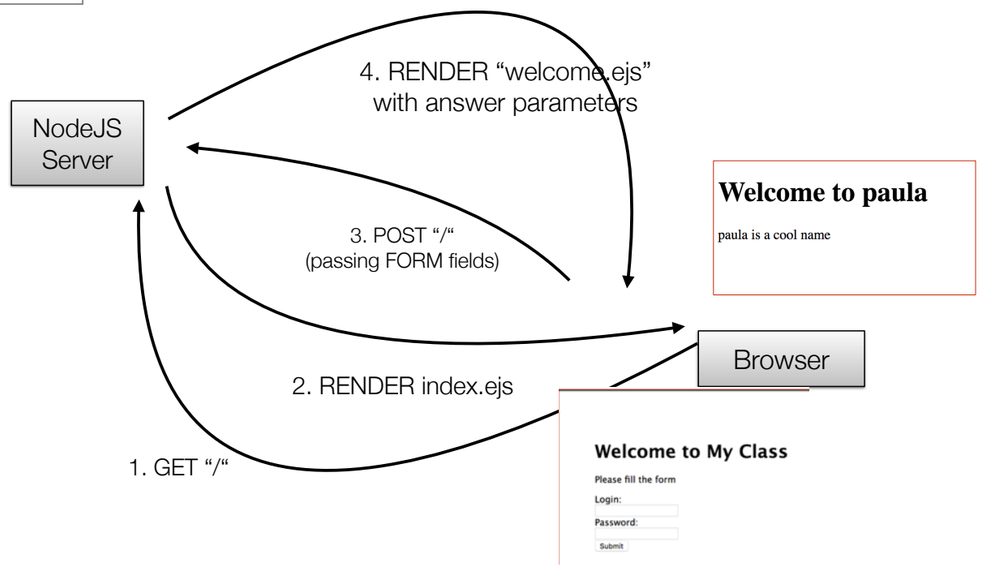
Week 2 Exercise 1

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

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

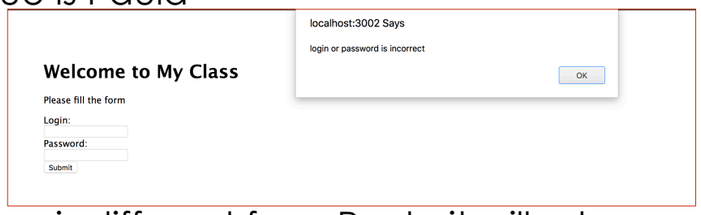
Client/server interplay



Next Step

- 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

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

Server side

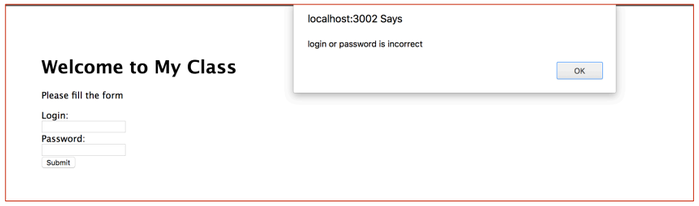
- As the form posts to the route /welcome. we must modify the file routes/index.js by adding:

  router.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

- otherwise render index.ejs again with an alert



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_correct %>)">
```

- then create public/javascripts/index.js and insert the following code

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

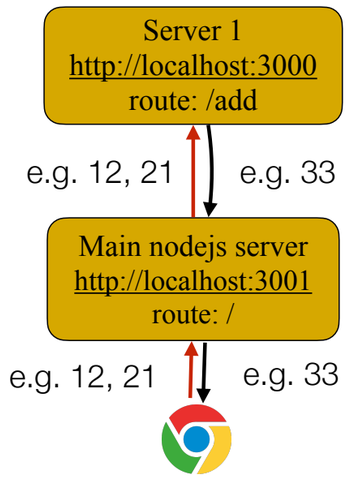

Week 2.b Lab Class: Server to Server Communication

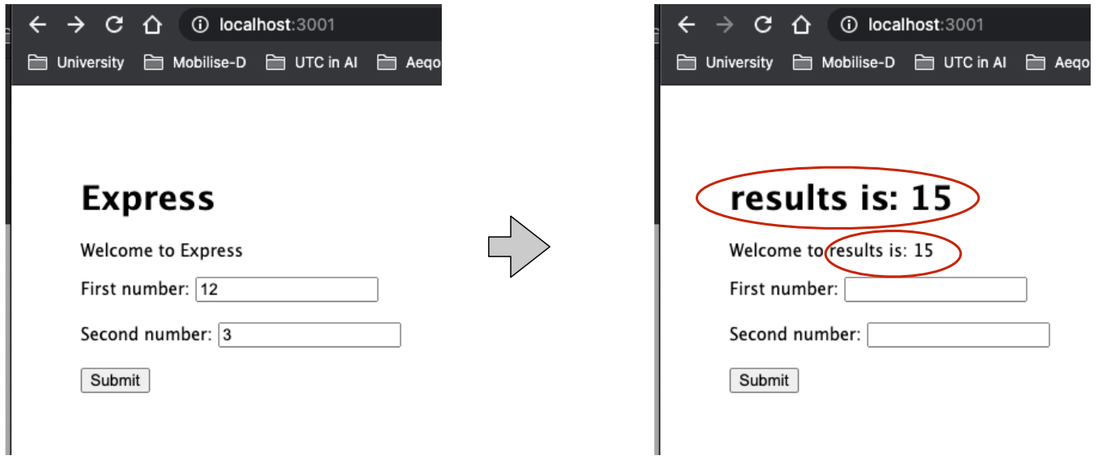
Professor Fabio Ciravegna
Department of Computer Science
University of Sheffield
f.ciravegna@shef.ac.uk
http://staffwww.dcs.shef.ac.uk/people/F.Ciravegna/

Scope

- In this exercise you will learn to create a server constellation composed of:
  - the client receives an EJS file with a form taking two integers
  - the client posts the numbers to the main node sever
  - the main node server receives the two numbers from the browser and sends them to the supporting server
  - The supporting server will sum the two numbers and will return the sum to the main server in json format
  - The main server will serve the EJS file again with the form but will change the title into the result of the sum

Example





Starting point

- you are provided with an initial main server serving the ejs file and returning a value
- you have to implement:
  - server 1 implementing the addition
  - the fetch to send the data and to retrieve the addition
  - the return of the result to the browser

Week 2 Lab Class 2.c
Working with Promises

Prof. Fabio Ciravegna
Department of Computer Science
The University of Sheffield
f.ciravegna@shef.ac.uk

Promises

- In this exercise we will work with promises
- In particular we will convert an express route based on callbacks
  - into an equivalent route using promises
- We will first see this in action with an example and then you will be asked to do a similar exercise

Writing two files

- We will use an npm library called fs which is used to access files
  - All functions accessing files are asynchronous operations
- The code provided (Week 2.c Lab Class - Promises Introduction) shows an eps file with two buttons
  - each of them fetches a route in routes/index.js
  - the first route writes the two files using callbacks
  - the second route does the same operations using promises
- Note:
  - we will use the npm module fs in its base form
    - however fs has a version using promises - so it is suggested that after today you use the versions with promises directly rather than the base implementation

With callbacks

```
/**
 * this route writes two files iin sequence using callbacks
 */
router.get('/get_photos', function (req, res, next) {
    const data = "console.log('Hello World')";
    fs.writeFile('file1.js', data, (error) => {
        if (error) {
            console.error(err);
            res.writeHead(500, {'Content-Type': 'text/plain'});
            res.end('error in writing files' + err);
        }
        console.log('The file2.js has been saved!');
        fs.writeFile('file2.js', data, (error) => {
            if (error) {
                console.error(err);
                res.writeHead(500, {'Content-Type': 'text/plain'});
                res.end('error in writing files' + err);
            }
            console.log('The file1.js has been saved!');
            res.writeHead(200, {'Content-Type': 'text/plain'});
            res.end('both files were written');
        });
    });
});
```


With Promises

```
let writeFilePromise= function (data, path) {
    return new Promise((resolve, reject) => {
        fs.writeFile(path, data, (error) => {
            if (error) reject();
            else resolve();
        })
    });
}
router.get('/get_photos_promises', function (req, res, next) {
    const data = "console.log('Hello World')";
    writeFilePromise(data, './public/images/image1.png')
        .then(() => writeFilePromise(data, './public/images/image2.png'))
        .then(() => {
            res.writeHead(200, {'Content-Type': 'text/plain'});
            res.end('both files were written');
        })
        .catch(err => {
            console.error(err);
            res.writeHead(500, {'Content-Type': 'text/plain'});
            res.end('error in writing files' + err);
        })
});
```


Exercise

- Do a similar task with another function from the fs module: fs.access
  - which checks if a file exists
- You are given a version of a programme that checks if three images exist
  - the images are under /public/images/
  - the code uses callbacks
- Replace the code using promises