1. **Functional**

This web app has for objective to collect induvial information without losing input data on at client side and submit it to a nodejs server.

The application uses HTML pages which request **js** files that renders components and elements within a browser. The following directory structure gives an overview of how the server paths are configured in order to cater from client to server and vice versa.

1. **Directory Structure**

index.htm

client.js

server.js

package.json

**signup**/

index.htm

signup.js

**api**/v1/signup

**test**/

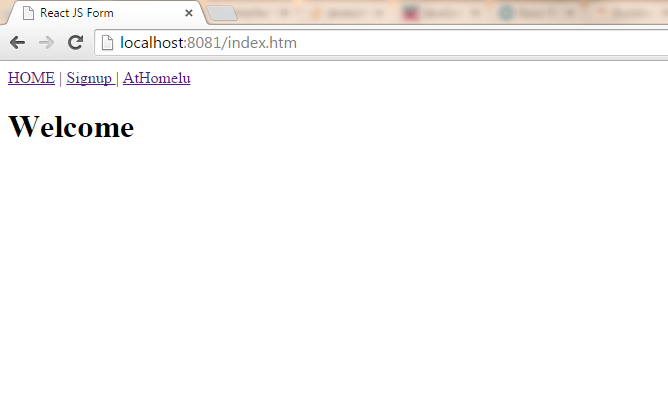
**node\_modules**/

Once the index file in the main path is accessed, the page header makes request for external library support to help render ReactJS components in the browser:

<script src="https://fb.me/react-0.14.1.js"></script>

<script src="https://fb.me/react-dom-0.14.1.js"></script>

<script src="https://cdnjs.cloudflare.com/ajax/libs/babel-core/5.8.23/browser.min.js"></script>



Initially the **index.htm** page contains two Divs with ID “headermenuid” and “container” each with the respective purpose to hold the menu header and welcome message, resulting in the above image when accessed in the environment. Note that these two divs are rendered with the help of **client.js**.

1. **Home page**
2. **Headers**

The header menu helps user navigate the application as follow:

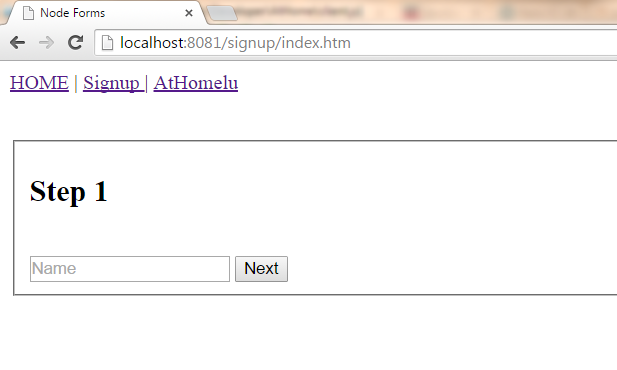
* Home, when click will always redirect you to this index.htm page, even when accessed from other locations.
* Signup, will direct you to the signup directory and loading the signup menu and form through an **index.htm** page.
* Athome, will redirect you to an external link (http://www.athome.lu/)

1. **Welcome**

This section is rendered with the help of the **Welcomestr** reactjs class.

1. **Signup page**

Let assume you have clicked on the signup form, the application will load the **index.htm** page located in the **signup** directory. The page is used as the platform to help render both the head menu and multi-step form rendered by a **signup.js,** see image.



1. **Headers**

The head menu on this page is generated as on the home page with only link changes to adapt to current directory, home page and **Athomelu** staying the same.

1. **Multi-steps Form**

In order to generate and render this multi-step form, I had to break form in several components and have reactjs render it for me. Considering that the form has only three main collection facets and a data acknowledgement component to notify the user (see picture below).

* **Component I**: Helps to get the user name. Using div and field set, I render the step one component together with its children (input, button, etc.). The “Next” button, when clicked, will request the rendering for the second components.
* **Component II**: Helps to get the user First Name and has the same functions on its Next button as component I. There is however an extra feature allowing user to GO back to the previous component (I) to modify name if needed. Thanks to the ReactJS **defaultvalue** input property which helps set initial value and keep change if input entry changes.
* **Component III**: Last step of data collection, has also the functionality of both component I and II together with an added button (Validate) which serves to submit the form to the servers.

****

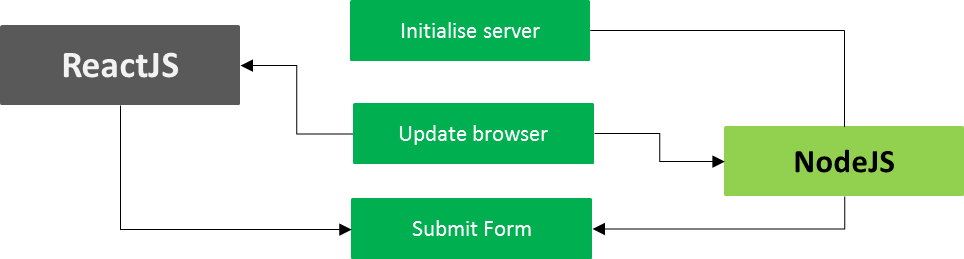
* **Component IV**: As per the picture above, this component when rendered, displays a notification to the user together with the “Name, First name and email” the user entered throughout the multi-step process. The “back to home” button for function to redirect the user to the home page.

1. **API**

So far, I have elaborated on how Html element where rendered at runtime with the help of reacts together with the approach of populating a multi-step form.

Nodejs in our case, have been instrumental in helping on the following:

* **Server initialisation**: Act as a platform to help run the application on the web.
* **Updating the browser**: Helps update the client side without the need to reload each time the web page. The request of a new components such as step 2 or step 3 to be displayed the web page
* **Form submit**: Data generated by Reactjs throughout the multi-step form is submitted to server using an AJAX POST method to the path **api/v1/signup** to which NodeJS is listening (Configuration set in the **server.js** file). The server reply to the Reactjs with a status code or error to update the state whose value is used to update the **component IV** accordingly as above.



**Signup.js**

This file is used to render form components in divs on the multi-step form. The structure is as follow:

* **Step I:** Create 4 reacts classes which includes divs and components forming part of every form step. Based on input validation (Is not empty/null), each input value is stored in variable and its state saved. Using the **defaultvalue** to make to keep input value even if new components have been rendered on the webpage.
* **Step II**: Data gathered is then submitted to server where the server.js through nodejs is listening to the path **api/v1/signup** (no need to create the directory) and update the client with POST data on status 200 code. Data received is then used to notify the user through a success reactjs class which renders components accordingly.

1. **Step by step guide**

First download/clone the file from the Github <https://github.com/joembaya/ReatcJS-Form> and save to a directory on your computer.

1. **Requirements**

* Nodejs
* Reactjs (CDN/local) //no need to install
* Mocha // for testing purpose
* Chai // for testing purpose

1. **Starting the server**

Make sure that node is **installed** on the PC, if not get an install based on your OS. After nodejs is installed, open command and navigate to the working directory (the project directory) and make use of NPM to install globally or locally, -g or –save respectfully, the **requirements** above.

Next, still in command run “**node server.js**”, first, the **server.js** will start server and listen to port **8081** (configured in the server.js file and can be changed) from which the application will be ran. Note that port can be modified in the server file and rerun node server.js command to restart the server.

Next go to **http://localhost:8081/index.htm** to access web app and follow the links.