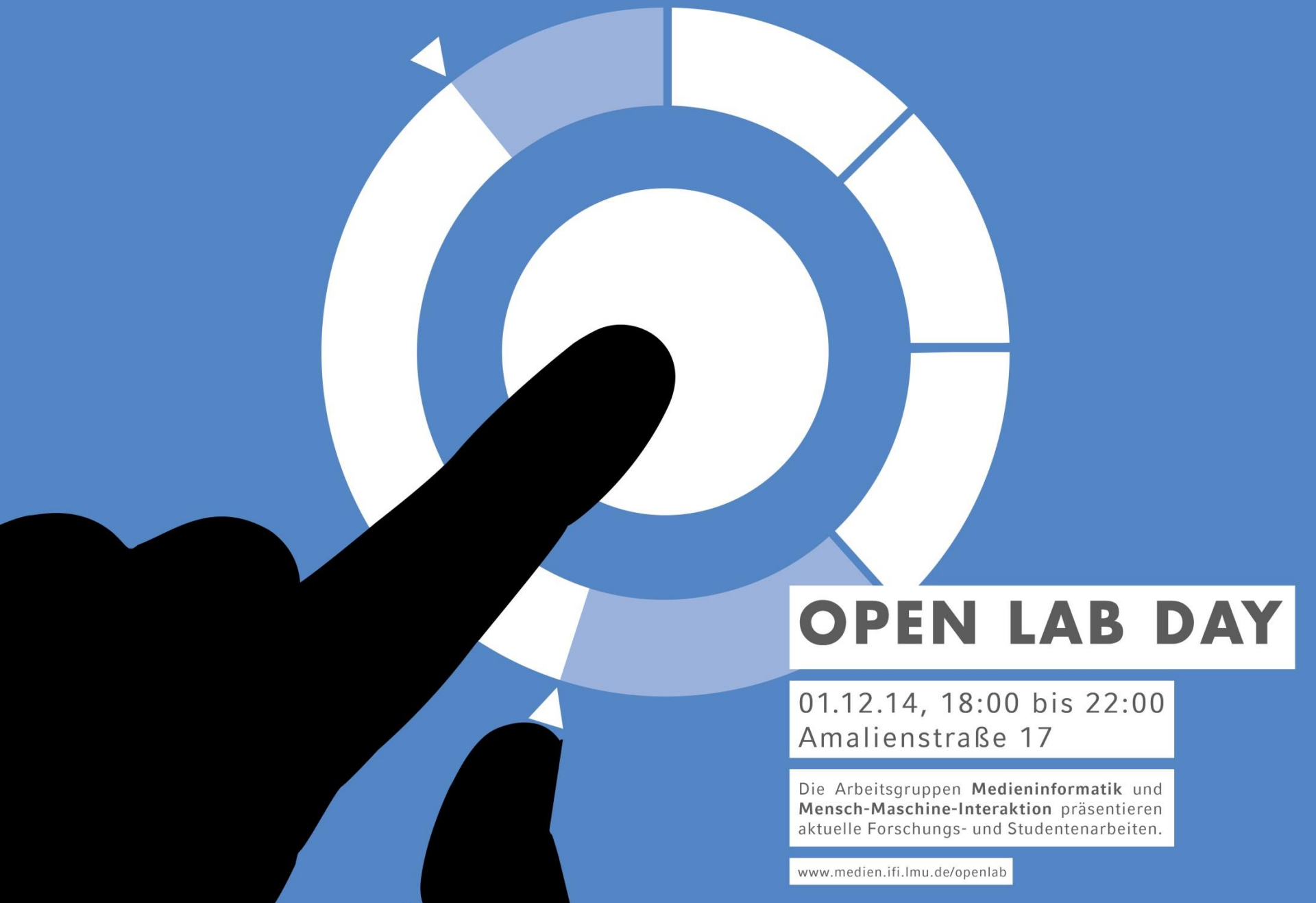
Multimedia im Netz (Online Multimedia)

Wintersemester 2014/15

Übung 07 (Hauptfach)

Today‘s Agenda

- Announcements
- NodeJS Basics
  - Setting up NodeJS
  - Using Node at the CIP Pool
  - Basic middleware functionality
  - Side notes
  - Template Engines: Jade
- Solution: Currency converter.



OPEN LAB DAY

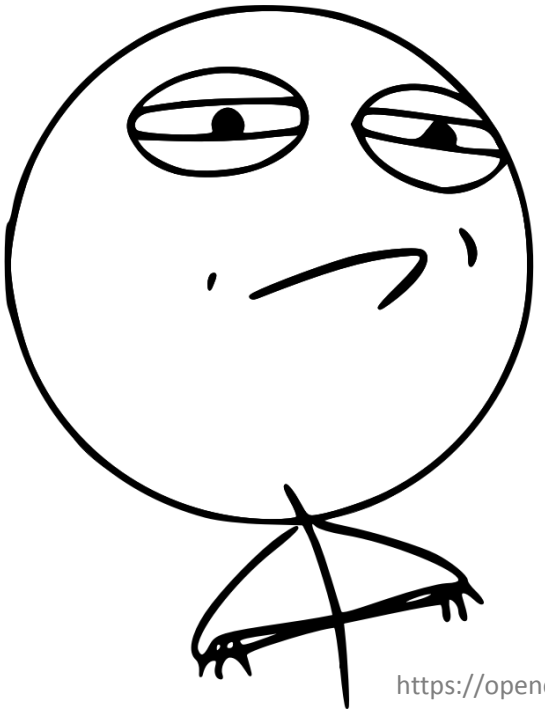
01.12.14, 18:00 bis 22:00
Amalienstraße 17

Die Arbeitsgruppen **Medieninformatik** und **Mensch-Maschine-Interaktion** präsentieren aktuelle Forschungs- und Studentenarbeiten.

www.medien.ifi.lmu.de/openlab

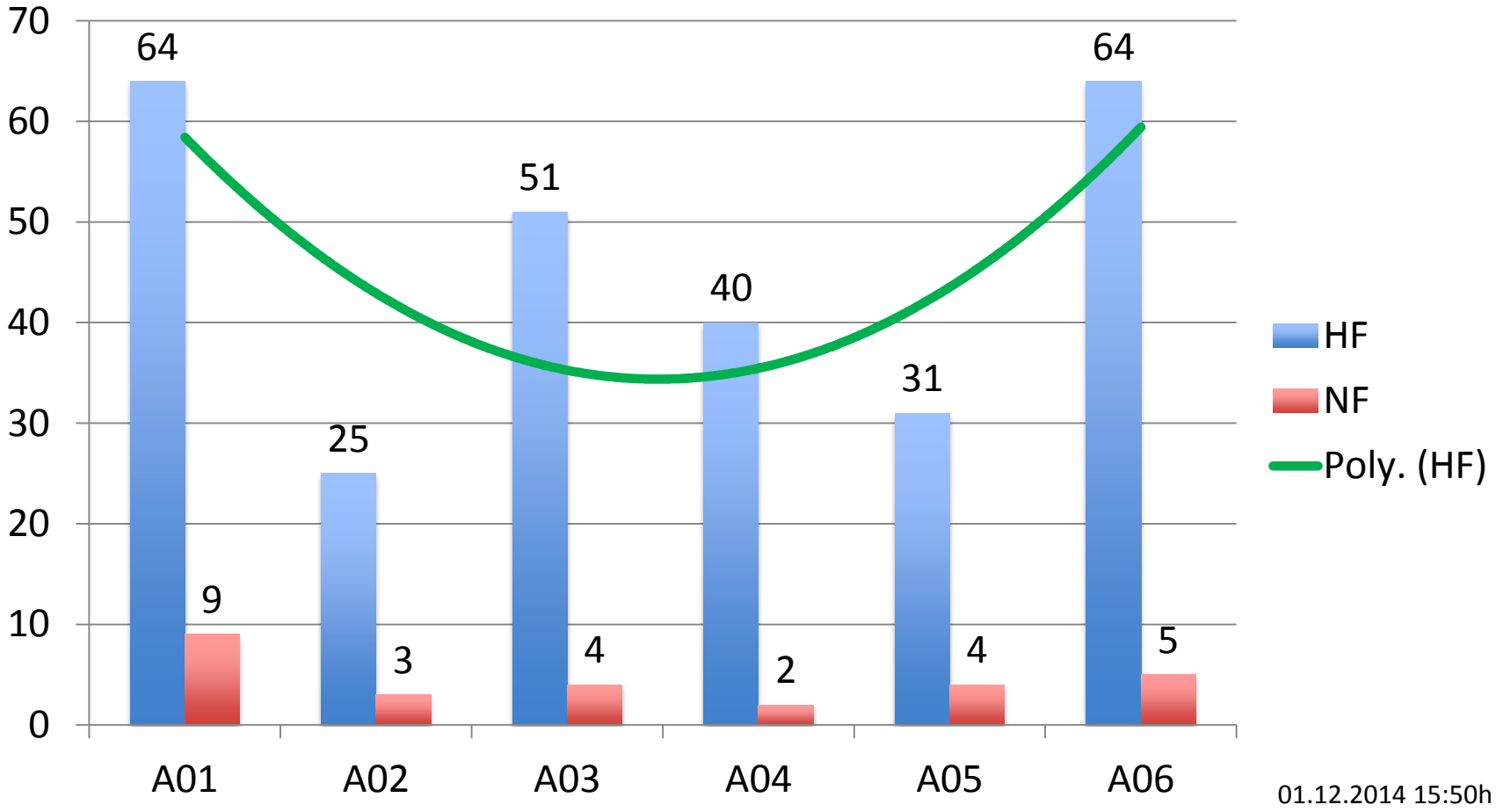
Looking back... The Challenge

- If more than **75** students (HF & NF) submit assignment 06, there will be a short **mock exam** prior to the final exam.



https://openclipart.org/detail/168636/challenge-accepted-by-tavin

Assignment Submissions



01.12.2014 15:50h

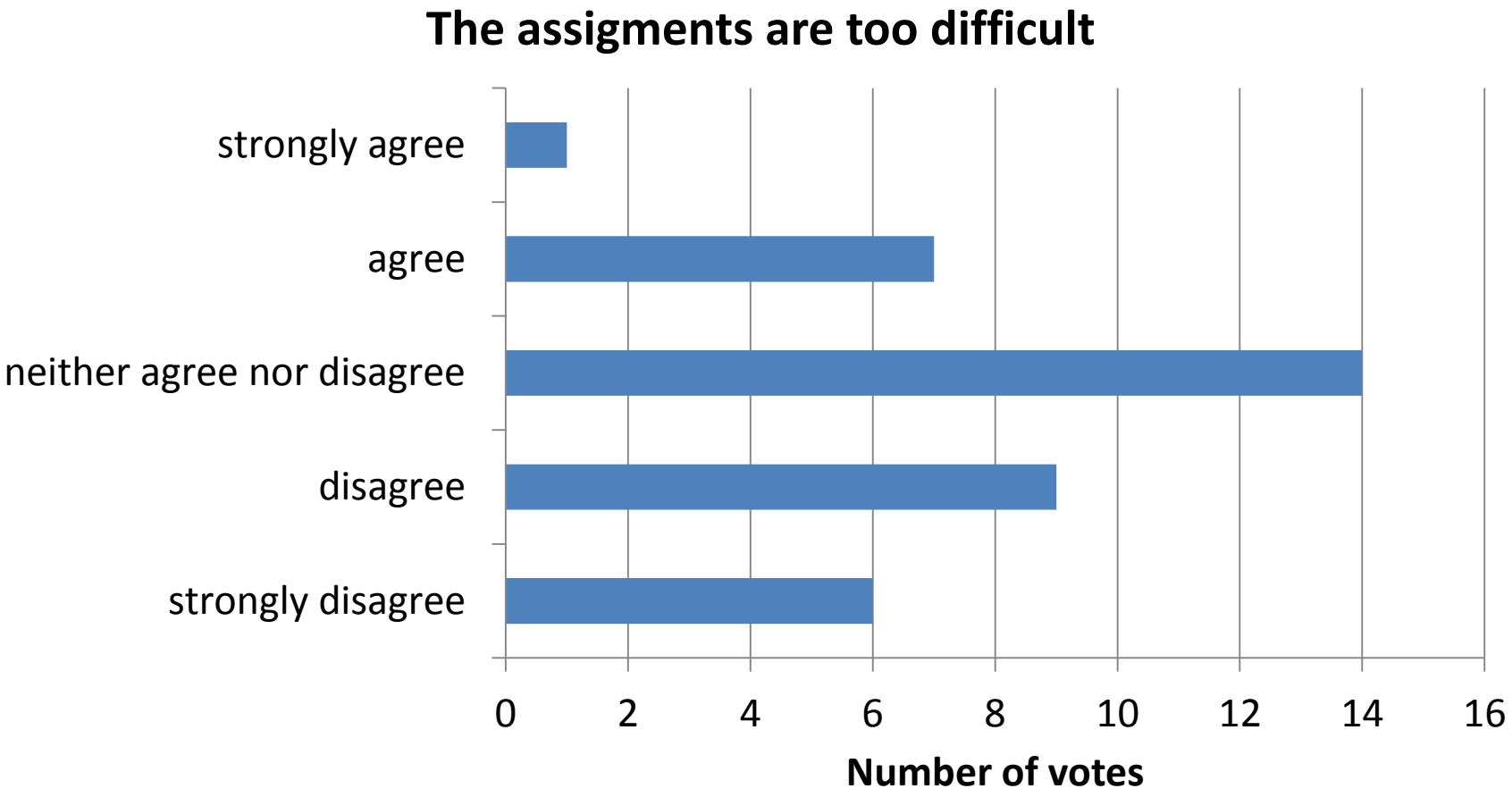
Announcements

- Assignment 06
  - Assignment 06 has been submitted **69** times (HF & NF)
  - Therefore there will be **no written mock exam (n < 75)**
  - However, we will discuss exemplary questions in the tutorials after the Christmas break.

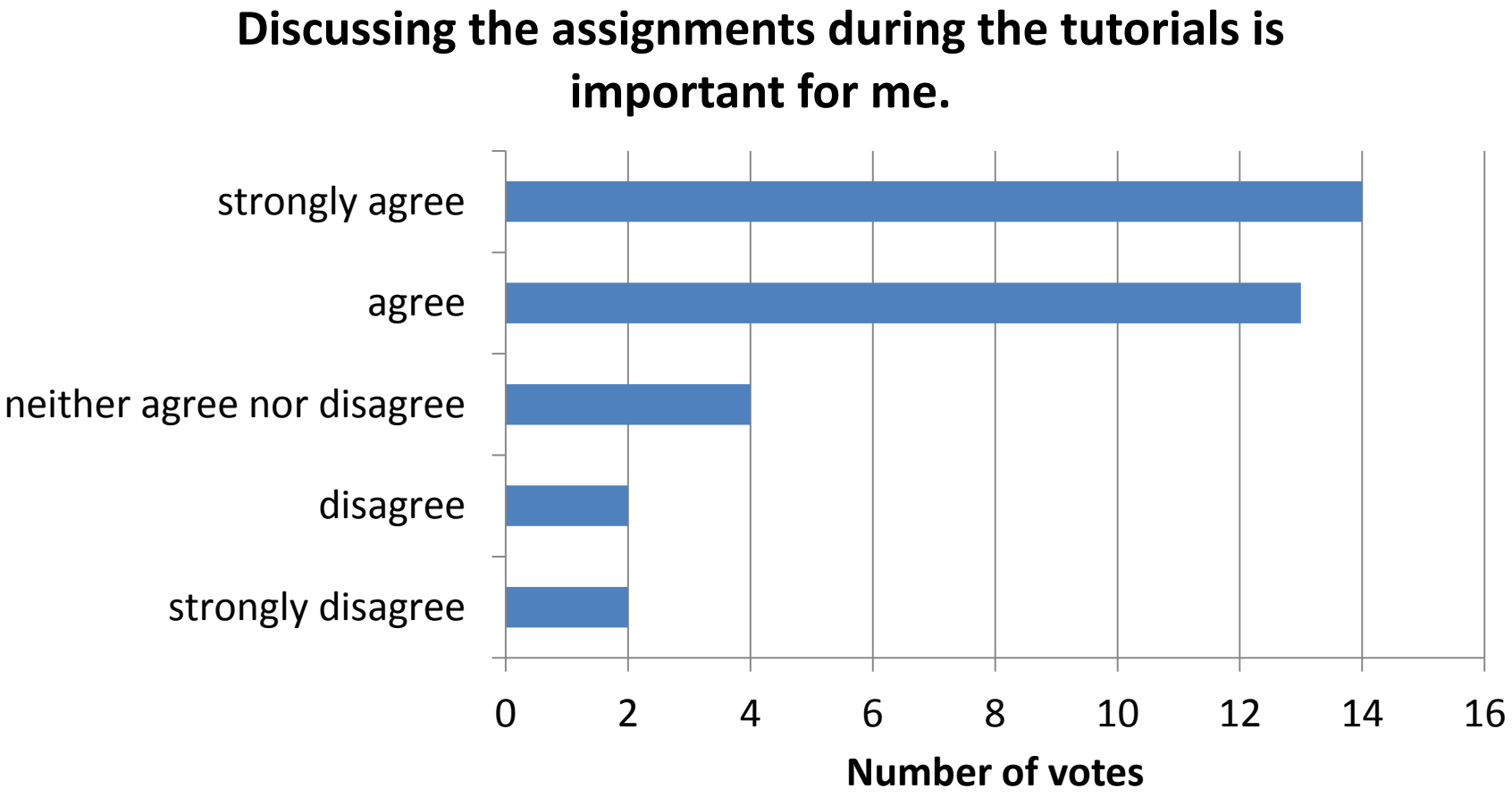
Inofficial Evaluation Results

- Thank you for filling out the form and your valuable feedback!
- Changes:
  - Sample solutions: if you allow us, we‘ll put a sample solution from your submissions on the website for all assignments
  - Corrections will be somewhat more detailed
- Find the results here: https://docs.google.com/forms/d/1q3lw3l59ixq2ZpIYsE8afTavDS1AYNJKIdM6gdlqXgA/viewanalytics

Selected Results: Difficulty



Selected Results: Discussion of Assignments



Node JS – What is it?

- "Node.js® is a platform built on **Chrome's JavaScript runtime** for easily building fast, scalable network applications" (official website, nodejs.org)

- Key features:
  - Asynchronous operations, event driven → callback pattern!
  - Server process included (no Apache is needed)
  - Modularized
  - Only language: JavaScript

Installing Node

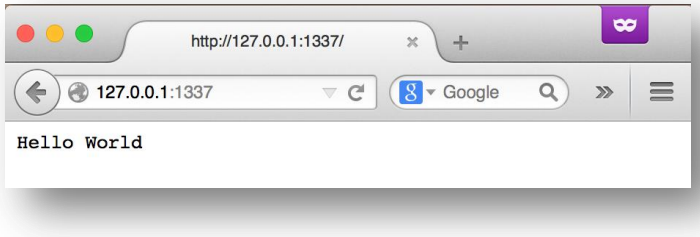
- NodeJS is available for all major platforms
- Download at www.nodejs.org
- Remarks:
  - on Ubuntu you might have to add the NodeJS repository to get a fairly up-to-date version. More info here: https://www.digitalocean.com/community/tutorials/how-to-install-node-js-on-an-ubuntu-14-04-server

Hello World!

```
var http = require('http');
var port = 1337;
var host = '127.0.0.1';

var server = http.createServer(function (request, response) {
    response.writeHead(200, {'Content-Type': 'text/plain'});
    response.end('Hello World\n');
});

server.listen(port, host);
console.log('Server running at http://'+host+':'+port+'/');
```



Node Package Manager

- Easy handling of packages from the command line

- Most important operations:
  - Global package installation
    ```
    npm install -g PACKAGE [PACKAGE2 PACKAGE3 ...]
    ```
  - Local package installation
    ```
    npm install PACKAGE [PACKAGE2 PACKAGE3 ...]
    ```
  - Local package installation & saving to dependencies list:
    ```
    npm install PACKAGE [PACKAGE2 ...] --save
    ```

- Unfortunately, npm is not available at the CIP Pool.

Package script – package.json

```
{
  "name": "TestExpress",
  "version": "0.0.0",
  "private": true,
  "scripts": {
    "start": "node ./bin/www"
  },
  "dependencies": {
    "express": "~4.9.0",
    "body-parser": "~1.8.1",
    "cookie-parser": "~1.3.3",
    "morgan": "~1.3.0",
    "serve-favicon": "~2.1.3",
    "debug": "~2.0.0",
    "jade": "~1.6.0",
    "less-middleware": "1.0.x"
  }
}
```

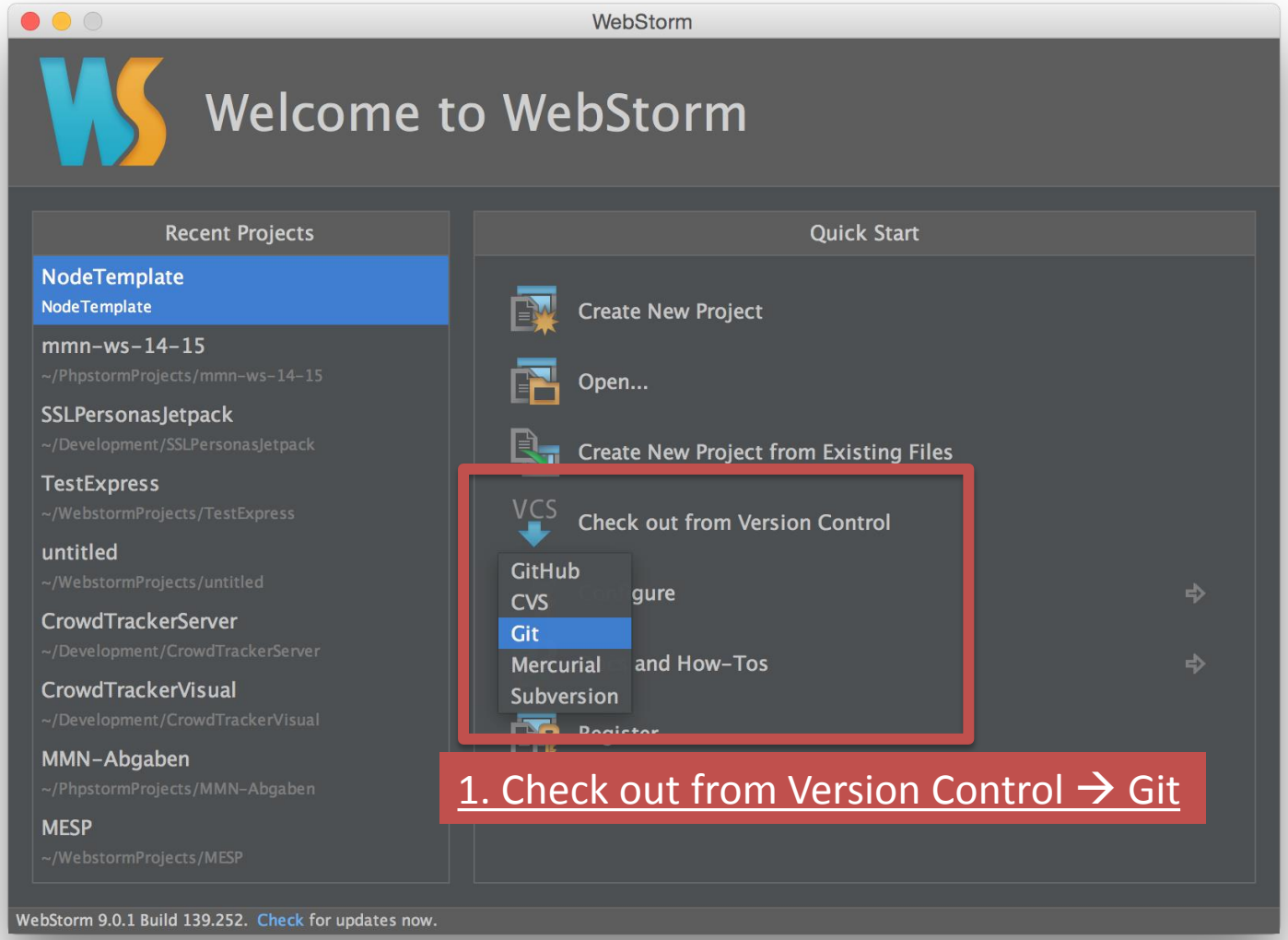
`$ npm install`
installs all dependencies that are listed in the package script into the node_modules directory

Excursus: NodeJS at the CIP Pool

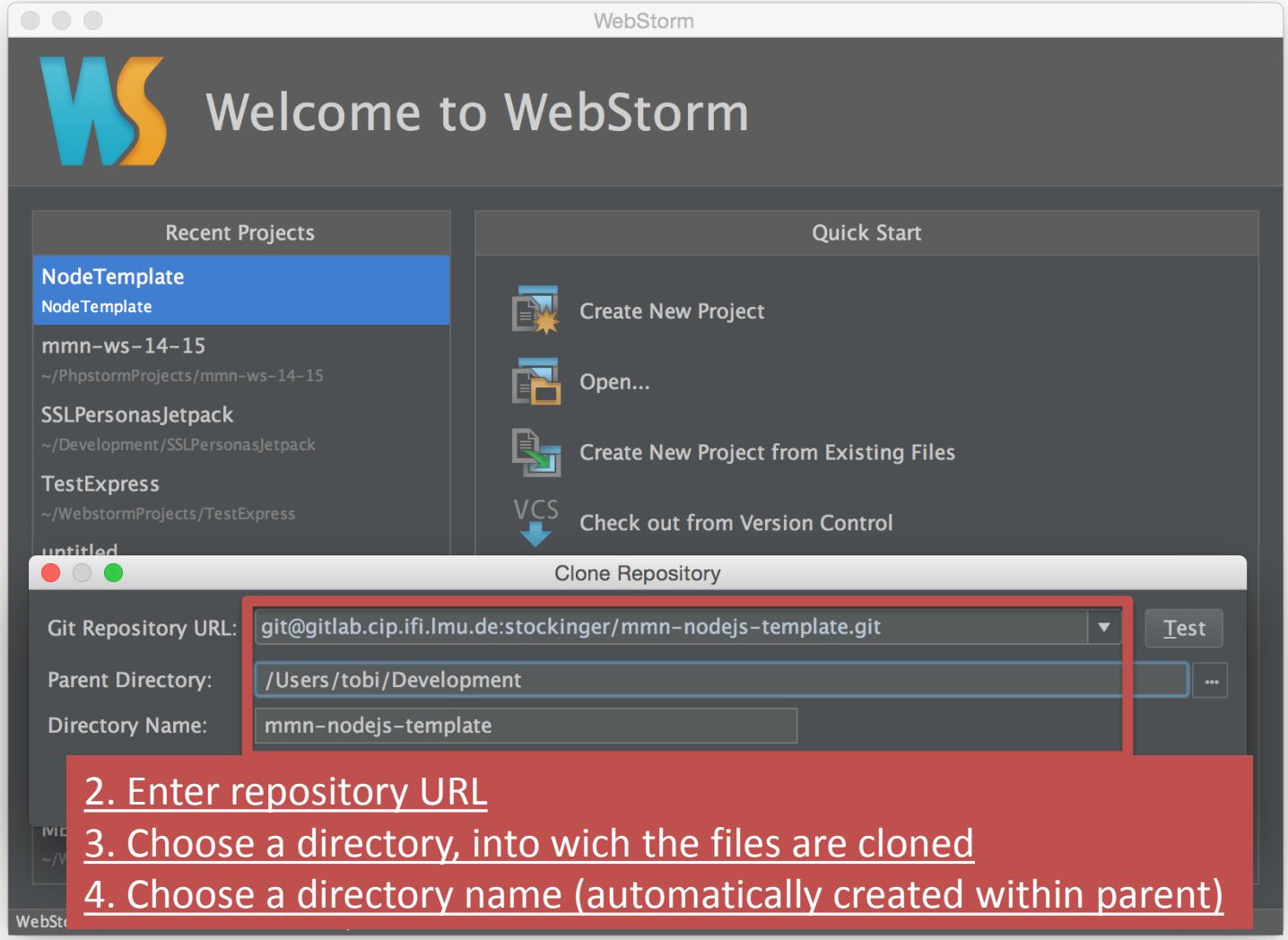
- Unfortunately, npm is not available at the CIP pool at this time
- Solution: use a template project provided from us.
- Option 1 (preferred):
  - Sign up for a GitLab account here: https://tools.rz.ifi.lmu.de/cipconf/index.rb?op=gitlab
  - Generate an SSH Key (if you haven‘t already)
    `$ ssh-keygen`
  - Copy the content of ~/.ssh/id_rsa.pub (SSH Key) to your GitLab profile: https://gitlab.cip.ifi.lmu.de/profile/keys
  - Clone the template to your computer:

```
$ mkdir assignment07 && cd $_
$ git clone git@gitlab.cip.ifi.lmu.de:stockinger/mmn-nodejs-template.git
```

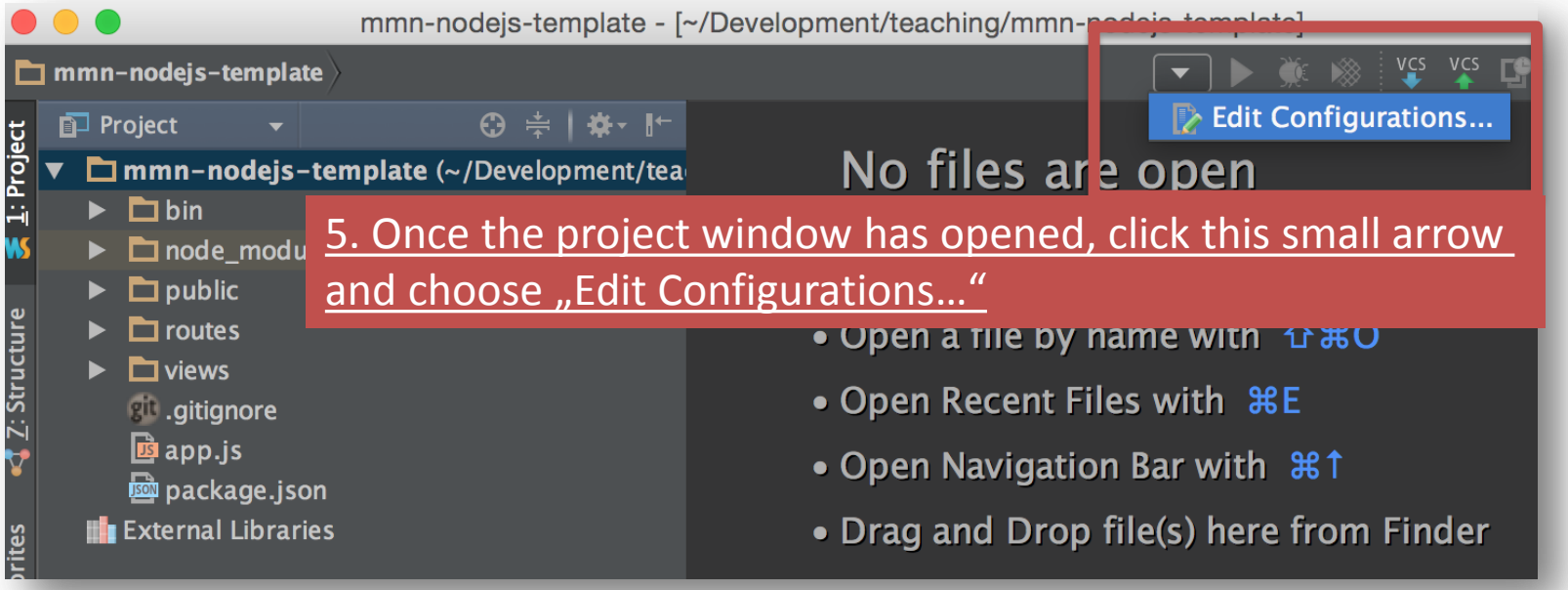
Step-by-Step: WebStorm Project Setup (1)



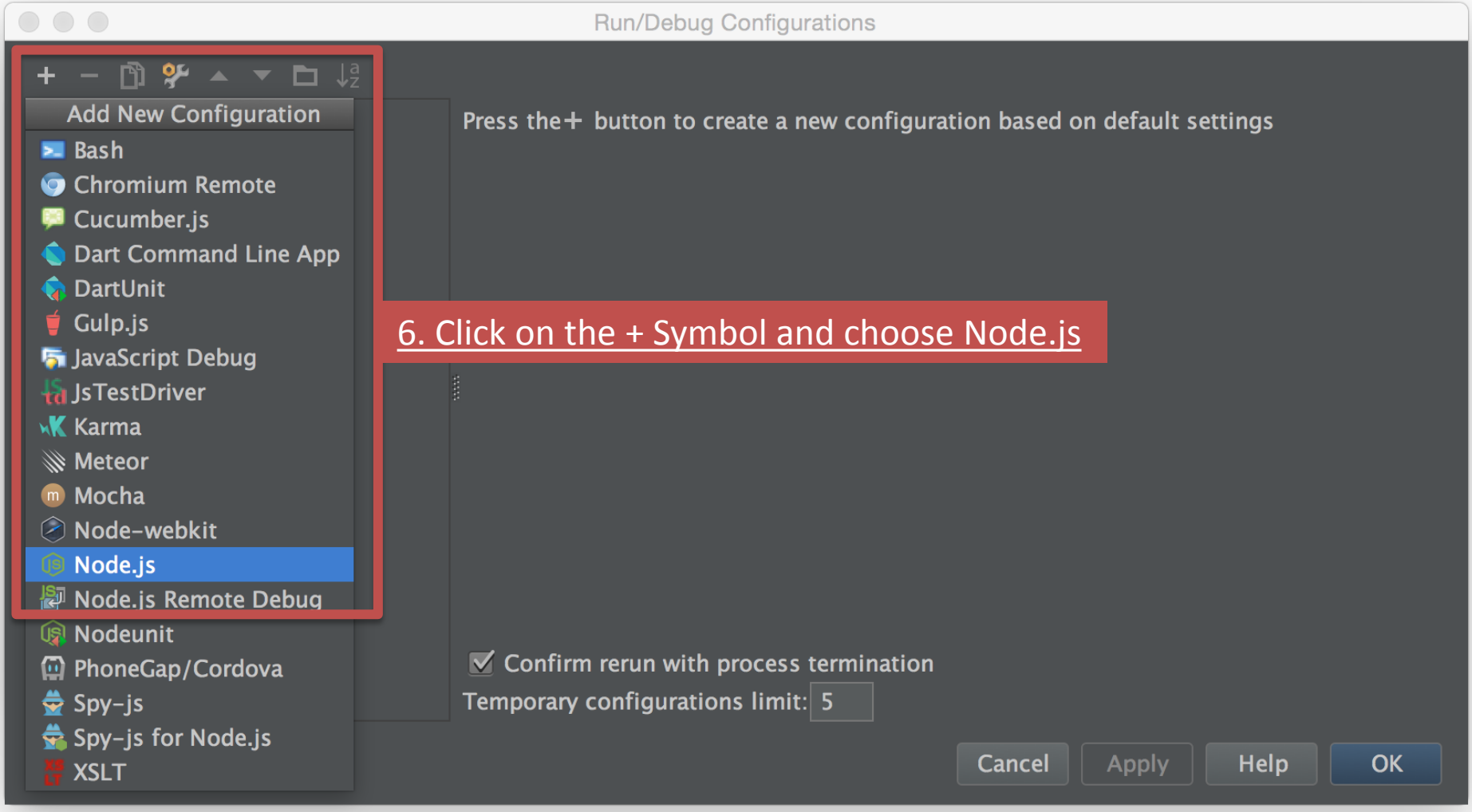
Step-by-Step: WebStorm Project Setup (2)



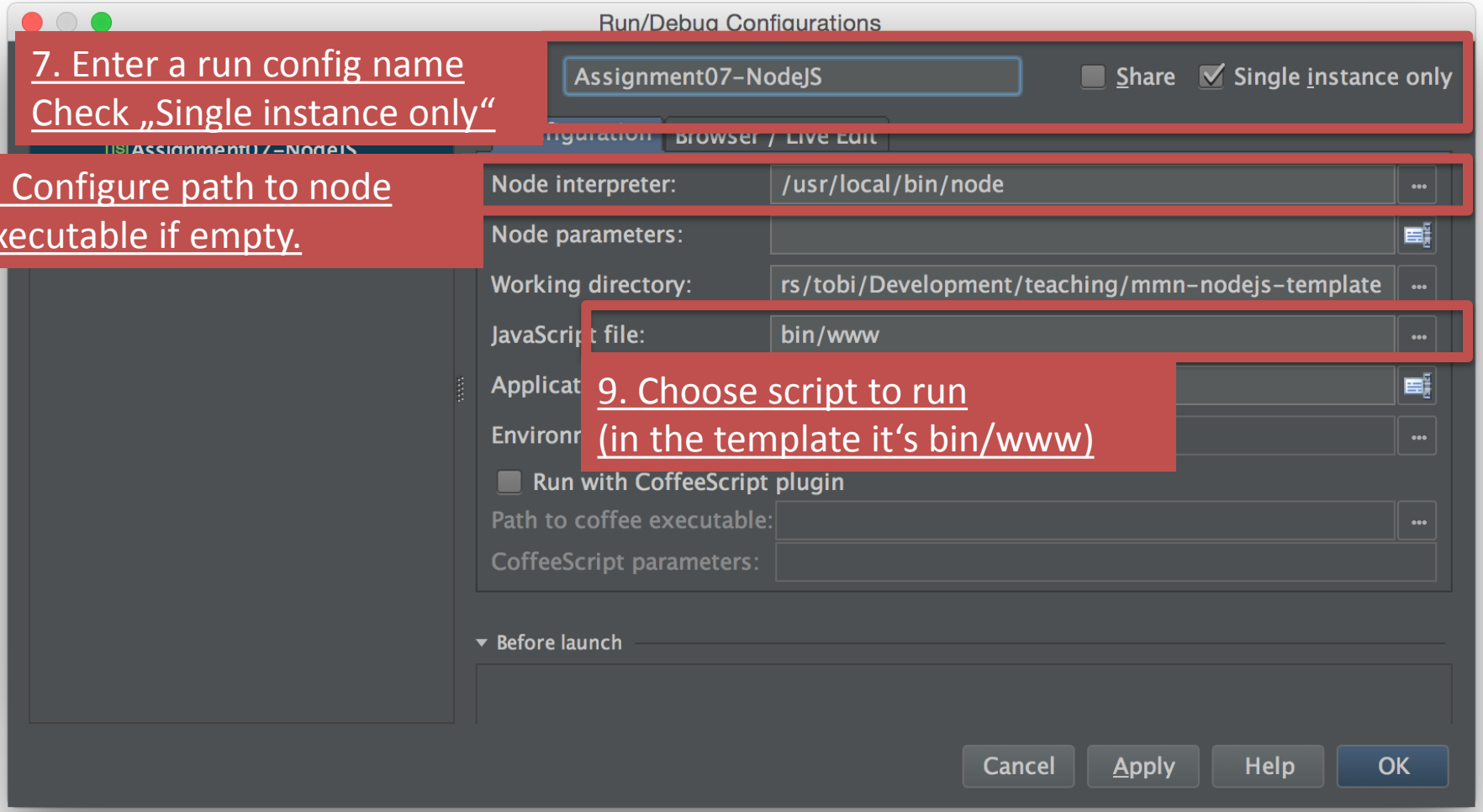
Step-by-Step: WebStorm Project Setup (3)



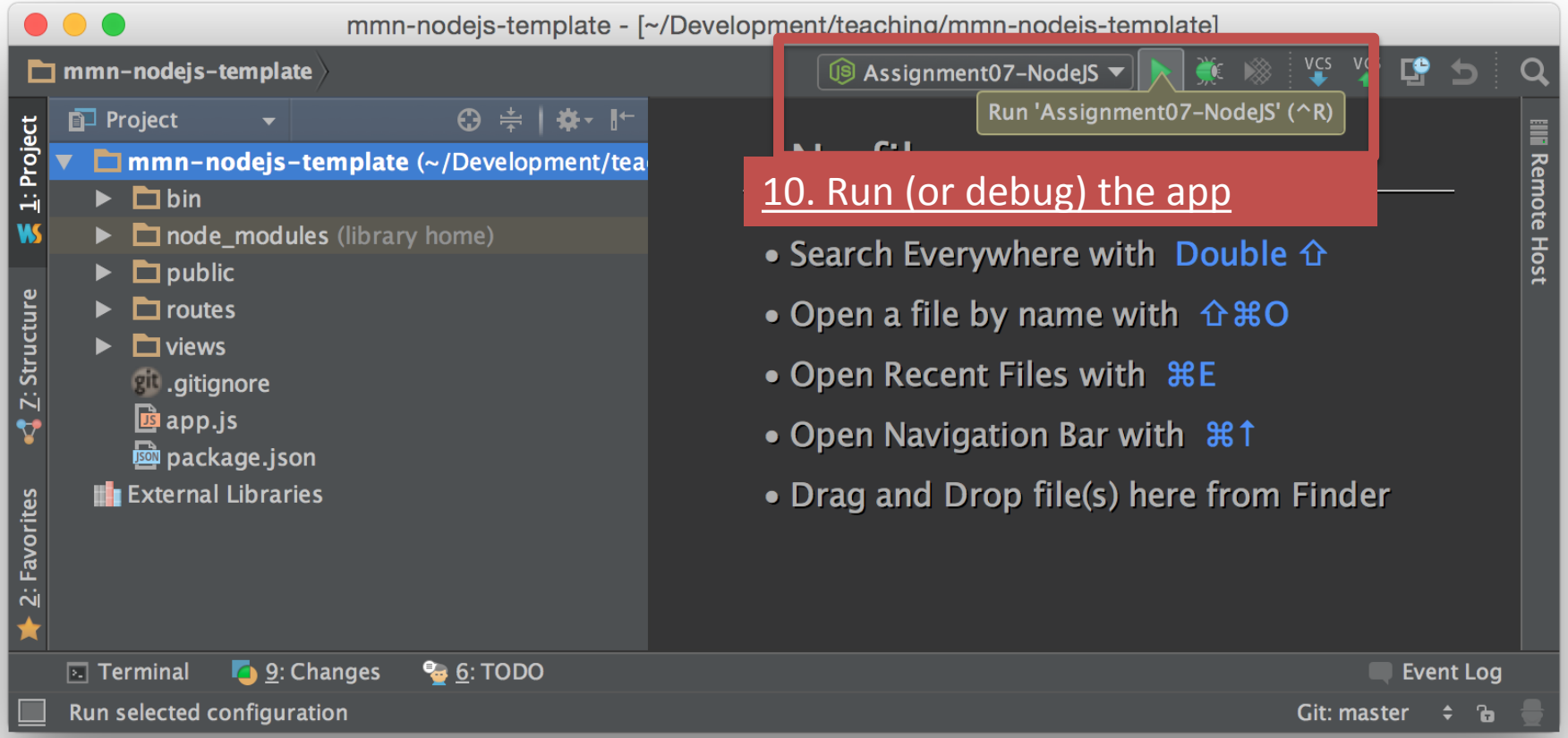
Step-by-Step: WebStorm Project Setup (4)



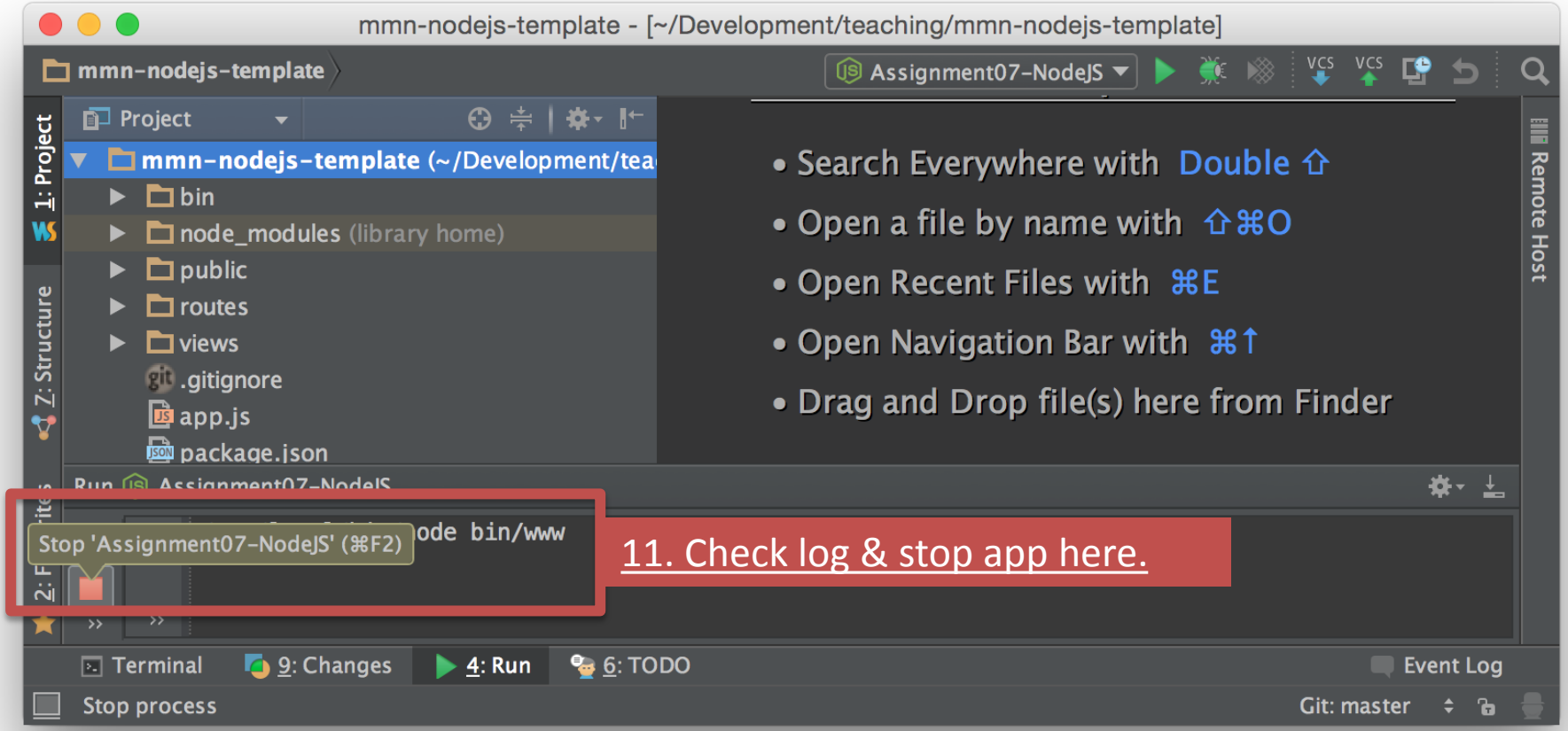
Step-by-Step: WebStorm Project Setup (5)



Step-by-Step: WebStorm Project Setup (6)



Step-by-Step: WebStorm Project Setup (7)



Starting a node app from the Command Line

- On your own machine:
  `$ node <path_to_app>`

- On a CIP pool computer:
  `$ nodejs <path_to_app>`

- Generate an express app (alternatives)
  1. Create a new WebStorm project with NodeJS Express template
  2. install the Express generator:
     $ npm install –g express-generator
     $ express myApp

Express – a web application framework

- One of the most popular NodeJS frameworks.
- Characteristics:
  - minimalistic
  - easy to use API
  - many utility methods and middleware functionalities
  - thin layer on top of NodeJS
- Side notes:
  - responsible for the letter E in the MEAN stack

- Find the documentation here: http://expressjs.com/

Basic Express App

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

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

var server = app.listen(3000, function () {
    var host = server.address().address;
    var port = server.address().port;
    console.log('app listening at http://%s:%s',
        host, port)
});
```

Middleware & Routes

- Terminology:
  - Routes ≈ mount paths
  - At the end of routes: „end points“
- We can define more than one middleware for a route!
  (see example **`03_middlewareChain.js`**)
- Middleware delivers static or dynamic content
  (e.g. files in a directory or REST API call)
- Middleware is in the “middle” between the request and the response.
- Using a middleware function in express:

```
app.use(['path'], [function...],
function)
```

Router (Express > 4.0)

- Lightweight express app only for routing
- Obtaining a router sub-app:
  ```
  var router = express.Router();
  ```
- `.get` and `.post` routes posses different request objects
  - `req.params` inside a GET request
  - `req.body` inside a POST request
- Tutorials:
  - http://scotch.io/tutorials/javascript/learn-to-use-the-new-router-in-expressjs-4 (you can use the template and insert the code, if you are at the CIP Pool)
  - https://www.packtpub.com/books/content/understanding-express-routes

Router Example

04_routerExample.js

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

router.get('/',function(req,res){
   res.send('Homepage');
});
router.get('/about',function(req,res){
   res.send('About Page');
});

app.use(router); // or: app.use('/',router);

var server = app.listen(3000, function () {
    var host = server.address().address;
    var port = server.address().port;
    console.log('Example app listening at http://%s:%s',
        host, port)
});
```

Extending the template: New route

- Task (5-10 Minutes):
  - add a new route that supports the POST method
  - send a response:
    - JSON object
    - includes all key/value pairs from the request.body object
    - to access the request.body object, make sure to require the body-parser module.
  - test your route with a REST client browser extension, e.g. Postman

Basic routes with JSON response

stuff.js

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

/* GET home page. */
router.get('/', function(req, res) {
    res.json({
        hello : 'world'
    })
});

module.exports = router;
```

app.js

```
var index = require('./routes/index');
var stuff = require('./routes/stuff');
var app = express();
app.use('/', index);
app.use('/stuff', stuff);
```

Common Problems

- Error: listen EADDRINUSE
  - usually means that you tried to listen on a port that‘s already used
  - solution: quit any running instances of nodejs and restart
- Express dependencies are not resolved
  - npm install express -g

PHP & NodeJS – advantage comparison

PHP	NodeJS
Widely distributed, large community	Fast, performant execution
Finding a web hoster is easy	Variable access (same variables for multiple requests possible)
PHP code can be embedded into HTML	Less memory intensive (arguably)
No restart after code modifications	Shared JS code on Client and Server
Easier for less dynamic websites	Suitable for websites with many database queries
Easy to learn for beginners	Traffic management for large websites
	WebSockets allow the subscriber pattern

http://nodecode.de/php-oder-nodejs

Template Engines

- Render templates and values to finished web pages.
- Examples:
  - Jade
  - Handlebars
  - Hogan
  - ... many more
- Usage with Express:
  - in the app module:

```
app.set('views', path.join(__dirname, 'views'));
app.set('view engine', 'jade');
```

  - in the route handlers:

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

Jade – Basic Layout

index.jade

```
extends layout

block content
  h1= title
  p Welcome to #{title}
```

layout.jade

```
doctype html
html
  head
    title= title
    link(rel='stylesheet', href='/stylesheets/style.css')
  body
    block content
```

Rendering the template

- Providing values:
  - pass a JS object / JSON
  - access the values from the jade template
- Note: we can easily switch the template engine and this still works!

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

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

module.exports = router;
```

Extending the template: New template

- Task (5 minutes):
  - add a new template file
  - elements:
    - dynamic heading (h1) content
    - div with class container
    - dynamic content inside the container
  - new route /templateTest
    - render the template
    - pass random parameters for the heading and content

Jade: Further details

- Javascript code can be escaped with a leading dash (-), e.g.

```
-    for(var i=0;i<someCollection.length;i++){
-    var element = someCollection[i];
-    console.log(element);
     div(class='element')
          div(class='text')= element.text
          div(class='number')= element.someNumber
- }
```

- Side notes:
  - there are file watchers in WebStorm that allow generating the HTML files for preview on the fly
  - You can convert your existing HTML files to Jade using this web app: http://html2jade.aaron-powell.com/

CSS Preprocessors

- Problem:
  - CSS files become messy and unreadable
  - CSS files don‘t match the HTML file‘s structure
- Solution:
  - CSS Preprocessors. Most famous examples are SASS & LESS
  - Key features:
    - Variables and Operators
    - Nesting
    - Mixins
- Make sure to add file watchers to the SASS or LESS files.

Assignment 7

- **Topic: Feedback voting app**
- **Due in:** 2 Weeks
- **Due date:** 15.12.2014
- **Please note**: There will be another assignment next week!

JS Code Retreat Munich

- Code all day long on your project with other JS enthusiats
- http://www.meetup.com/JavaScript-CodeRetreat/events/209200652/
- Thanks to V. Böhner for pointing out this event

Thanks!

What are your questions?