



UNIVERSITÀ  
DI TORINO

# Debugging node.js

Prof. Fabio Ciravegna

Dipartimento di Informatica

Università di Torino

fabio.ciravegna@unito.it



# Learning Objectives

- You will learn to debug a client server architecture where the client is a browser and the server is an Express server
  - how to debug the server using WebStorm
  - how to debug the client (browser) using Chrome

# Debugging node.js

- Node.js does not run in a browser
  - it runs on a server

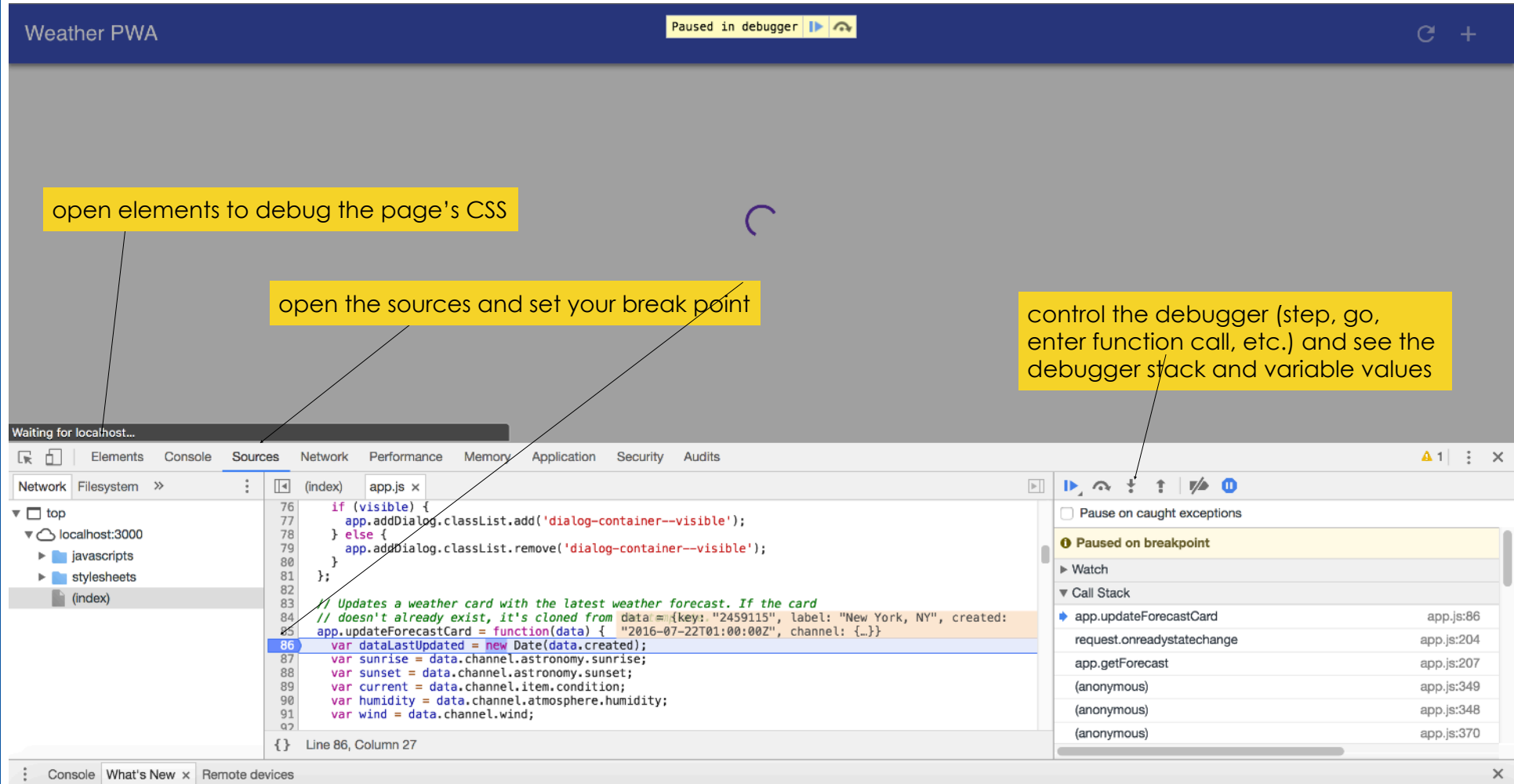
# Debugging HTML/Javascript in Chrome

<https://developer.chrome.com/devtools/docs/javascript-debugging>

- We will be using IntelliJ
  - if you have used AndroidStudio or WebStorm, it is the same product with different flavours
- There are two parts in any client/server architecture:
  - the client (e.g. a browser): this can be debugged with Chrome
  - the server (the nodejs server), this is to be debugged with IntelliJ

# Chrome debugging (client)

- open from right menu (or view>developers>javascript console on a Mac)



Weather PWA

Paused in debugger

open elements to debug the page's CSS

open the sources and set your break point

control the debugger (step, go, enter function call, etc.) and see the debugger stack and variable values

Waiting for localhost...

Elements Console Sources Network Performance Memory Application Security Audits

Network Filesystem >> (index) app.js x

top

localhost:3000

- javascripts
- stylesheets
- (index)

```

76  if (visible) {
77    app.addDialog.classList.add('dialog-container--visible');
78  } else {
79    app.addDialog.classList.remove('dialog-container--visible');
80  }
81  };
82
83  // Updates a weather card with the latest weather forecast. If the card
84  // doesn't already exist, it's cloned from data = {key: "2459115", label: "New York, NY", created:
85  app.updateForecastCard = function(data) { "2016-07-22T01:00:00Z", channel: {...}}
86  var dataLastUpdated = new Date(data.created);
87  var sunrise = data.channel.astronomy.sunrise;
88  var sunset = data.channel.astronomy.sunset;
89  var current = data.channel.item.condition;
90  var humidity = data.channel.atmosphere.humidity;
91  var wind = data.channel.wind;
92
93  }
  
```

Line 86, Column 27

Paused on breakpoint

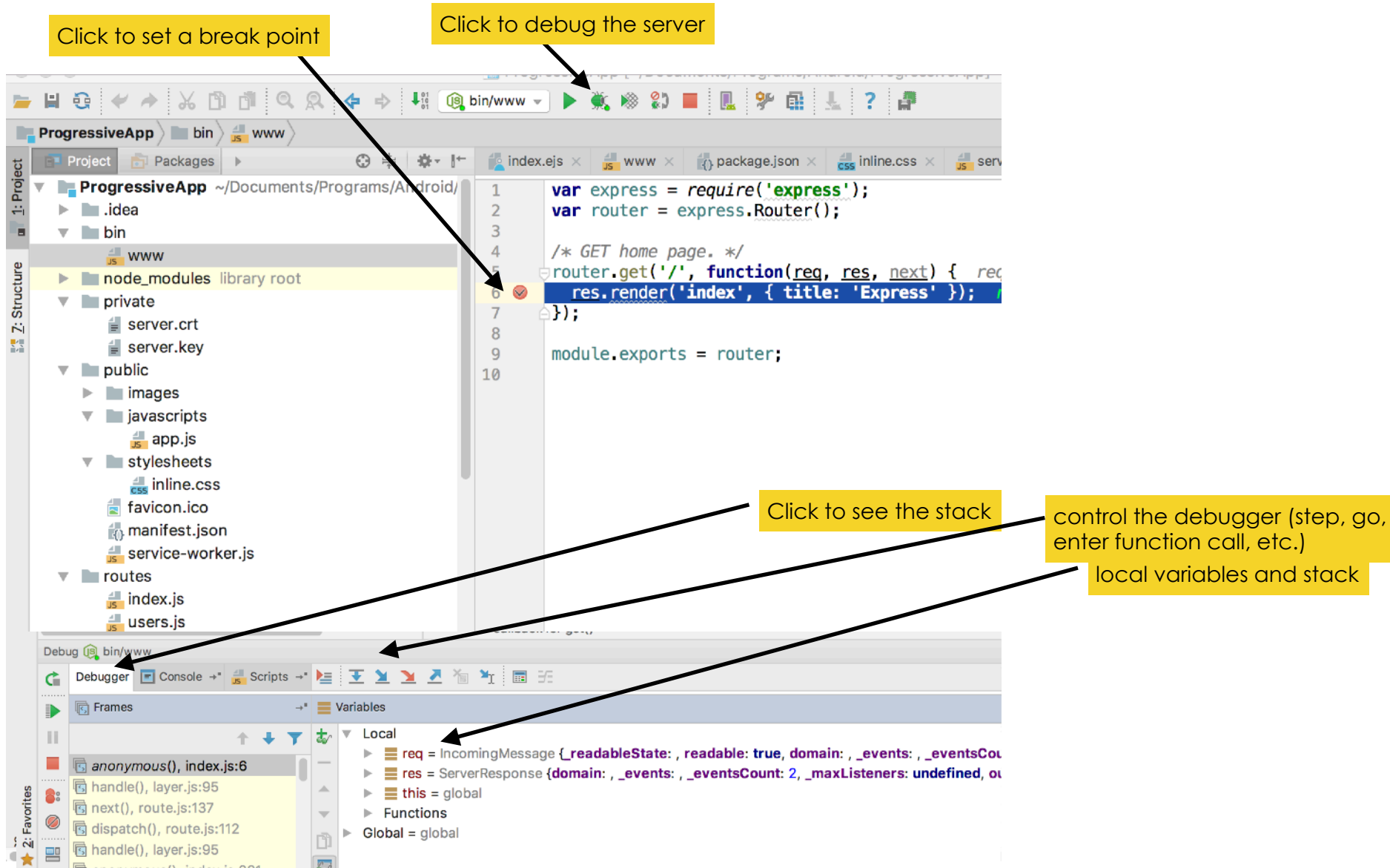
Watch

Call Stack

app.updateForecastCard	app.js:86
request.onreadystatechange	app.js:204
app.getForecast	app.js:207
(anonymous)	app.js:349
(anonymous)	app.js:348
(anonymous)	app.js:370

Console What's New Remote devices

# Debugging the server



Click to set a break point

Click to debug the server

Click to see the stack

control the debugger (step, go, enter function call, etc.)

local variables and stack

```

1  var express = require('express');
2  var router = express.Router();
3
4  /* GET home page. */
5  router.get('/', function(req, res, next) { req
6  res.render('index', { title: 'Express' });
7  });
8
9  module.exports = router;
10

```

Debug Console

Frames

- anonymous(), index.js:6
- handle(), layer.js:95
- next(), route.js:137
- dispatch(), route.js:112
- handle(), layer.js:95

Variables

Local

- req = IncomingMessage {readableState: true, domain: , \_events: , \_eventsCou
- res = ServerResponse {domain: , \_events: , \_eventsCount: 2, \_maxListeners: undefined, ou
- this = global

Functions

Global = global

# 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

## Welcome to My Class

Please fill the form

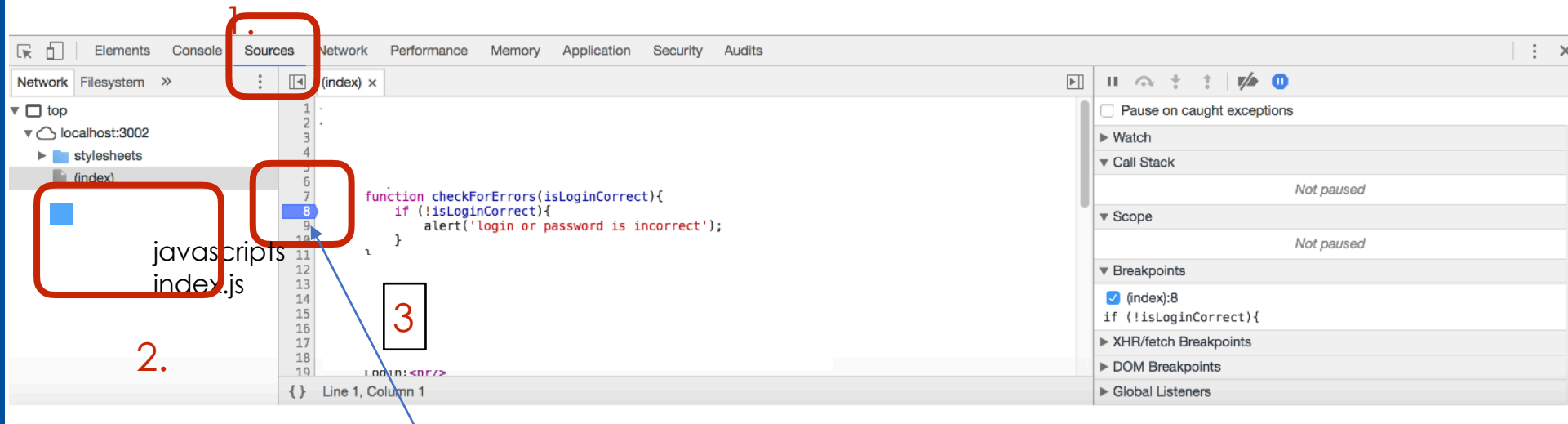
Login:

ww

Password:

Submit

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



Click on the line number to set up a break point



Week1b [Week1] ~/Documents/Teaching/COM3504-6504 Intelligent Web/Lab Classes/Week 2/Week1b - .../routes/index.js [Week1]

bin\www

2. Click debug

1. Click to set break point

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

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

/* POST from form. */
router.post('/welcome', function(req, res, next) {
  var login= req.body.login;
  var password= req.body.password;
  if (login=='paula'){
    res.render('welcome', { title: login, login_is_correct: true });
  } else {
    res.render('index', { title: 'My Class', login_is_correct: true });
  }
});

module.exports = router;
```

callback for post()

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

POST /welcome 200 1.287 ms - 661

GET /stylesheets/style.css 304 0.398 ms --

GET /stylesheets/style.css 304 0.396 ms --

4: Run 6: TODO Terminal

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