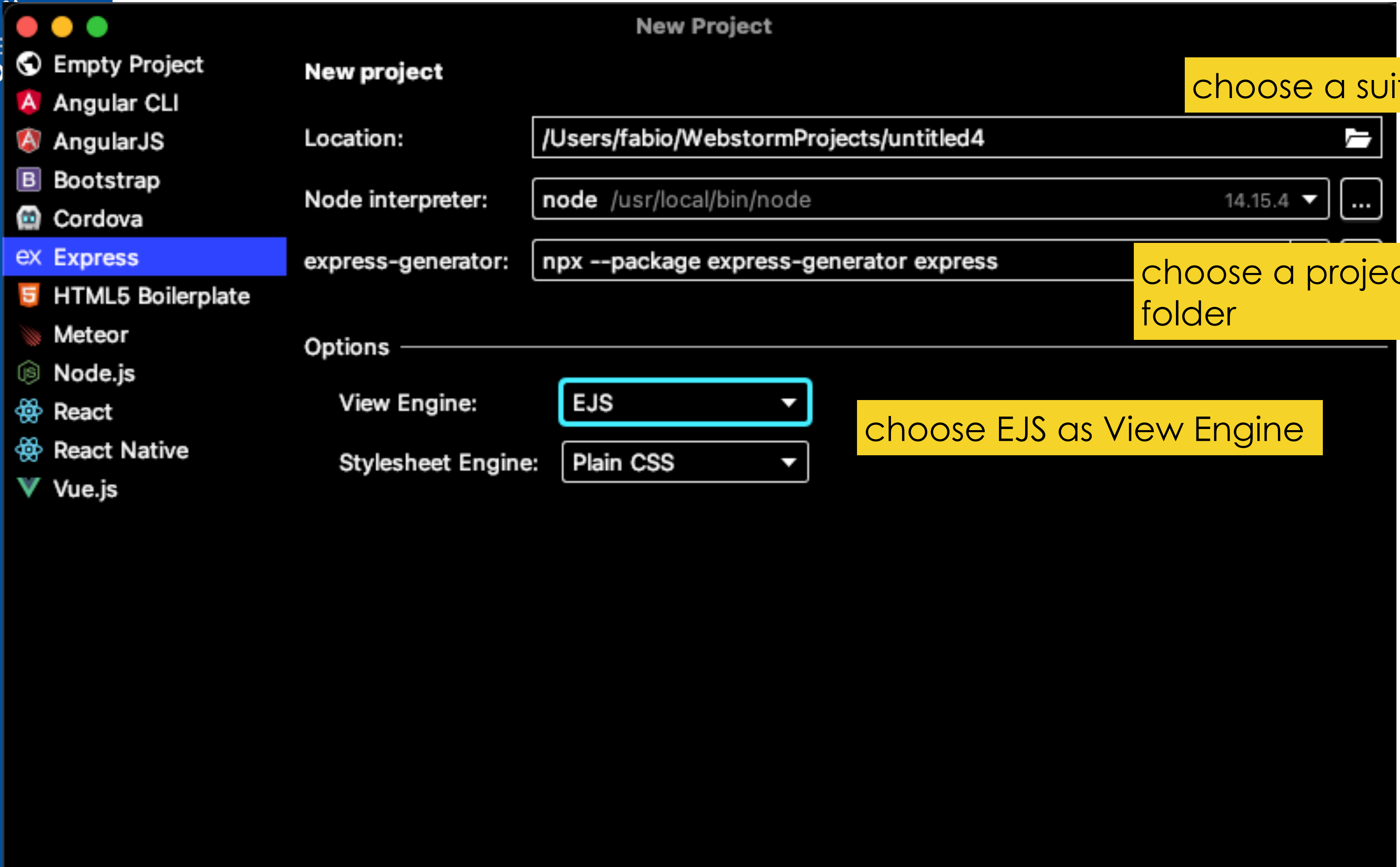
express-generator: npx --package express-generator express 4.16.1

Options

View Engine: EJS

Stylesheet Engine: Plain CSS

© Prof. Fabio Ciravegna, Università di Torino



New project

Location: /Users/fabio/WebstormProjects/untitled4

Node interpreter: node /usr/local/bin/node 14.15.4

express-generator: npx --package express-generator express

Options

View Engine: EJS

Stylesheet Engine: Plain CSS

choose a suitable Project Nam

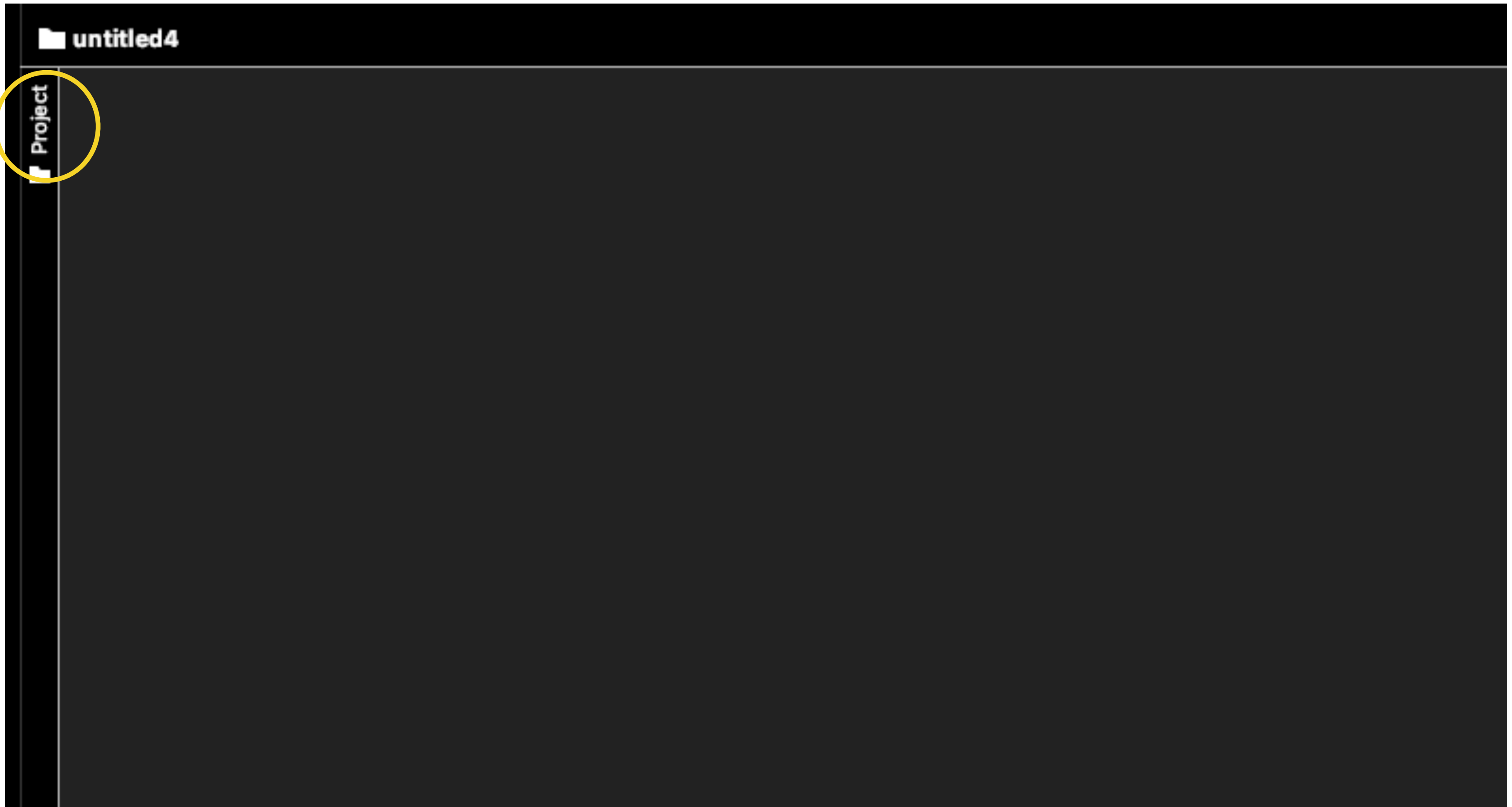
choose a project location folder

choose EJS as View Engine

© Prof. Fabio Ciravegna, Università di Torino

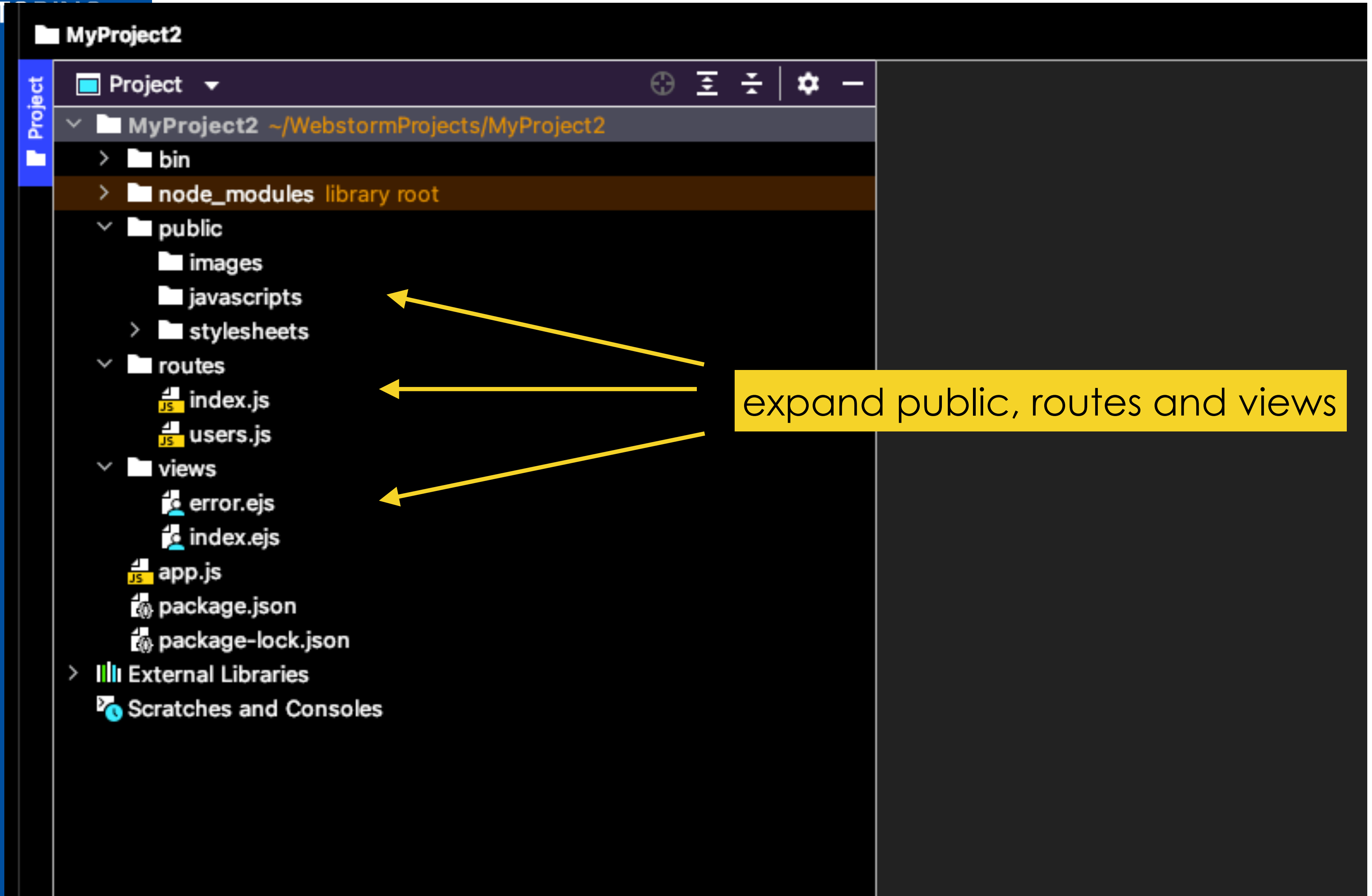
Here is your project

- If you see a missing left panel, click on Project
  - (vertical writing top left)



© Prof. Fabio Ciravegna, Università di Torino

Here is your project (it may take a while)



MyProject2

Project

MyProject2 ~/WebstormProjects/MyProject2
- 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

External Libraries

Scratches and Consoles

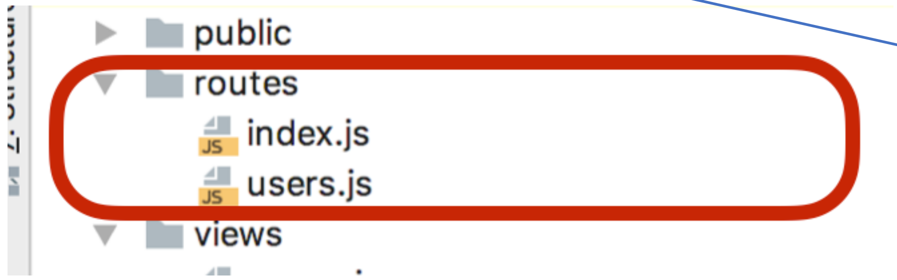
expand public, routes and views

© Prof. Fabio Ciravegna, Università di Torino

In IntelliJ

- in app.js

```
        var users = require('./routes/users');
        …
        app.use('/users', users);
```



- in e.g. routes/users.js

```
/* GET users listing. */
router.get('/', function(req, res, next) {
  res.send('whatever');
});
```

  - `this will respond to http://localhost:3000/users/`

- what are the routes files?
  - the default index.js responds to paths that follow /
  - users.js responds to path that follow the path /users/; that is declared here
  - if you added app.use('/**whatever**', whatever); in app.js
    - and a file called whatever.js under routes
    - all the routes there will respond to /whatever/

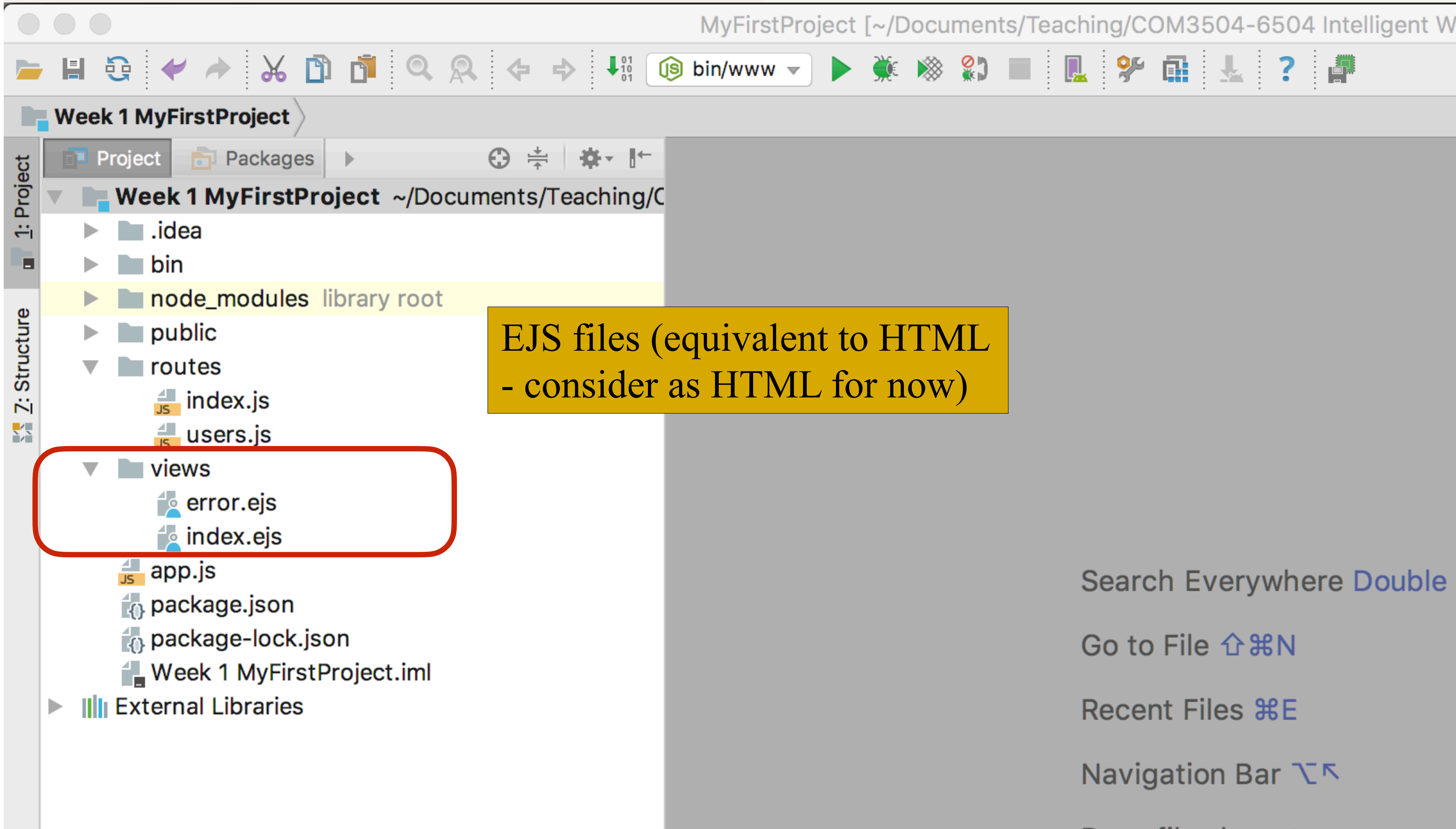
© Prof. Fabio Ciravegna, Università di Torino

UNIVERSITÀ DI TORINO

The client side

© Prof. Fabio Cravegna, Università di Torino

EJS Template files



MyFirstProject [~/Documents/Teaching/COM3504-6504 Intelligent W

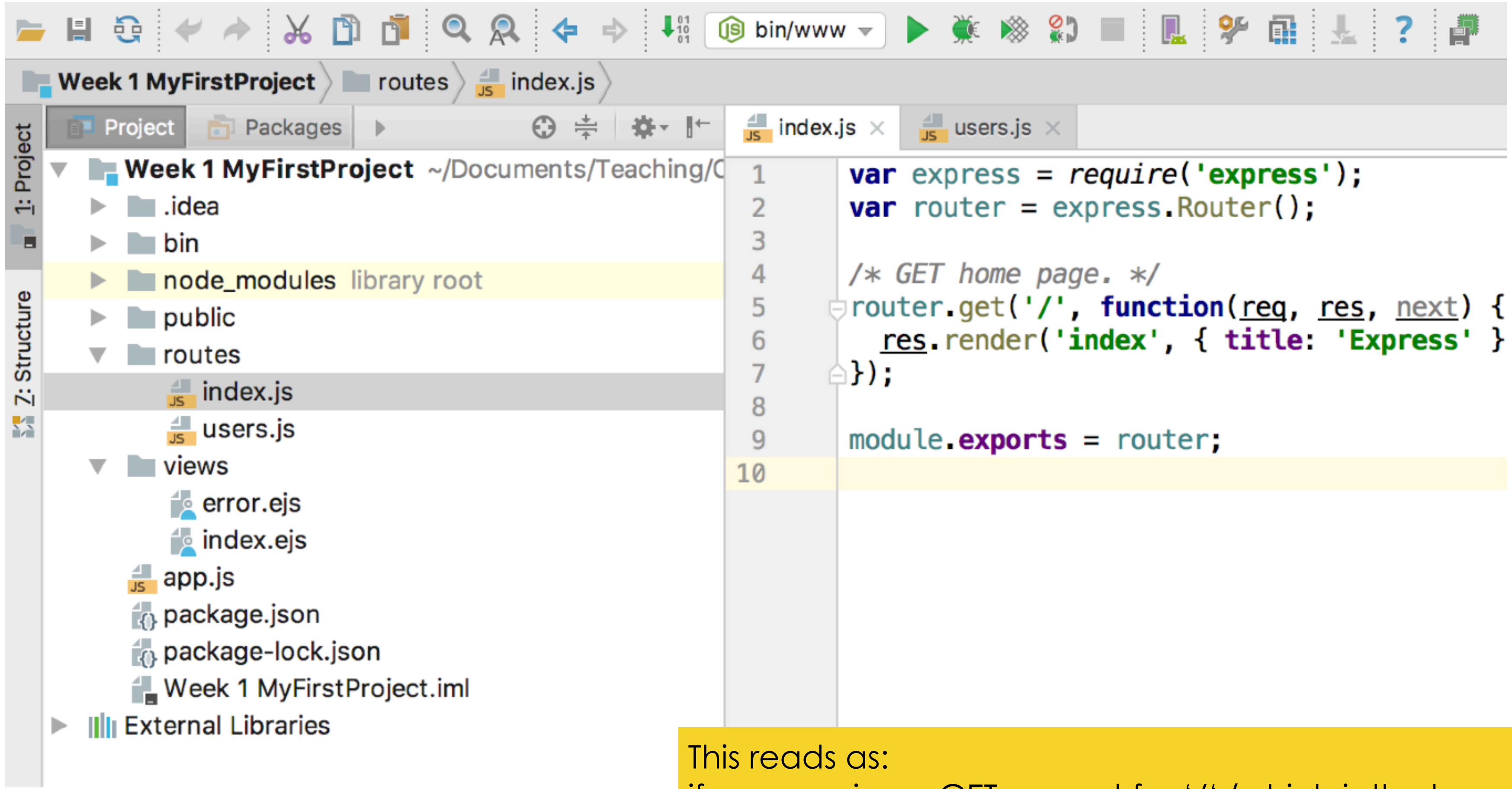
Week 1 MyFirstProject

```
Week 1 MyFirstProject  ~/Documents/Teaching/C
    .idea
    bin
    node_modules  library root
    public
    routes
        index.js
        users.js
    views
        error.ejs
        index.ejs
    app.js
    package.json
    package-lock.json
    Week 1 MyFirstProject.iml
External Libraries
```

EJS files (equivalent to HTML - consider as HTML for now)

UNIVERSITÀ DI TORINO

Serving EJS Template files in routes



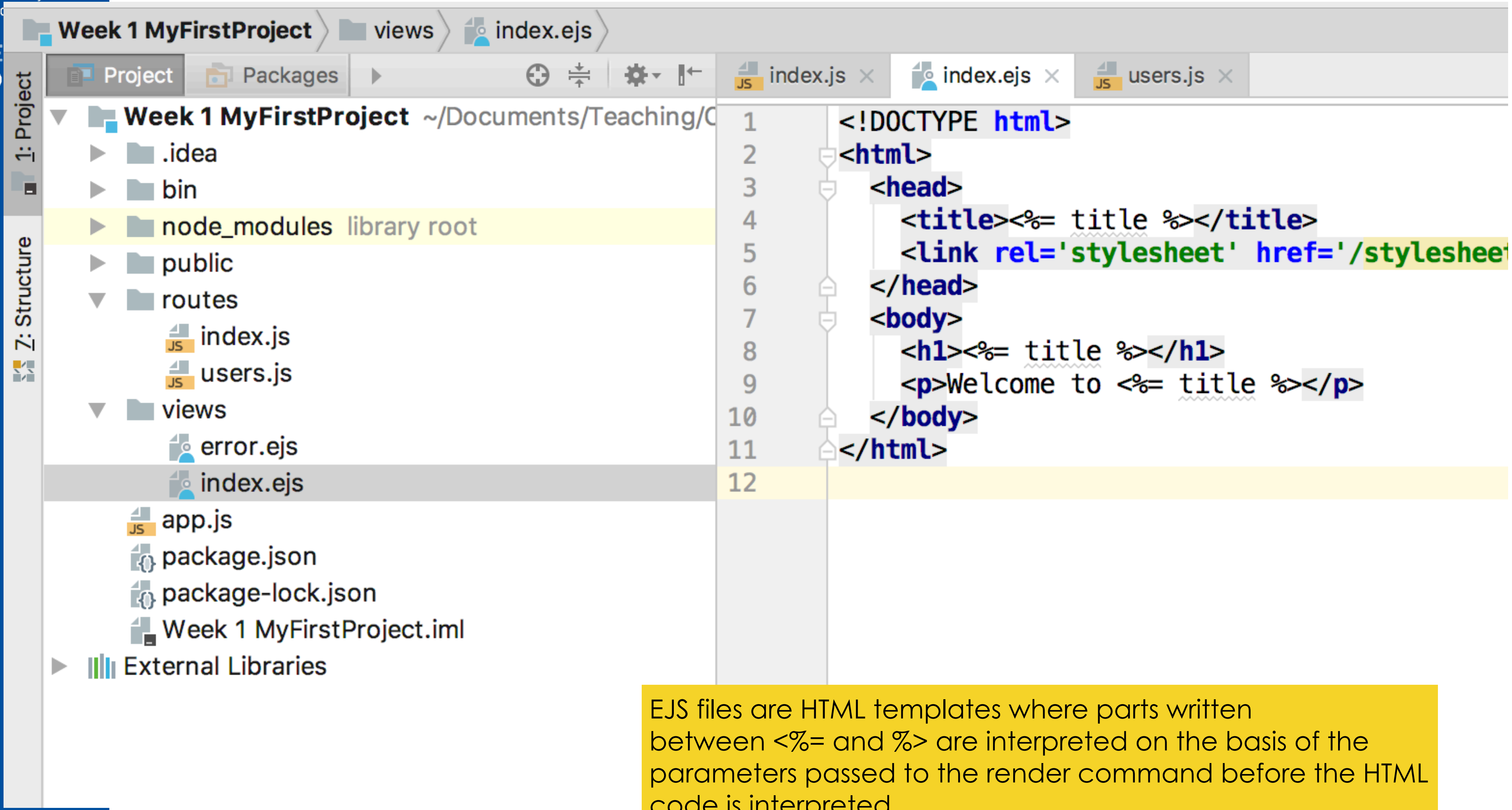
```
Week 1 MyFirstProject ~/Documents/Teaching/C
  .idea
  bin
  node_modules library root
  public
  routes
    index.js
    users.js
  views
    error.ejs
    index.ejs
  app.js
  package.json
  package-lock.json
  Week 1 MyFirstProject.iml
External Libraries
```

```
var express = require('express');
var router = express.Router();

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

module.exports = router;
```

This reads as:
if you receive a GET request for '/' (which is the homepage)
then render the EJS file index which is located in the folder views
the file gets a parameter which is the title. You can find it in the EJS file as
`<h1><%= title %></h1>`

{title: 'Express'}

© Prof. Fabio Ciravegna, Università di Torino



```
Week 1 MyFirstProject ~/Documents/Teaching/C
  .idea
  bin
  node_modules library root
  public
  routes
    index.js
    users.js
  views
    error.ejs
    index.ejs
  app.js
  package.json
  package-lock.json
  Week 1 MyFirstProject.iml
External Libraries
```

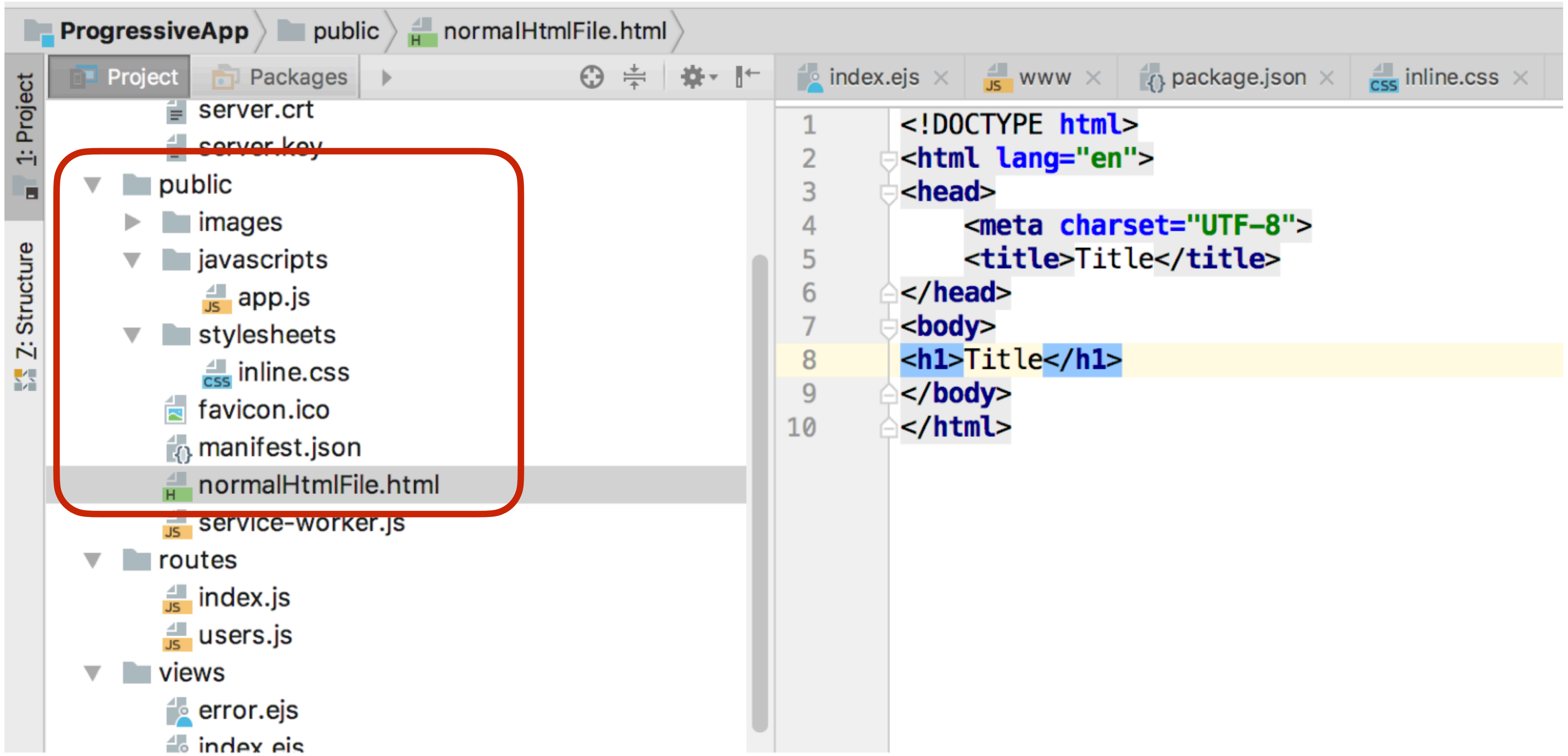
```
<!DOCTYPE html>
<html>
  <head>
    <title><%= title %></title>
    <link rel='stylesheet' href='/styleshee
  </head>
  <body>
    <h1><%= title %></h1>
    <p>Welcome to <%= title %></p>
  </body>
</html>
```

EJS files are HTML templates where parts written between <%= and %> are interpreted on the basis of the parameters passed to the render command before the HTML code is interpreted
(e.g. if the parameter is {title: ‘this is a file’})

© Prof. Fabio Ciravegna, Università di Torino

How to serve static files

- If no special rendering is needed, you can insert HTML files under the public directory



```
server.crt
server.key
▼ public
    ▶ images
    ▼ javascripts
        app.js
    ▼ stylesheets
        inline.css
    favicon.ico
    manifest.json
    normalHtmlFile.html
    service-worker.js
▼ routes
    index.js
    users.js
▼ views
    error.ejs
    index.ejs
```

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Title</title>
</head>
<body>
<h1>Title</h1>
</body>
</html>
```

© Prof. Fabio Ciravegna, Università di Torino

Serving static files

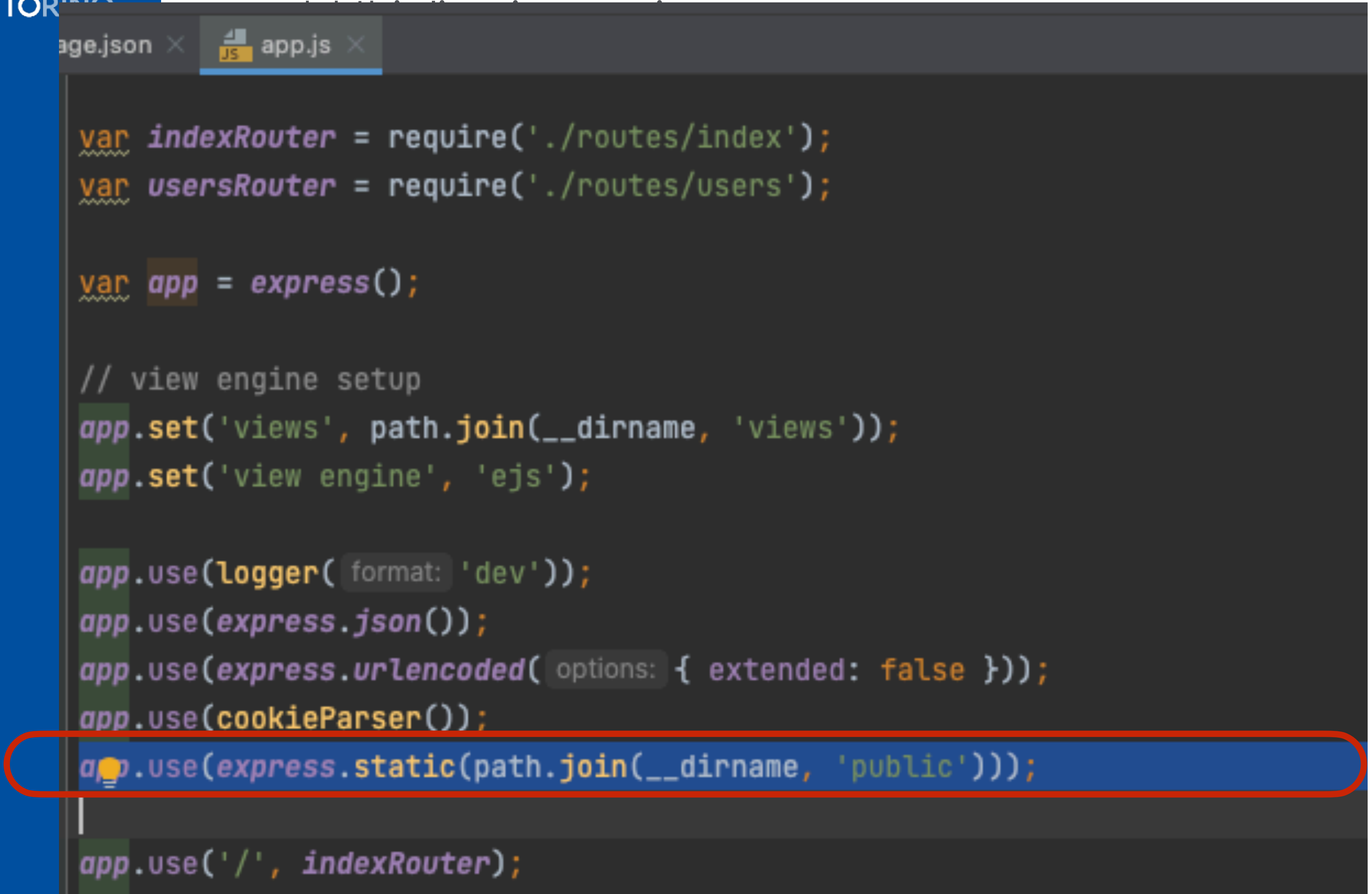
http://expressjs.com/starter/static-files.html

- Serving static files is accomplished with the help of a built-in middleware in Express
  - express.static.
- In app.js just the name of the directory where you keep you static files
- Note: WebStorm does it for you - the official name of the folder is *public*
- If you want to change the name of the folder to public_files (not suggested) insert the following line in app.js

```
app.use(express.static(path.join(__dirname, 'public_files')));
```

© Prof. Fabio Ciravegna, Università di Torino

Where to declare the middleware



```
var indexRouter = require('./routes/index');
var usersRouter = require('./routes/users');

var app = express();

// view engine setup
app.set('views', path.join(__dirname, 'views'));
app.set('view engine', 'ejs');

app.use(logger( format: 'dev'));
app.use(express.json());
app.use(express.urlencoded( options: { extended: false }));
app.use(cookieParser());
app.use(express.static(path.join(__dirname, 'public')));

app.use('/', indexRouter);
```

© Prof. Fabio Ciravegna, Università di Torino

Static Files (ctd)

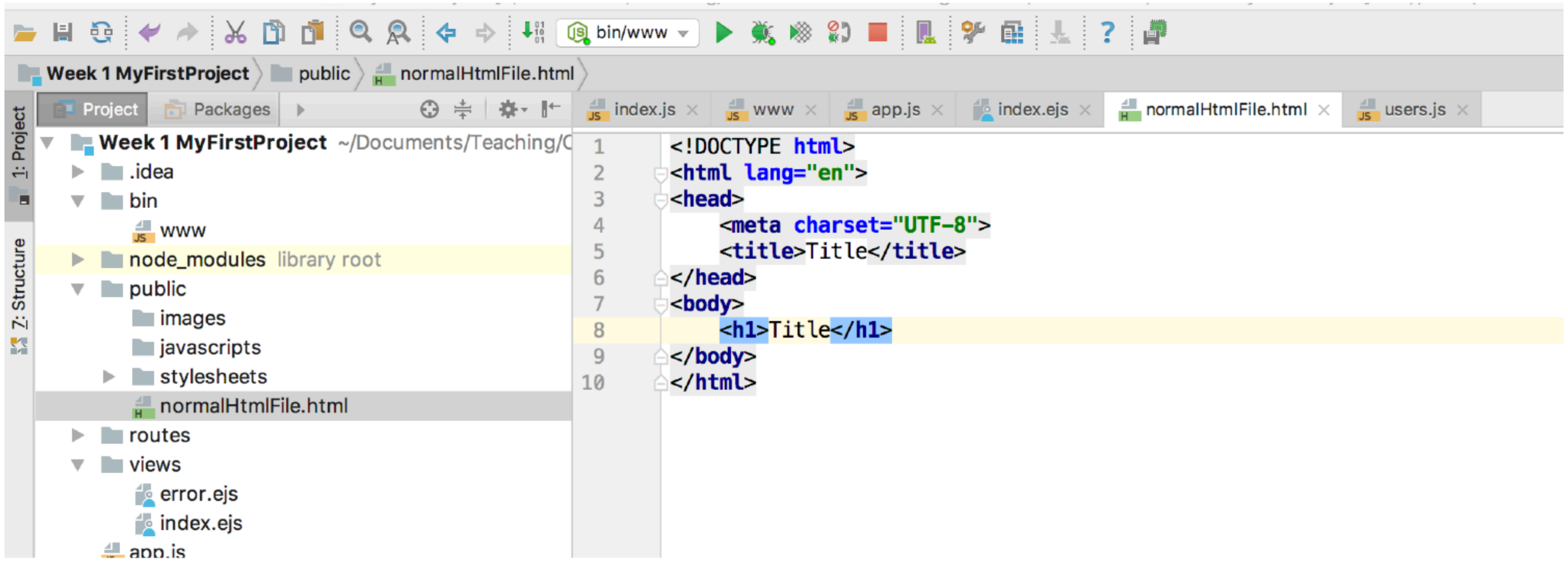
- Now, you will be able to load ALL files under the public directory:

- http://localhost:3000/images/kitten.jpg
- http://localhost:3000/css/style.css
- http://localhost:3000/js/app.js
- http://localhost:3000/images/bg.png
- http://localhost:3000/hello.html

© Prof. Fabio Ciravegna, Università di Torino

Task: create a file under public

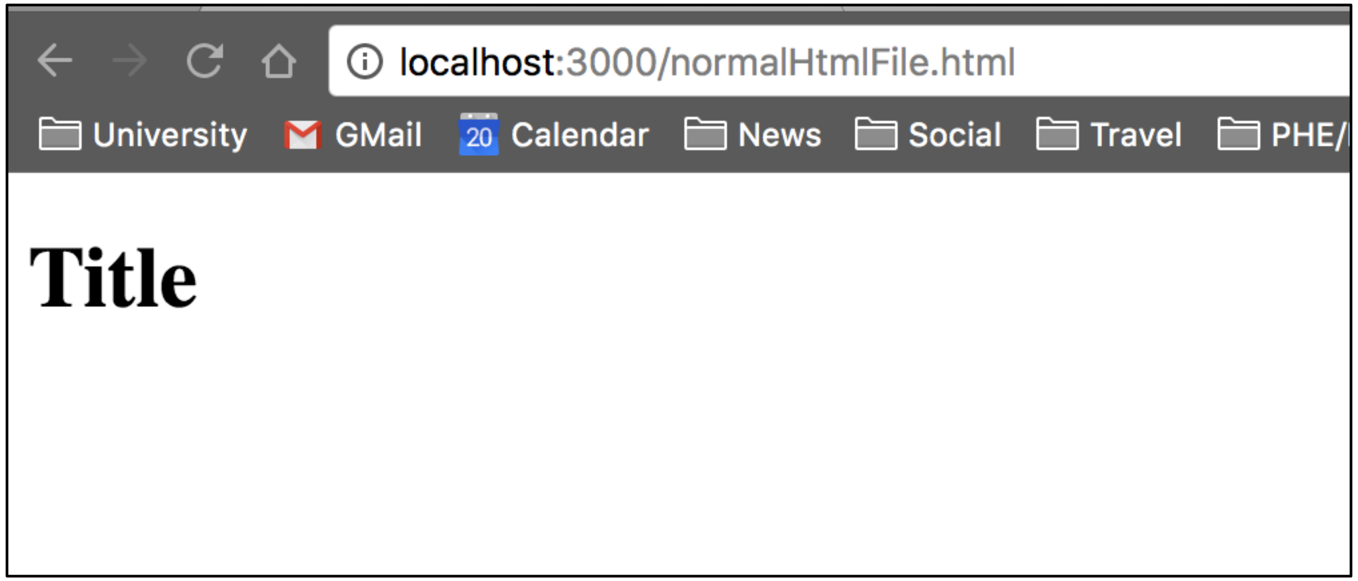
- Right click on the public folder and choose 'new'. Choose new HTML file



```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Title</title>
</head>
<body>
    <h1>Title</h1>
</body>
</html>
```

© Prof. Fabio Ciravegna, Università di Torino

Open **Chrome**: go to http://localhost:3000/normalHtmlFile.html

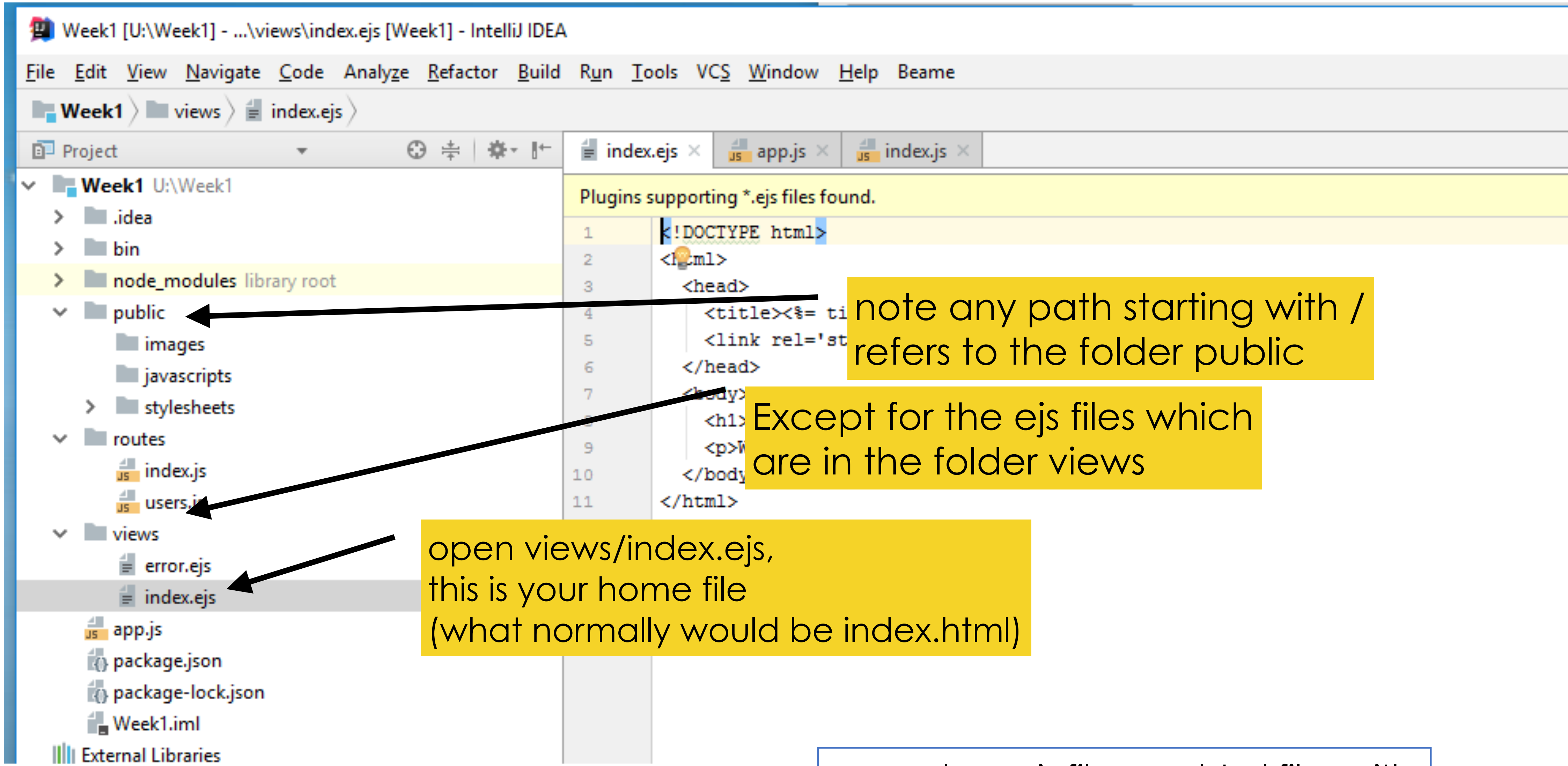


localhost:3000/normalHtmlFile.html

Title

NOTE: FOR THE MODULE YOU ARE **REQUIRED TO USE CHROME**

© Prof. Fabio Ciravegna, Università di Torino



```
<!DOCTYPE html>
<html>
  <head>
    <title><%= ti
    <link rel='st
  </head>
  <body>
    <h1>
    <p>W
  </body
</html>
```

note any path starting with / refers to the folder public

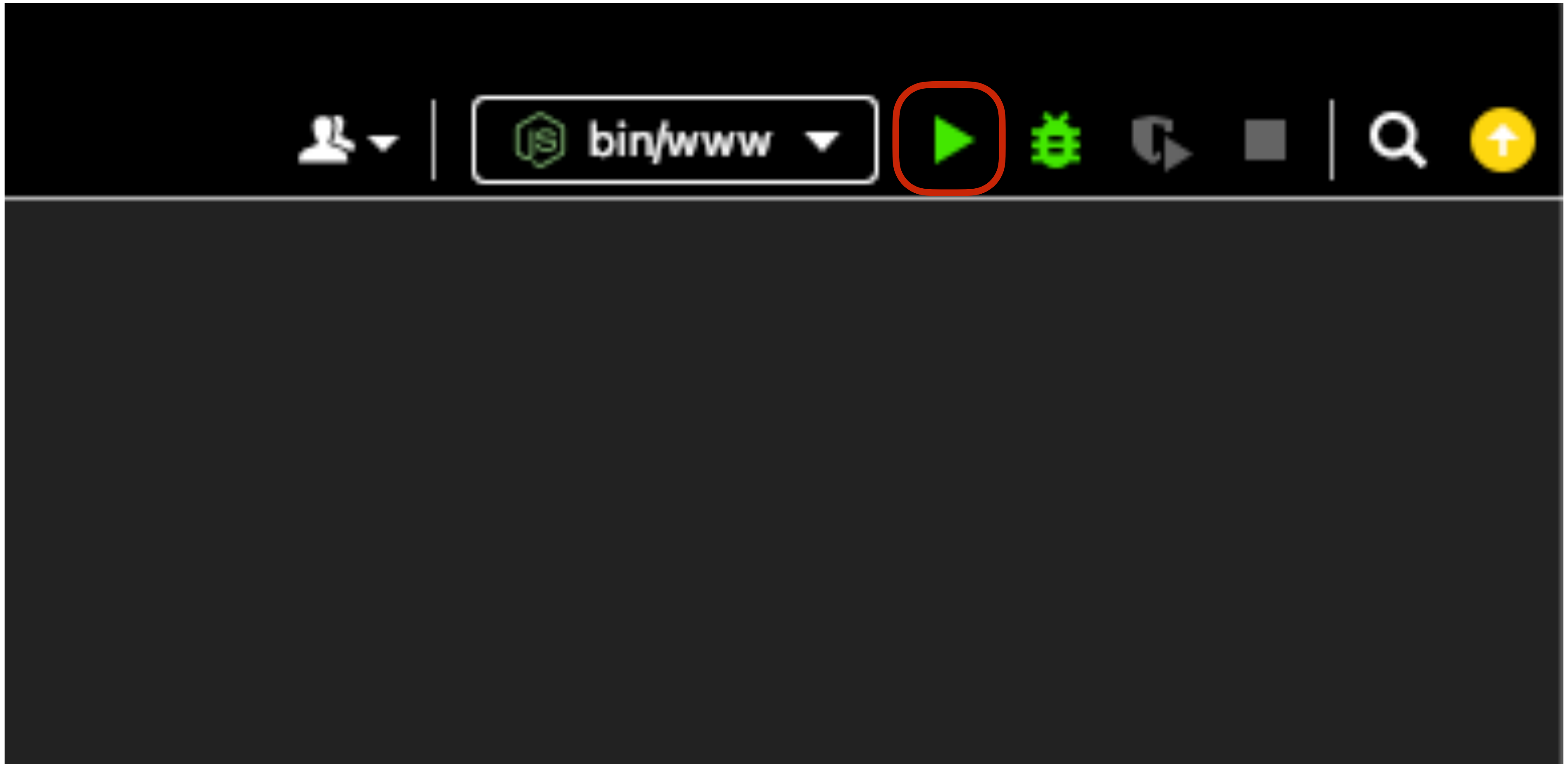
Except for the ejs files which are in the folder views

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!

To run the server

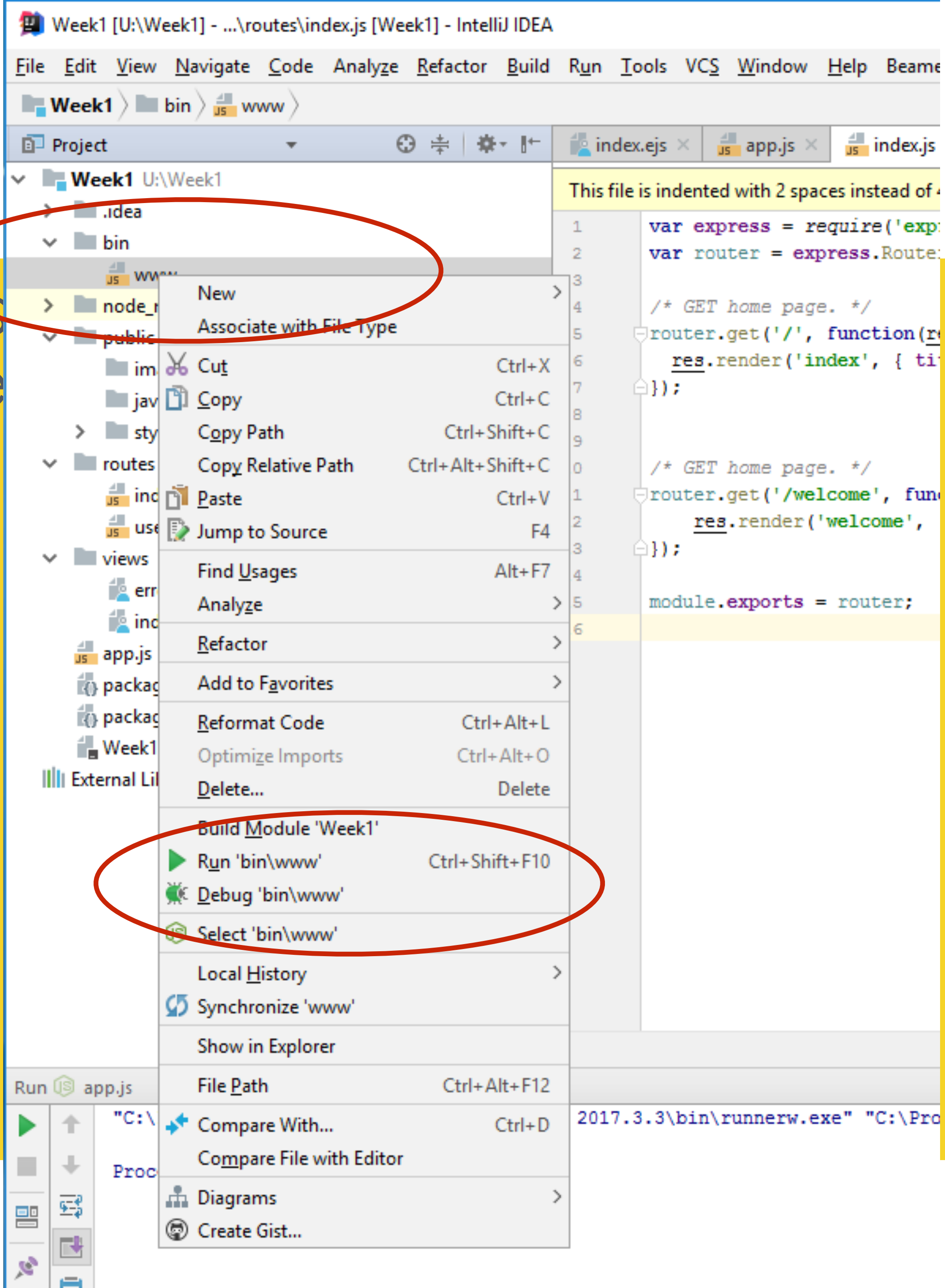
- top right of screen
  - click on the green arrow



© Prof. Fabio Ciravegna, Università di Torino

UNIVERSITÀ DI TORINO

- if the arrow is unselectable
  - Right Click on bin>www
  - Choose Run


Week1 [U:\Week1] - ...\routes\index.js [Week1] - IntelliJ IDEA

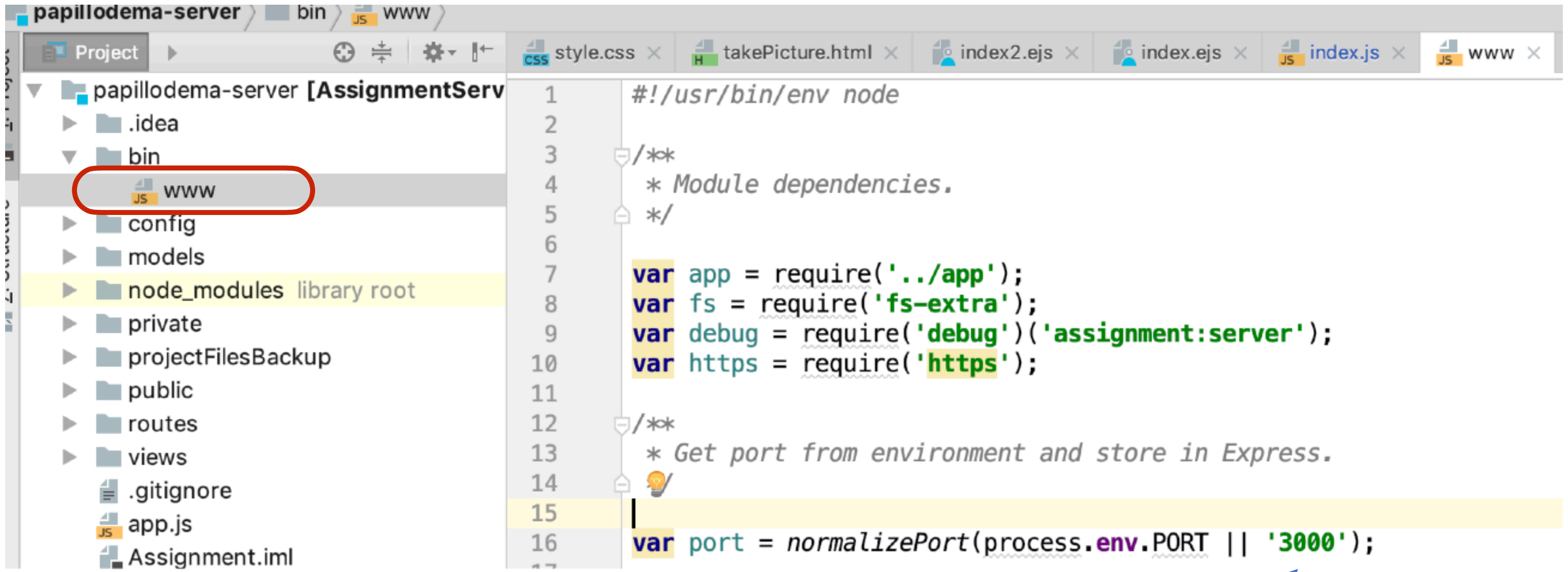
This file is indented with 2 spaces instead of

```
var express = require('exp
var router = express.Route

/* GET home page. */
router.get('/', function(r
  res.render('index', { ti
});

/* GET home page. */
router.get('/welcome', fun
    res.render('welcome',
});

module.exports = router;
```

```
"C:\ 2017.3.3\bin\runnerw.exe" "C:\Pro
Proc
```

© Prof. Fabio Ciravegna, Università di Torino

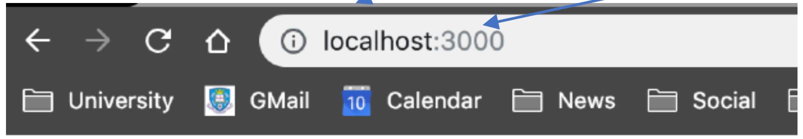
Running the client



```
#!/usr/bin/env node

/**
 * Module dependencies.
 */

var app = require('../app');
var fs = require('fs-extra');
var debug = require('debug')('assignment:server');
var https = require('https');

/**
 * Get port from environment and store in Express.
 */

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

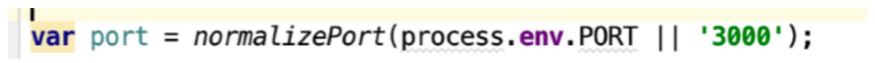
your server is on localhost or 127.1.1

... port



© Prof. Fabio Ciravegna, Università di Torino

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');
```

© Prof. Fabio Ciravegna, Università di Torino

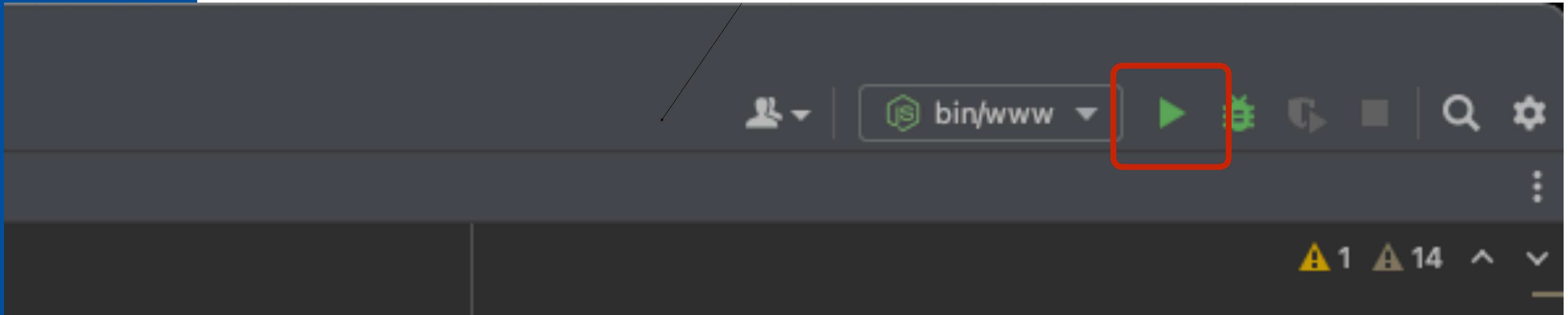
Ports (ctd)

- 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

© Prof. Fabio Ciravegna, Università di Torino

Changes?

- Note: changes to the code have different effects:
  - changes to the **client**
    - i.e. in the Views and Public directories
    - require reloading the page in the browser
  - changes to the **server** (node js)
    - require restarting the server from IntelliJ



© Prof. Fabio Ciravegna, Università di Torino