

Projet 11 REACT BASE + Json-Server

Ressources :

<https://github.com/michelonlineformapro/Projet-11-React-BASIC-START-Json-server>

<https://fr.reactjs.org/>

Objectif :

Comprendre le fonctionnement de la librairie (pseudo-Framework) React JS et utilisé la programmation modulaire à l'aide de composants réutilisable, comprendre les refs, les props, les classes et les hooks pour réaliser une application web API REST via json server et effectuer des opérations de CRUD simple.

Présentation :

React est un projet open-source, distribué sous la licence MIT et piloté par Facebook. Leurs produits web et mobile tels que Facebook, Messenger, Instagram, reposent en grande partie sur cette technologie. Comme React est open-source, vous pouvez accéder au code source directement sur GitHub, proposer une feature, ou même notifier d'un problème (*issue*).

L'ambition de React est de **créer des interfaces utilisateurs**, avec un outil **rapide** et **modulaire**. L'idée principale derrière React est que vous construisiez votre application à partir de composants. **Un composant regroupe à la fois le HTML, le JS et le CSS**, créés sur mesure pour vos besoins, et que vous pouvez **réutiliser** pour construire des interfaces utilisateurs.

Un composant = HTML + CSS + JS.

Pourquoi React

Je vous parlais plus tôt des avantages/inconvénients de chaque Framework. Sans même rentrer dans l'aspect technique, voici quelques-uns des atouts de React :

Sa communauté

Particulièrement active, elle vous facilite la vie. Lorsque vous cherchez votre problème sur Internet, il est quasiment impossible que personne n'ait déjà rencontré le même problème que vous. D'autant plus que React compte de très grosses entreprises parmi ses utilisateurs (Netflix, Twitter, Paypal, Airbnb pour n'en citer que quelques-unes). Vous pouvez être sûr qu'un autre ingénieur s'est déjà trouvé confronté à votre problème. Par exemple, lorsque vous trouvez une question posée sur [Stack Overflow](#) (en anglais). L'équipe de React répond également aux *issues* (problèmes) [sur le repository GitHub de React](#) (en anglais). Mais il existe un nombre immense de newsletters, blogs, chaînes YouTube, créés par des utilisateurs - leur dynamisme vous donne toujours envie de tester de nouveaux outils. Je vous en conseillerai d'ailleurs quelques-uns dans le dernier chapitre de ce cours.

Sa documentation

La [documentation de React](#) est riche, régulièrement mise à jour et intégralement traduite en français FR.

Ses opportunités professionnelles

Comme il s'agit d'un des frameworks les plus populaires, les opportunités professionnelles sont particulièrement nombreuses. Dans [l'enquête annuelle State of JS de 2020](#), 100 % des personnes déclaraient connaître React, et sur 21 000 sondés, 17 000 déclaraient utiliser React (qui a d'ailleurs [gagné l'Award 2019](#) de la technologie la plus utilisée).

Alors, je ne suis peut-être pas très objective car il s'agit de ma bibliothèque de prédilection, mais vous l'aurez compris : maîtriser React offre de nombreux avantages.

- Un **framework JS** est un **ensemble de classes, fonctions et utilitaires** qui nous facilitent la création d'applications pour les navigateurs ou mobiles.
- L'un des outils les plus populaires, **React**, qui est une bibliothèque aussi bien qu'un framework, permet de **créer des interfaces utilisateurs**.
- L'approche technique de React est de créer du **code modulaire**, à base de **composants réutilisables**.
- Trois des avantages de React sont sa **communauté**, sa **documentation** et ses **opportunités professionnelles**.
- Vous savez maintenant comment **transformer un simple fichier de HTML en React** – et avez créé votre premier composant !

Du code modulaire et JSX

- Une interface utilisateur (ou *UI*) est **constituée de multiples composants React** qui :
 - Sont **réutilisables** ; par exemple, un bouton, un élément dans une liste, un titre,
 - **Regroupent** la structure, les styles et le comportement d'un élément,
 - Sont **traduits par React** en gros objets, qui sont **ensuite greffés au DOM** ;
- Le JSX est une syntaxe créée par React permettant d'écrire du JavaScript. Il faut suivre quelques règles :
 - Deux composants doivent toujours être wrappés dans **un seul composant parent**,
 - Les noms des composants **commencent par une majuscule**,
 - Les balises des composants **doivent être refermées**.

Create React App (CRA) Web Pack + Babel + ESLint

Fonctionnalité de ces 3 outils :

- Gérer les différentes dépendances (bibliothèques) utilisées par notre app ;
- Optimiser le chargement de notre code dans les navigateurs ;
- Importer du CSS et des images ;
- Gérer les différentes versions de JavaScript ;
- Faciliter l'expérience de développement, en rechargeant la page lorsque le code est modifié.

1. Web pack est un outil logiciel open-source de type « module bundler », conçu pour faciliter le développement et la gestion de sites et d'applications web modernes

<https://webpack.js.org/>

2. Traduit de l'anglais-Babel est un transcompilateur JavaScript gratuit et open source qui est principalement utilisé pour convertir le code ECMAScript 2015+ en une version rétrocompatible de JavaScript pouvant être exécutée par des moteurs JavaScript plus anciens.

<https://babeljs.io/>

3. Traduit de l'anglais-ESLint est un outil d'analyse de code statique pour identifier les modèles problématiques trouvés dans le code JavaScript. Il a été créé par Nicholas C. Zakas en 2013. Les règles dans ESLint sont configurables et des règles personnalisées peuvent être définies et chargées.

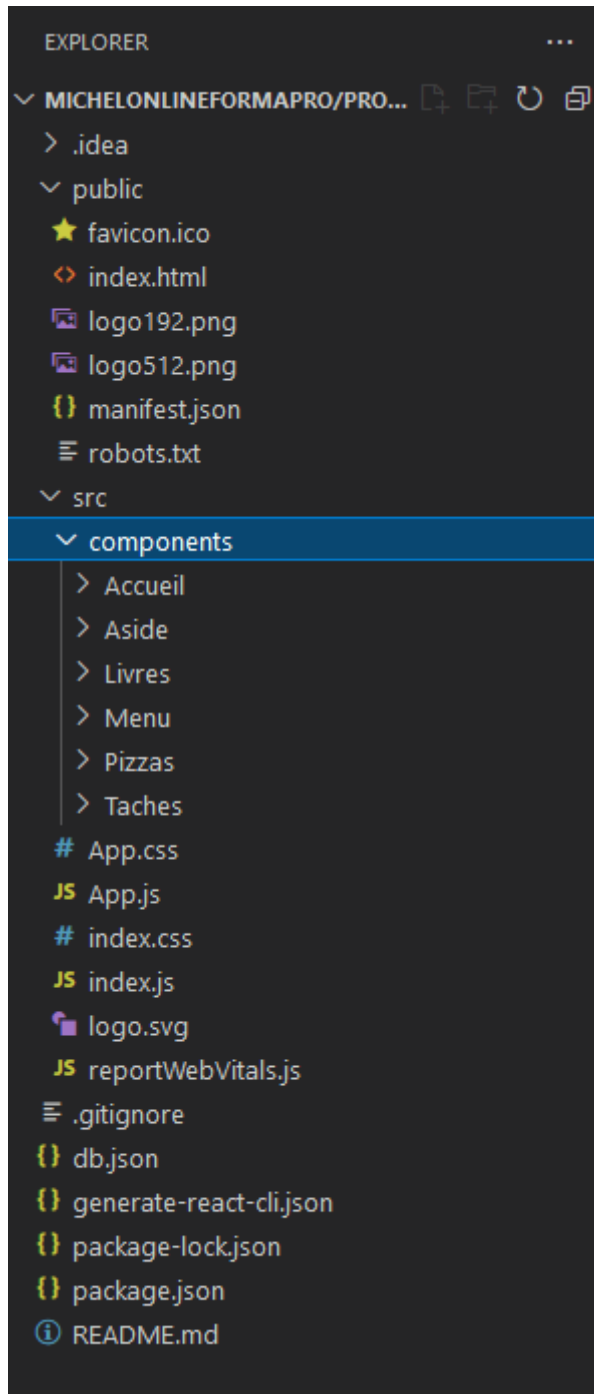
<https://eslint.org/>

CREER UN PROJET AVEC CRA :

Create React App va vous permettre de **générer un squelette de code** pour votre application. Il embarque un certain nombre d'**outils préconfigurés**, tels que Web pack, Babel et ESLint, afin de vous garantir la meilleure expérience de développement possible.

4. RAPPEL : <https://fr.reactjs.org/docs/hello-world.html>
5. IDE : Web Storm
6. Créer un projet : `npx create-react-app react-base`

LA STRUCTURE DU PROJET :



- `node_modules` : c'est là que sont installées toutes les **dépendances** de notre code. Ce dossier peut vite devenir très volumineux.
- `public` : dans ce dossier, vous trouverez votre **fichier `index.html`** et d'autres fichiers relatifs au référencement web de votre page.
- `src` : vous venez de rentrer dans le cœur de l'action. **L'essentiel des fichiers que vous créerez et modifierez seront là.**

Les fichiers importants :

- `package.json` : situé à la racine de votre projet, il vous permet de **gérer vos dépendances** (tous les outils permettant de construire votre projet), vos scripts qui peuvent être exécutés avec `yarn`, etc. Si vous examinez son contenu, vous pouvez voir des dépendances que vous connaissez : React et ReactDOM :
 - Vous y trouverez `react-scripts`, créé par Facebook, qui permet d'installer Webpack, Babel, ESLint et d'autres pour vous faciliter la vie ;
- Dans `/public`, vous trouvez `index.html`. Il s'agit du **Template de votre application**. Il y a plein de lignes de code, mais vous remarquez `<div id="root"></div>` ? Comme dans les chapitres précédents, nous allons y ancrer notre app React ;
- Dans `/src`, il y a `index.js` qui permet d'**initialiser notre app React** ;
- Et enfin, dans `/src`, vous trouvez `App.js` qui est **notre premier composant React**.
- Les développeurs utilisent des outils automatisés pour faciliter leur expérience de développement.
- Create React App (CRA) est la boîte à outils créée par Facebook, qui reste encore la référence pour initier un projet React.
- Un projet initialisé avec CRA possède toujours :
 - Un fichier `index.html` qui est le Template où vivra notre app React ;
 - Un `package.json` qui liste les dépendances et les scripts ;
 - Un fichier `index.js` dans lequel notre app React est initialisée, et greffée au HTML.
- CRA s'exécute avec l'aide d'un gestionnaire de paquet (dans ce cours, `yarn`).
- Webpack permet d'importer simplement les fichiers entre eux.

DU STYLE ET BULMA CSS

7. Installer le framework CSS : BULMA CSS via npm : `npm i bulma`
8. Importer le fichier css dans votre fichier de configuration `index.js`

```
view] \README.md JS index.js X
JS index.js
import React from 'react';
import ReactDOM from 'react-dom';
import './index.css';
import App from './App';
import reportWebVitals from './reportWebVitals';

import 'bulma/css/bulma.css';
import Menu from './components/Menu/Menu';
import 'animate.css';

ReactDOM.render(
  <React.StrictMode>
    <Menu />
    <App />
  </React.StrictMode>,
  document.getElementById('root')
);

// If you want to start measuring performance in your app, pass a function
// to log results (for example: reportWebVitals(console.log))
// or send to an analytics endpoint. Learn more: https://bit.ly/CRA-vitals
reportWebVitals();
```

9. Utilisé des classes bulma dans le JSX via className=""
10. En effet class="" est un mot clé JavaScript réservé on utilise donc className={}

VOTRE PREMIER COMPOSANT : LISTE DES LIVRES

11. Générer des composants : npx generate-react-cli component ListeLivres
12. Répondre au question posées
13. Pas de Type Script, pas de module css, créer un dossier composants + composant.js
14. Ajouter un fichier css à chaque composant
15. Pas de fichier test, history et lazy
16. Dans votre fichier ListeLivres.js :
 - a. Créer un objet state (état locale de vos variables)
 - b. Dans votre objet : Créer un tableau livres : []
 - c. Une chaîne de caractère livreID : ""
 - d. Une chaîne de caractère rechercher : ""

```
class LivresListe extends Component{  
  
  //Etat locale des données  
  state = {  
    //Tableau vide des livres  
    livres: [],  
    livreID: "",  
    rechercher: ""  
  }  
}
```

VOTRE FAUX BACKEND JSON-SERVER

17. Installer json-server : npm i json-server
18. Générer votre fichier db.json : json-server --watch db.json --port 3001
19. Ici React et json-server sont sur le même port locale 3000, on spécifie à json-server de démarrer sur le port 3001 grâce au drapeau (flags) --port
20. Modifier votre fichier db.json pour le transformer en une collection de Livres
21. Installer et importer le middleware Axios : npm i axios
22. Axios = Promesse basée sur http Client pour navigateur et NodeJs
23. <https://www.npmjs.com/package/axios>
24. <https://axios-http.com/>

```

db.json
{
  "livres": [
    {
      "id": 1,
      "nomLivre": "Livre javascript",
      "descriptionLivre": "JavaScript est un langage de programmation de scripts principalement employé dans les pages web interactives et à ce titre est une par",
      "prixLivre": 12.5,
      "imageLivre": "https://static.fnac-static.com/multimedia/Images/FR/NR/0d/4a/99/10045965/1507-1/tsp20181225081034/Tout-Javascript.jpg"
    },
    {
      "id": 2,
      "nomLivre": "Livre PHP",
      "descriptionLivre": "PHP: Hypertext Preprocessor, plus connu sous son sigle PHP, est un langage de programmation libre, principalement utilisé pour produire",
      "prixLivre": 45458.25,
      "imageLivre": "https://m.media-amazon.com/images/I/319uThTl0SL.jpg"
    },
    {
      "id": 3,
      "nomLivre": "Livre Python",
      "descriptionLivre": "Python est un langage de programmation interprété, multi-paradigme et multiplateformes. Il favorise la programmation impérative struct",
      "prixLivre": 142.35,
      "imageLivre": "https://www.dunod.com/sites/default/files/styles/principal_desktop/public/thumbnails/image/9782100804832-001-X.jpeg"
    },
    {
      "id": 4,
      "nomLivre": "Livre JAVA EE",
      "descriptionLivre": "Java SE Jakarta EE Java ME JavaFX Java Card Jakarta EE, est une spécification pour la plate-forme Java d'Oracle, destinée aux applicat",
      "prixLivre": 653.52,
      "imageLivre": "https://images-na.ssl-images-amazon.com/images/I/51vifBnNFzL.jpg"
    },
    {
      "nomLivre": "Livre jQuery",
      "descriptionLivre": "jQuery est une bibliothèque JavaScript libre et multiplateforme créée pour faciliter l'écriture de scripts côté client dans le code HT",
      "prixLivre": 45.25,
      "imageLivre": "https://images-na.ssl-images-amazon.com/images/I/518iL-gevyL.jpg",
      "id": 5
    },
    {
      "nomLivre": "Livre Unity",
      "descriptionLivre": "Unity est un moteur de jeu multiplateforme développé par Unity Technologies. Il est l'un des plus répandus dans l'industrie du jeu vid",
      "prixLivre": 35.25,
      "imageLivre": "https://www.d-booker.fr/399-thickbox/jeux-2d-et-multijoueurs-2e.jpg",
      "id": 6
    }
  ]
}

```

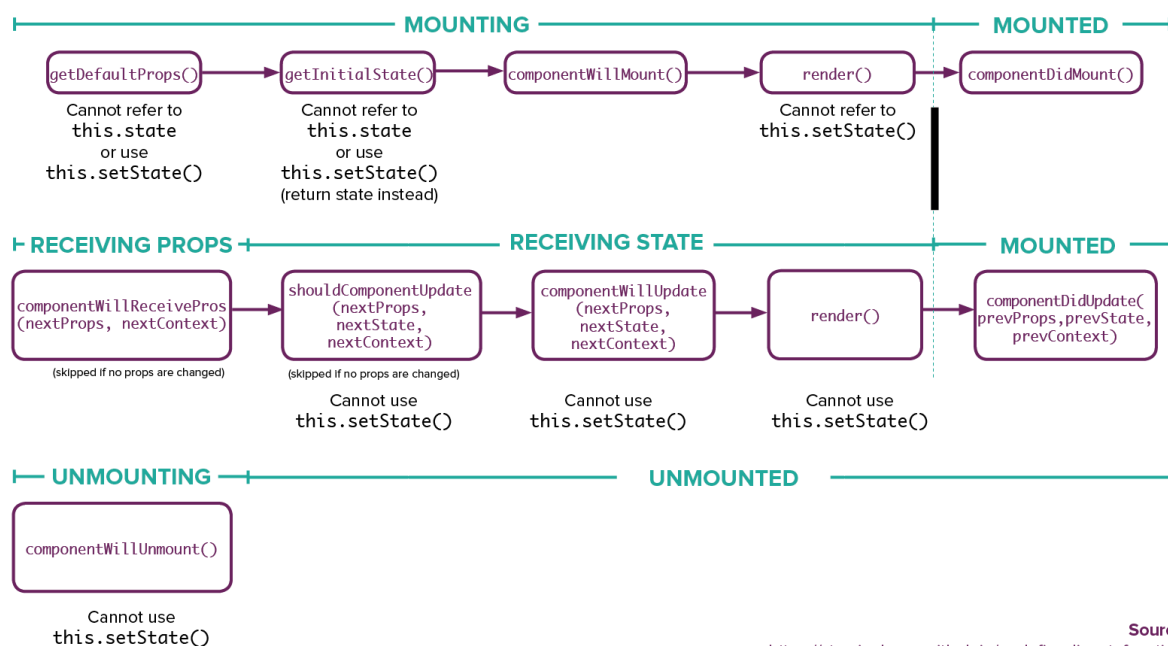
25. Créer une fonction `afficherLivres()`
26. Effectuer une requête http via axios méthode GET + URL
27. Créer une promesse
 - a. Dans votre résolve : replisser votre tableau de livre avec les données json
 - b. Modifier l'état local du tableau livres avec `setState`
 - c. Grace à ce dernier (setter ou mutateur), il remplit le tableau livres initialisé dans l'état local (state) avec les données de votre fichier db.json
 - d. Sinon (reject) déclencher une erreur et un debug

```

//Requête HTTP
afficherLivre = () => {
  //Appel axios et la methode get + URL
  axios.get('http://localhost:3001/livres')
    //Promesse
    .then(response => {
      const livres = response.data
      //Mise a jour du tableau rempli pa la requête HTTP
      this.setState({
        livres
      })
    })
  //Sinon une erreur
  .catch(err =>{
    console.log("Erreur de lecture du json livres " + err)
  })
}

```

28. Appeler cette fonction dans la fonction `componentDidMount()` (cycle de vie React après le 1^{er} rendu (`render()`) : le composant `ListeLivre` est monté après le 1^{er} `render()`)
 - i. Cette fonction sera donc déclenchée lors de l'affichage (`render()`) de la page.

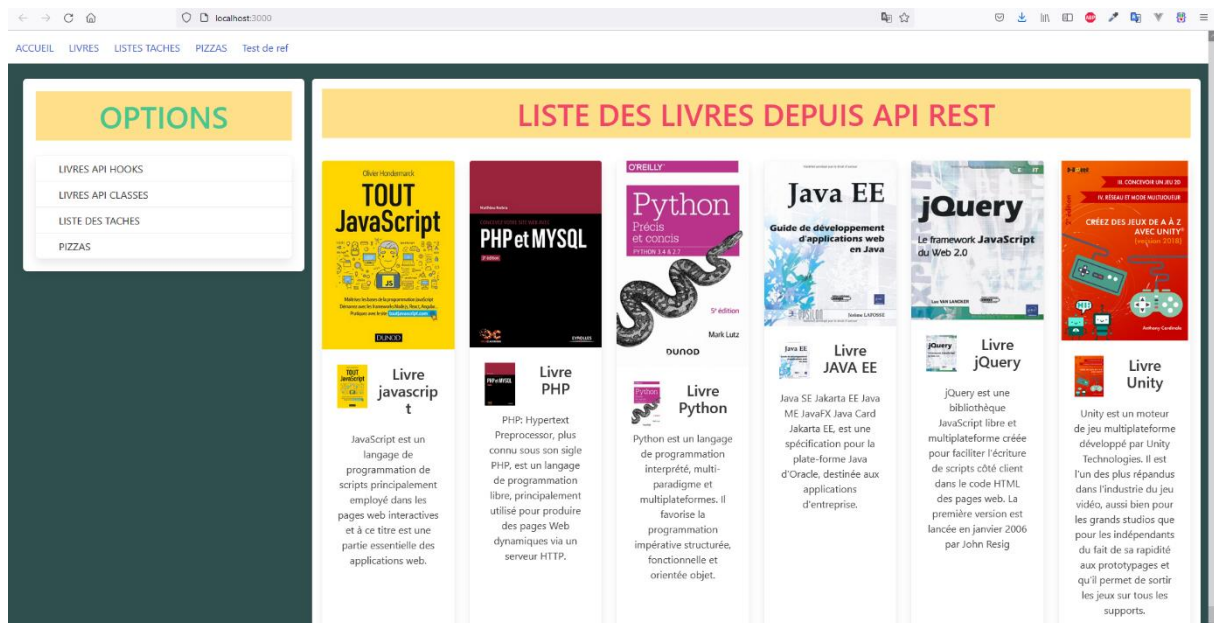


```
//Cycle de vie => apres render() le composant est monté on appelle la fonction afficher livres
componentDidMount() {
  this.afficherLivres();

  //Test click sur le bouton ajouter un livre
  $("#toggle-add-form").click(function () {
    $("#form-add-livre").toggle("slow");
  })
}
```

29. Le tableau livres désormais rempli des données json, on va donc boucler sur ce dernier à l'aide de la fonction JavaScript. `map()` dans le JSX.
30. <https://fr.reactjs.org/docs/lists-and-keys.html>
31. <https://openclassrooms.com/fr/courses/7008001-debutez-avec-react/7135593-gagnez-en-temps-et-en-efficacite-grace-aux-listes-et-aux-conditions>

```
<ul>
  {this.state.livres.map(livre =>
    <li key={livre.id}>
      <p>{livre.nomLivre}</p>
      <img src={livre.imageLivre} alt="Placeholder image" />
      <p className="subtitle is-4 has-text-info"><b>{livre.prixLivre} €</b></p>
      <p>{livre.descriptionLivre}</p>
    </li>
  )}
</ul>
```

AFFICHER LES DETAILS D'UN LIVRE :

32. Créer une fonction `livreById(id)` avec `id` en paramètre
33. Récupérer l'id et assigner ce dernier votre propriété locale (state) `livreID` à l'aide de la fonction `JavaScript filter()`
34. Exécuter une requête `http`, méthode `GET` + `URL` avec votre `id` en paramètre
35. Rappel : avec `json-server` ex : <http://localhost:3001/livres/2>
36. Créer une promesse et dans `réserve` : créer une constante `livreID = response.data`
37. Mettre à jour l'état locale de votre propriété `livreID` grâce à votre mutateur `setState`
38. Sinon retourner une erreur et un `debug` dans votre `reject` à l'aide de `catch`

```
livreById = id => {
  //Récupération de l'id
  const livreID = this.state.livres.filter(livre => livre.id !== id)

  //Requête HTTP get avec axios passage de id en paramètre
  axios.get('http://localhost:3001/livres/${id}')
    //Promesse
    .then(response => {
      //LivresId déclaré dans le state = au valeur de l'objet livre en fonction de son ID
      const livreID = response.data
      //Mise à jour de l'objet Livre
      this.setState({
        livreID
      })
      //Debug
      console.log(livreID)
    })
    //Sinon on affiche une erreur
    .catch(err => {
      console.log("Pas de livre pour cet ID " + err);
    })
}
```


39. Créer un bouton qui appelle votre fonction `livreById(livre.id)` (le paramètre est accessible à l'aide de la boucle `map()`)

```
<div onClick={() => this.livreById(livre.id)} id="card-content" className="column is-2" key={livre.id}>
```

40. Afficher les détails du livre dans le `JSX`
41. A l'aide d'un `ternaire` : si `livreID` est `!= null` : on affiche un `block`, sinon on affiche tous les livres

42. RAPPEL : condition ? true : false

```
/*TERNAIRE = block a afficher au click sur un livre si l'i existe et possède des données*/
{this.state.livreID ? (
  <div className="animate _animated animate _slideInRight">
    <div className="title is-1 has-text-success has-text-centered">DÉTAILS DU LIVRE {this.state.livreID.nomLivre}</div>
    <div className="has-text-centered">
      <img src={this.state.livreID.imageLivre} alt={this.state.livreID.nomLivre}/>
    </div>
    <div className="media-content">
      <p id={prix} className="title is-4">PRIX :</p>
      <p className="subtitle is-4 has-text-info "><b>{this.state.livreID.prixLivre} €</b></p>
    </div>
    <div className="content">
      {this.state.livreID.descriptionLivre}
    </div>
    <div>
      {/*On recharge la page pour le retour car url est la meme : http://localhost:3000/livres*/}
      <button onClick={() => window.location.reload()} className="button is-info">RETOUR</button>
      {/*Appel de la fonction handleDelete depuis une fonction anonyme + passage de l'id du livre en paramètres*/}
      <button onClick={() => this.handleDelete(this.state.livreID.id)} className="button is-danger">SUPPRIMER {this.state.livres.nomLivre}</button>
    </div>
  </div>
) : (
  <div>
```



SUPPRIMER UN LIVRE

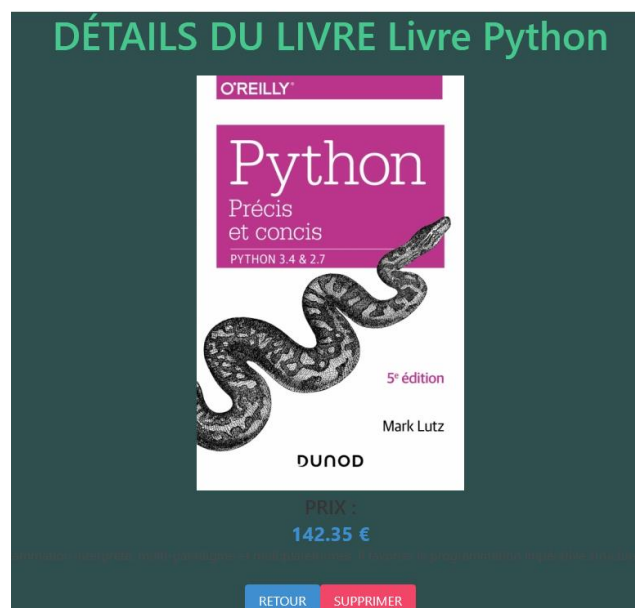
43. Créer une fonction handleDelete(id) avec id en paramètre
44. Comme pour les détails récupérer l'id et assigner ce dernier a la propriété locale (state) livreID
45. Mettre à jour votre livreID locale grâce au mutateur setState()
46. Créer votre requête http + méthode DELETE + URL et votre id
47. TEST POSTMAN : ex : <http://localhost:3001/livres/2>
48. Créer une promesse qui retourne une alerte JavaScript pour confirmer la suppression de votre livre et recharger la page en cas de réussite
49. Sinon retourner une erreur et un debug

```
//Fonction pour supprimer un livre depuis le bouton supprimer
//<button onClick={() =>this.handleDelete(this.state.livreID.id)} className="button is-danger">SUPPRIMER {this.state.livres.nomLivre}</button
handleDelete = (id) => {
  //Recuperation de id grace a js Filter
  //on recup livreID: "" depuis l'etat locale (state) et filter variable + fonction var.id + paramètre de la fonction (ici id recup dans le JSX)
  const livreID = this.state.livres.filter(livre => livre.id !== id)
  //Debug
  console.log(livreID)
  //Mise a jour de l'etat locale grace au mutateur (setter) setState
  this.setState({
    livreID
  })

  //Requete HTTP axios methode = delete + concaténation de id (passer en paramètre de la fonction)
  axios.delete(`http://localhost:3001/livres/${id}`)
  //Promesse
  .then(response => {
    //Debug f12
    console.log(response.data)
    //On declenche une alerte pour confirmer la suppression ou annuler
    alert('Confirmer la suppression du livre : ${livreID} ?');
    //On refresh la page
    window.location.reload();
  })
  //Sinon on declenche une erreur
  .catch(err => {
    console.log("Erreur de suppression du livre " + err)
  })
}
```

50. Appeler la fonction handleDelete() dans votre bloc JSX dans la condition ou livreID est !== null

```
<div>
  { /*On recharge la page pour le retour car url est la meme : http://localhost:3000/livres*/ }
  <button onClick={() => window.location.reload()} className={"button is-info"}>RETOUR</button>
  { /*Appel de la fonction handleDelete depuis une fonction anonyme + passage de l'id du livre en paramètres*/ }
  <button onClick={() =>this.handleDelete(this.state.livreID.id)} className="button is-danger">SUPPRIMER {this.state.livres.nomLivre}</button>
</div>
```



AJOUTER UN LIVRE

51. Créer un composant AjouterLivre.js

52. Dans votre état locale (state) créer un objet avec les propriétés suivantes

```

import React, {Component} from "react";
import './Livres.css';
import axios from "axios";

class AjouterLivres extends Component{

  constructor(props) {
    super(props);

    //Etat locale des données d'un livre
    this.state = {
      livres:[],
      id: null,
      nomLivre : "",
      descriptionLivre: "",
      prixLivre: "",
      imageLivre: ""
    }
    //Garder le context de la fonction
    this.handleChange = this.handleChange.bind(this)
  }
}

```

53. Créer 2 fonctions : handleChange(event) et handleSubmit(event)

```

//A chaque entrée dans le formulaire = on met a jour les valeurs en récupérant <input name = input value
//Detecter les changements d' etat de chaque input
handleChange = (event) => {
  this.setState({
    //Recuperation des tous les <input name="" === input value=""/>
    [event.target.name]: event.target.value,
    id: event.target.value
  })
  //Debug
  //Test de recup des inputs a checker dans f12 console
  console.log(this.state.nomLivre)
  console.log(this.state.descriptionLivre)
  console.log(this.state.prixLivre)
  console.log(this.state.imageLivre)
}

```

```

//Soumission du formulaire = dans jsx <form onSubmit={this.handleSubmit}
handleSubmit = (event) => {
  //On evite le rechargement de la page
  event.preventDefault()
  //Requête HTTP avec axios + passage d'option
  //Requête http methode post avec axios
  axios.post("http://localhost:3001/livres",{
    //Recup des valeurs des inputs mis a jour par handleChange
    nomLivre: this.state.nomLivre,
    descriptionLivre: this.state.descriptionLivre,
    prixLivre: this.state.prixLivre,
    imageLivre: this.state.imageLivre
  })
  //Promesse
  //Creation d'une promesse (resolve ou reject)
  .then(response => {
    //On stock les valeurs du formulaire dans une constante
    const livres = response.data
    console.log(livres)
    //Modifier l'etat locale des données (on remplit le tableau livres[] declarer dans le state
    this.setState({
      livres
    })
    //On recharge la page
    window.location.reload()
  })
  .catch(err => {
    console.log("Erreur lors de l'ajout du livres " + err)
  })
}
}

```

54. Créer un formulaire pour ajouter vos livres
55. Chaque input appelle la fonction handleChange() grâce à l'attribut onChange={this.handleChange}
56. La fonction handleSubmit() est appelée dans l'attribut onSubmit={handleSubmit} dans la balise <form>

```

//Formulaire d'ajout du livre
//handleChange recupere les entrées clavier de input grace au mutateur setState et input name === input value
render() {
  return(
    <div className="container">
      <div className="title is-1 has-text-centered has-text-success">LE FORMULAIRE AJOUT DE LIVRE</div>
      <form onSubmit={this.handleSubmit}>
        <div className="field">
          <label className="label">Nom du livre</label>
          <input
            type="text"
            placeholder="Nom du livre"
            className="input"
            required
            name="nomLivre"
            onChange={this.handleChange}
          />
        </div>
        <div className="field">
          <label className="label">Description du livre</label>
          <textarea
            id="descriptionLivre"
            name="descriptionLivre"
            className="textarea"
            required
            onChange={this.handleChange}
            placeholder="Livre de cours et exercice PHP"
          />
        </div>
        <div className="field">
          <label className="label">Prix du livre</label>
          <input
            id="prixLivre"
            name="prixLivre"
            className="input"
            type="number"
            step="0.01"
            required
            onChange={this.handleChange}
            placeholder="25.25"
          />
        </div>
        <div className="field">
          <label className="label">Image du livre</label>
          <input
            id="imageLivre"
            name="imageLivre"
            className="input"
            type="text"
            required
            onChange={this.handleChange}
            placeholder="https://www.dunod.com/sites/default/files/styles/principal_desktop/public/thumbnails/image/9782100743209-001-X.jpeg"
          />
        </div>
        <div>
          <button type="submit" className="button is-success">Valider les données du formulaire</button>
        </div>
      </form>
    </div>
  )
}

```

57. Ajouter votre composant dans le JSX (avec un import en haut de page)

```

<div id="form-add-livre" className="mt-3 box">
  <AjouterLivres />
</div>

```

UNE BARRE DE RECHERCHE PAR TITRE DE LIVRE

58. Créer une fonction handleRechercher(event)
59. Ajouter un debug : console.log("Valeur du champ de recherche", event.target.value)
60. Mettre à jour l'état locale de la propriété rechercher: "" du state

```
//Fonction rechercher par titre
handleRechercher = (event) => {
  //Debug du champ input = value
  console.log("Valeur du champ de recherche", event.target.value)
  //Mettre a jour l' etat locale de la propriété rechercher:"" du state
  this.setState({
    rechercher: event.target.value
  })
}
```

61. Créer un input de type texte dans votre JSX
62. <input/> prend pour valeur votre state rechercher et détecte le changement d'état avec onChange={handleChange} + this et bind pour garder le contexte

```
{/*BARRE DE RECHERCHE*/}
{/*Appel de la fonction handleRechercher dans onChange qui recupère la valeur du champ input target value grace au setState */}

<div className="mt-3 field box">
  <label className="label">RECHERCHER</label>
  <input
    className="input"
    placeholder="php"
    type="text"
    name="rechercher"
    onChange={this.handleRechercher.bind(this)}
    value={this.rechercher}
  />
</div>
```

63. Faite des tests de debug en entrant des lettres dans votre champ

64. Avant votre boucle map() ajouter des paramètres de filtre et des éléments inclus à vos résultats via les fonctions filter() et includes() de JavaScript
65. {this.state.livres.filter(recherche => recherche.nomLivres.toLowerCase().includes(this.state.rechercher.toLowerCase()))}.map(livre

Tableau des livres . Filtre (nom des livres en minuscule). Qui sont inclus (entrées de la barre de recherche en minuscule). Boucle des résultats

```

    {this.state.livres.filter(recherche => recherche.nomLivre.toLowerCase().includes(this.state.rechercher.toLowerCase())).map(livre =>
      <div onClick={() => this.livreById(livre.id)} id="card-content" className="column is-2" key={livre.id}>
        <div className="card">
          <div className="p-3 title is-4 has-text-centered has-text-danger">{livre.nomLivre}</div>
          <div className="card-image p-3">
            <figure className="image">
              { /* eslint-disable-next-line jsx-a11y/img-redundant-alt */ }
              <img src={livre.imageLivre} alt="Placeholder image" />
            </figure>
          </div>
          <div className="card-content">
            <div className="media">
              <div className="media-left">
                <figure className="image is-48x48">
                  <img src={livre.imageLivre} alt="Placeholder image" />
                </figure>
              </div>
              <div className="media-content">
                <p id={"prix"} className="title is-4">PRIX :</p>
                <p className="subtitle is-4 has-text-info"><b>{livre.prixLivre} €</b></p>
              </div>
            </div>
            <div className="content">
              {livre.descriptionLivre}
            </div>
          </div>
        </div>
      </div>
    )}
  )}

```

RESULTAT POUR LA LETTRE "J" :

LES + :

1. Réalisé le même exercice avec NodeJs – Express – Mongo DB à la place de json-server
2. Ajouter un système de routing avec React Router DOM :
3. <https://reactrouter.com/web/guides/quick-start>
4. Remplace vos classes par des hooks
5. <https://fr.reactjs.org/docs/hooks-intro.html>
6. Découper vos composants et passer des données et fonctions à l'aide des props
7. Ajouter redux pour gérer l'état de vos rendus

