Information collection Form

**Overview**

This web pages has for objective to collect induvial information without losing theme on at client side.

|  |  |  |
| --- | --- | --- |
| **Main directory** | **subfolders/files** | **Description** |
| index.html |  | main page |
| signup | index.htm | handling the form |
| Api | v1/signup , server.js | normally intended to process form data |
| dist |  | contains ReactJS linked locally |
| .idea |  | to be ignored, generated by IntelliJ |
| components | signup.js, data.js |  |
|  | | |

**Setting up**

**How it works**

When the main index.htm is access, ReactJS renders the approiate data to configured element. That is the header and the welcome message display on page load.

**Data.JS**

var Counter = React.createClass({

render: function () {

return (

<div> <a href="/" >HOME</a> | <a href="signup" > Signup </a> | <a href="http://www.athome.lu"> AtHomelu </a> </div>

);

}

});

ReactDOM.render(

<Counter />,

document.getElementById('menuhead')

);

var Counters = React.createClass({

render: function () {

return (

<h1>Welcome</h1>

);

}

});

ReactDOM.render(

<Counters />,

document.getElementById('welcomeid')

);

**Note**: Test utilities could have been useful here.

The same setup is configure when loading the signup section of the program. Which allows users to capture their name, first name and email. The form is entirely populated through ReactJS and rendered on the screen using the index.htm files in that directory.

**Signup.JS**

//Showing Menu

var Menuheadload = React.createClass({

render: function () {

return (

<div> <a href="../" >HOME</a> | <a href="/" > Signup </a> | <a href="http://www.athome.lu"> AtHomelu </a> </div>

);

}

});

ReactDOM.render(

<Menuheadload />,

document.getElementById('menuhead')

);

var Counters = React.createClass({

getInitialState: function() {

return {

name: '',

fname:'',

email: '',

message: ''

};

},

//Handling validation and current data

handleSubmit: function(e) {

e.preventDefault();

var userName = this.state.name.trim();

var userFname = this.state.fname.trim();

var userEmail = this.state.email.trim();

if(!userName || !userEmail) return;

this.setState({

name: '',

email: '',

message: 'Please wait...'

});

this.props.onFormSubmit({

userName: userName,

userName: userFname,

userEmail: userEmail,

url: "/api/v1/signup"

}, function(data) {

this.setState({ message: data });

});

},

changeName: function(e) {

this.setState({

name: e.target.value

});

},

changeFname: function(e) {

this.setState({

fname: e.target.value

});

},

changeEmail: function(e) {

this.setState({

email: e.target.value

});

},

render: function () {

return (

//Creating form

<form onSubmit={this.handleSubmit} method="post" action="./">

<div id='fieldsets'>

<fieldset id='firstfld' >

<h3>Step 1</h3>

<br / ><br />

<input name='name' type='text' placeholder='Name'/>

<span> </span>

<input type='button' name='next' id='currid' value='Next' />

</fieldset>

<fieldset id='sndfld' class='' style={{display: 'none'}} >

<h3>Step 2</h3>

<br / ><br />

<input name='fname' type='text' placeholder='Firstname'/>

<span> </span><input type='button' name='next' class='' value='Next' id='sndid' />

<br / ><br />

<input type='button' name='next' class='' value='Back' id='fstbckid' />

</fieldset>

<fieldset id='trdfld' style={{display: 'none'}}>

<h3>Step 3</h3>

<br / ><br />

<input name='email' type='email' placeholder='Email'/>

<input type='submit' name='submit' class='submit action-button' value='Validate' />

<br / ><br />

<input type='button' name='next' class='next action-button' value='Back' id='thrdidbck' />

</fieldset>

</div></form>

);

}

});

//Displaying

ReactDOM.render(

<Counters />,

document.getElementById('datafrmid')

);

Note: The form submission is then managed with the event handler **handleSubmit**, which I try and set the Server.js to listen to the path. Might not be the best approach but my logic was alright as I need more practices with node and Reactjs.

**Server.JS**

It basically meant to listen to the request of the **api/v1/signup** page and trigger the POST element. The layout below demonstrate the way to get it done but had challenges linking the path and updating the front-end.

**var *http*** = require(**'http'**),  
 ***util*** = require(**'util'**),  
 ***fs*** = require(**'fs'**),  
 ***url*** = require(**'url'**),  
 ***qs*** = require(**'querystring'**);  
   
**var *server*** = ***http***.createServer(**function** (req,res){  
   
 **var** url\_parts = ***url***.*parse*(req.**url**,**true**);  
 **var** body = **''**;  
 **if**(url\_parts.**pathname** == **'/api/v1/signup'** && req.**method** === **'POST'**){  
  
 **console**.log(**'Request found with POST method'**);   
 req.on(**'data'**, **function** (data) {  
 body += data;  
 **console**.log(**'I have received the data data:'**+data);  
 });  
 req.on(**'end'**, **function** () {  
   
 **var** POST = ***qs***.*parse*(body);  
 *// sending data received* res.end(**"Sent data are name:"**+POST.name+**" First Name:"**+POST.**fname** +**" First Name:"**+POST.**email**);  
   
 });  
   
 }  
  
});  
***server***.listen(8080);  
**console**.log(**'Server listening at localhost:8080'**);

In closing the application does most of the job beside the submitting and updating user with feedback. I recon that my ReactJS and NodeJS need some practices in order to get going. Which is not impossible.