**Exercise: Creating a Tree View**

**ReactJS Component Life Cycle Methods**

**React Router**

**ReactJS Tutorials**

**Build Your React App**

**Getting Started with React**

**Presented by:**

Safia Zihmou

safiazihmou@mphasis.com

**ReactJS**

**Training**

1. **Getting Started with React:**

|  |  |
| --- | --- |
| **Install**  **NodeJS** | <https://wsvincent.com/install-node-js-npm-windows/> |
| **Source Code Editor** | <https://code.visualstudio.com/> |

1. **Build Your React App:**

|  |  |
| --- | --- |
| **Using**  **“Create-react-app”** | <https://facebook.github.io/create-react-app/docs/getting-started> |
| **Manually** | <https://blog.usejournal.com/creating-a-react-app-from-scratch-f3c693b84658> |

1. **ReactJS Tutorials:**

* **ReactJS Documentation:**

<https://reactjs.org/docs/getting-started.html>

* **OpenClassRoom:**

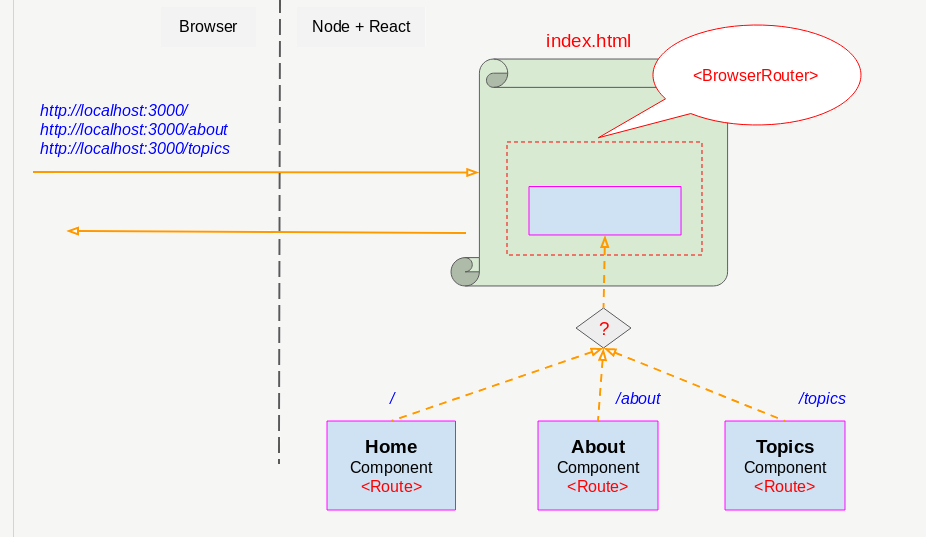
**EN Version :**

<https://openclassrooms.com/en/courses/4286486-build-web-apps-with-reactjs>

**FR Version :**

<https://openclassrooms.com/fr/courses/4664381-realisez-une-application-web-avec-react-js>

1. **React Router:**

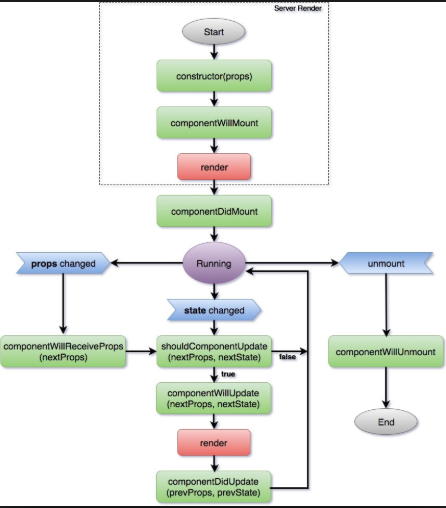


React Router means a standard routing library in React. This makes the interface of the application synchronous with the browser URL. The React Router allows you to clearly route the "data flow" in your application. This equates to an affirmation. If you have this URL, it will be equivalent to this Route and the interface will be as follows.

**Link:**

<https://reacttraining.com/react-router/web/guides/quick-start>

1. **ReactJS - Component Life Cycle:**



**Links:**

* <https://openclassrooms.com/fr/courses/4664381-realisez-une-application-web-avec-react-js/4664881-apprivoisez-le-cycle-de-vie-des-composants>
* <https://www.tutorialspoint.com/reactjs/reactjs_component_life_cycle.htm>

1. **Exercise:** **Creating a Tree View:**

The Goal of this Exercise is to Create extendable Tree View Component using ReactJS.

By following All the Steps below we will have the following result:

**Tasks:**

* **Task 1:** Create the Tree & Print the Principal Nodes:

**Goals:**

* Use Classe To Create the component .
* Import & Export Component.
* **Task 2:** Display the content of each Node by Clicking on its name

**Goals:**

* **Task 3:** Add a Button to display all the items of the tree.

**Goals:**

* **Task 4:** Add an Icon to Each Node : by clicking on this icon we can change the name of the selected Node .

**Goals:**

**Solution:**

* **Task 1:**

Create a new file “**Tree.js**” in **src** folder:

import React, { Component} from "react"

class Tree extends Component {

render(){

return(

<div>

OutPut Tree

<ul>

{cours.map((cour,index) => {

return <li key={index}>{cour.title}</li>

})}

</ul>

</div>

)

}

}

The key attribute preserve a correct association between an original datum ,in the list , and the corresponding component

const cours = [

{

title: "Folder\_1", children: [

{title: "SubFolder1.1"},

{

title: "SubFolder1.2", children: [

{title: "File1"},

{title: "File2"},

]

},

]

},

{

title: "Folder\_2", children: [

{title: 'SubFolder2.1'},

{title: 'SubFolder2.2'},

]

},

{title: "Folder\_3"},

]

export default Tree

In the “**App.js**” in **src** folder add the code bellow :

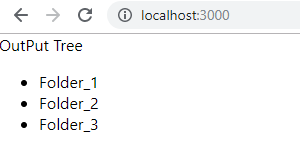
import React , {Component} from "react"

import Tree from "./Tree"

Import Tree from “../Tree” :

<Tree />

**Result :**



class Menu extends Component {

render(){

return(

<div>

<Tree />

</div>

)

}

}

export default Menu

* **Task 2:**

Create a new file “**TreeNode.js**” in **src** folder:

import React , {Component} from "react"

The key attribute preserve a correct association between an original datum ,in the list , and the corresponding component

class TreeNode extends Component {

state : StateType = { open : false}

handleClickTitle = () =>{

this.setState({ open : !this.state.open})

}

render(){

const {title , children } = this.props

return (

<div>

<div onClick={this.handleClickTitle}>{title}</div>

{this.state.open && (

<ul>

{children && children.map((item , index) => {

return <li key={index}><TreeNode {...item} /></li>

})}

</ul>

)}

</div>

)

}

}

export default TreeNode

Create the “**Tree.js**” file :

The key attribute preserve a correct association between an original datum ,in the list , and the corresponding component

import TreeNode from "./TreeNode"

class Tree extends Component {

The key attribute preserve a correct association between an original datum ,in the list , and the corresponding component

render(){

return(

<div>

OutPut Tree

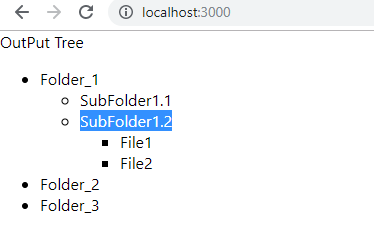
<ul>

{cours.map((cour,index) => {

return <li key={index}><TreeNode {...cour} /></li>

})}

**Result :**



</ul>

</div>

)

}

}

* **Task 3:**

Our Code becomes like this :

**“Tree.js”:**

import React , { Component} from "react"

import TreeNode from "./TreeNode"

class Tree extends Component {

state = {expandAll : false }

handleExpandAll = () => {

this.setState({ expandAll : true})

}

handleUnExpandAll = () => {

this.setState({ expandAll : false})

}

render(){

return(

<div>

OutPut Tree

<ul>

{cours.map((cour,index) => {

return <li key={index}><TreeNode {...cour} expandAll={this.state.expandAll} onUnExpandAll = {this.handleUnExpandAll} /></li>

})}

</ul>

<button onClick={this.handleExpandAll} >ExpandAll</button>

</div>

)

}

}

const cours = [

{

title: "Folder\_1", children: [

{ title: "SubFolder1.1" },

{

title: "SubFolder1.2", children: [

{ title: "File1" },

{ title: "File2" },

]

},

]

},

{

title: "Folder\_2", children: [

{ title: 'SubFolder2.1' },

{ title: 'SubFolder2.2' },

]

},

{ title: "Folder\_3" },

]

export default Tree

**“TreeNode.js”:**

import React , {Component} from "react"

class TreeNode extends Component {

state = { open : false}

handleClickTitle = () =>{

const {onUnExpandAll} = this.props

onUnExpandAll()

this.setState({ open : !this.state.open})

}

static getDerivedStateFromProps(props) {

if (props.expandAll) {

return { open: true }

}

else {

return null

}

}

render(){

const {title , children , expandAll , onUnExpandAll} = this.props

return (

<div>

<div onClick={this.handleClickTitle}>{title}</div>

{this.state.open && (

<ul>

{children && children.map((item , index) => {

return <li key={index}><TreeNode {...item} expandAll={expandAll onUnExpandALL={}} /></li>

})}

</ul>

)}

</div>

)

}

}

export default TreeNode

* **Task 4:**

**“Tree.js”:**

import React, {Component} from "react"

import TreeNode from "./TreeNode"

type PropsType = {}

type StateType = {

expandAll: boolean,

}

class Tree extends Component {

props: PropsType

state: StateType = {

expandAll : false ,

cours : [

{

title: "Folder\_1", children: [

{ title: "SubFolder1.1" },

{

title: "SubFolder1.2", children: [

{ title: "File1" },

{ title: "File2" },

]

},

]

},

{

title: "Folder\_2", children: [

{ title: 'SubFolder2.1' },

{ title: 'SubFolder2.2' },

]

},

{ title: "Folder\_3" },

]

}

handleExpandAll = () => {

this.setState({expandAll : true })

}

handleUnexpandAll = () =>{

this.setState({expandAll : false })

}

handleTreeNodeChange = (index) => (value) => {

let newCours = []

newCours = newCours.concat(

this.state.cours.slice(0, index),

value,

this.state.cours.slice(index+1)

)

this.setState({cours : newCours})

}

render() {

return (

<div>

<ul>

{this.state.cours.map((cour, index) => {

return <li key={index}><TreeNode value={cour} expandAll={this.state.expandAll} onTitleClick={this.handleUnExpandAll} onChange={this.handleTreeNodeChange(index)} /></li>

})}

</ul>

<button onClick={this.handleExpandAll}>ExpandALL</button>

</div>

)

}

}

export default Tree

**“TreeNode.js”:**

import React, { Component } from 'react'

type PropsType2 = {

value: Object,

onChange: Function,

expandAll: Boolean,

}

type StateType = {

open: Boolean,

readOnly : Boolean,

title: String,

}

class TreeNode extends Component {

props: PropsType2

state: StateType = {

open: false ,

readOnly : true

}

static getDerivedStateFromProps(props) {

if (props.expandAll) {

return { open: true }

}

else {

return null

}

}

handleTitleClick = () => {

const { onTitleClick } = this.props

onTitleClick && onTitleClick()

this.setState({ open: !this.state.open })

}

handleChange = () => {

const { onChange , value } = this.props

const { title } = this.state

onChange({

...value,

title: title,

})

}

handleToggleEdit = () => {

const { readOnly } = this.state

this.setState({readOnly : !readOnly})

if (readOnly) {

this.setState({title: this.props.value.title})

}

else {

this.handleChange()

}

}

handleInputBlur = () =>{

this.setState({readOnly : true})

this.handleChange()

}

handleTitleChange = (event) => {

this.setState({title: event.target.value})

}

handleSubTreeNodeChange = (index) => (value) => {

const { onChange } = this.props

let newChildren = []

newChildren = newChildren.concat(

this.props.value.children.slice(0, index),

value,

this.props.value.children.slice(index+1)

)

onChange && onChange({

...this.props.value,

children : newChildren,

})

}

render() {

const { value, expandAll , onTitleClick} = this.props

const { title } = this.state

return (

<div>

{this.state.readOnly

? (<label onClick={this.handleTitleClick}>{value.title}&nbsp;</label>)

: (<input value={title} onChange={this.handleTitleChange} onBlur={this.handleInputBlur} />)}

<i className="far fa-edit" onClick ={this.handleToggleEdit}></i>

{

this.state.open && (

<ul>

{value.children && value.children.map((child, index) => {

return <li key={index}><TreeNode value={child} expandAll={expandAll} onTitleClick={onTitleClick} onChange={this.handleSubTreeNodeChange(index)} /></li>

})}

</ul>

)

}

</div >

)

}

}

export default TreeNode