React

Mohamed ROMDANE

Septembre 2021

formation@ineotec.com

Présentation du cours

- Présentation
- Conseils pour suivre la formation
- Le parcours de formation

Formateur

- **Mohamed Romdane est expert spécialisé en architecture Big Data. Au sein de Ineotec, il offre des solutions d'audit de base de données, code review, analyse de logs et développement de logiciels dans le monde open source et Full-Stack notamment TypeScript-React côté front-end.**
- **Mohamed Romdane** est diplômé en électronique de l'**Université de Paris XI** avec un DEA sur le **traitement de l'information** entre l'*Institut d'Électronique Fondamentale* et l'*Institut National des Sciences et Techniques Nucléaires*, puis en thèse de doctorat au *Laboratoire de Recherche en Informatique* sur les **bases de données déductives**.
- Il travaillera à l'*Institut Universitaire Technique - IUT Sceaux* et au *CNAM Saclay* en tant qu'enseignant en **Bases de données** avant de s'attaquer au milieu professionnel avec une expérience avec plusieurs entreprises, institutions et groupes à une échelle internationale: *Talan France*, *DGI Maroc*, *Gide*, *SMAI* en France et en Nouvelle-Calédonie, au *Ministère des Finances* en Polynésie française, *France Telecom*, *Sagep*, *Bouygues*, *Star* et *Sofrecom*.
  Les compétences et l'expertise de Mohamed Romdane au sein de Ineotec vont de la ***Business Intelligence*** - ***Enterprise Service and Bus*** – ***Extract Transfer and Loading*** (Datastage, …) aux **bases de données (Oracle, SQL Server, Sybase, …)** , en passant par **JAVA** et **JEE**, les **solutions mobiles**, le **php**, plusieurs **langages de programmation**, **produits client/serveur** et **systèmes d'exploitation**.

Ce que vous apprendrez

- Créer une application React complète à partir d'un dossier vide
- Développer un système de navigation entre composants
- Ajouter des formulaires pour interagir avec l'utilisateur
- Effectuer des requêtes HTTP depuis son application
- Mettre en place un système d'authentification
- Déployer une application React en production

React

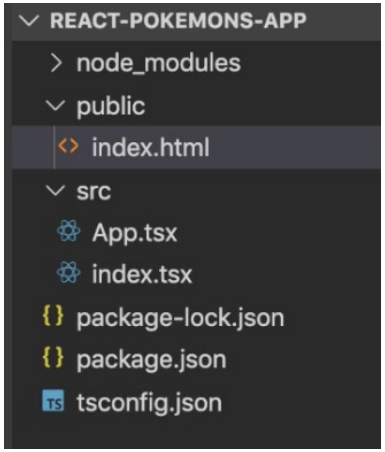
- Une bibliothèque JavaScript pour créer des interfaces utilisateurs
- Déclaratif
- À base de composants
- Utilisable partout

- React peut être utilisé côté serveur avec Node, ou pour créer des applications mobiles grâce à React Native.

Présentation React

- Les sites web deviennent de plus en plus de véritables applications, et une utilisation intensive du langage JavaScript devient nécessaire.
- React est une bibliothèque orienté composants, votre application entière est un assemblage de composants contrairement à Angular qui est un Framework.
- Les quatre éléments à la base de toute application React sont : les composants, les *props*, le DOM virtuel et la syntaxe JSX.
- React est conçu pour le web de demain et intègre déjà la norme ECMAScript6 (ES6) et les Web Components.

Premier pas



- On a besoin au minimum d'un composant par application.
- Le fichier *index.tsx* fait le lien entre notre composant racine, et le fichier *index.html*.
- L'ordre de chargement de l'application est le suivant : *index.html* > *index.tsx* > *App.tsx*.
- Dans une application React, l'extension des fichiers TypeScript est *tsx*, car nous utilisons JSX.
- Le fichier de configuration du compilateur de TypeScript se nomme *tsconfig.json* par convention.
- Le fichier *package.json* initial est fourni avec des commandes prêtes à l'emploi comme la commande *npm start*, qui nous permet de démarrer notre application web.

Les Composants

- Il existe deux types de composants en React

```
/*
import React from 'react';

export default class App extends React.Component {
  const name: string = "React";

  render() {
    return <h1>Hello, {name}</h1>;
  }
}
*/
```

Les composants de classes

```
App.tsx > App
import React, { FunctionComponent } from 'react';

const App: FunctionComponent = () => {
 const name: String = 'React';

 return (
  <h1>Hello, {name} !</h1>
 )
}

export default App;
```

Les composants de fonctions

Composant Fonction Vs Classe

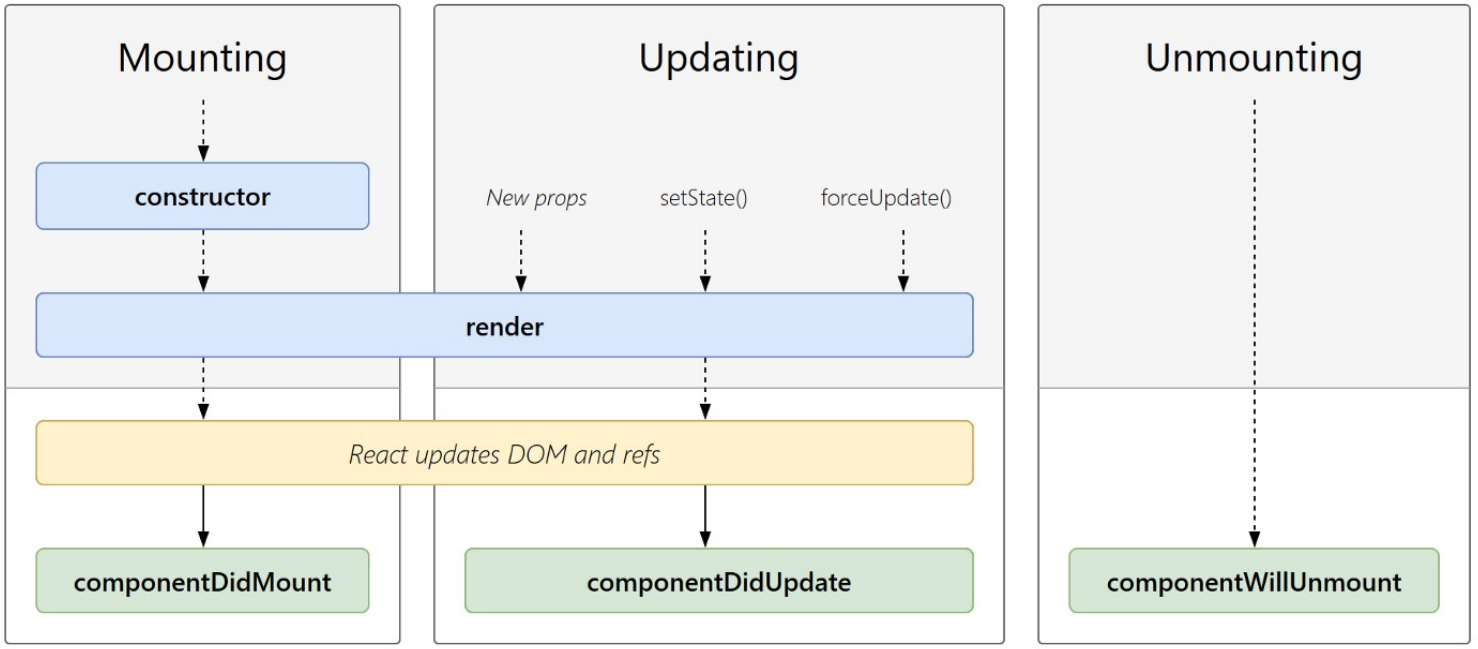
Composant de fonction	Composant de classe
– Plus performant. – Plus concis. – Pas de gestion du *state*. – Pas de gestion du cycle de vie du composant.	– Gestion du *state*. – Gestion du cycle de vie du composant. – Moins performants. – Plus long à écrire.

- Il est recommandé d'utiliser les composants de fonctions.
- Un composant React peut prendre en entrée une ou plusieurs *props*, qui sont l'équivalent des attributs en HTML.
- Un composant React retourne une portion de DOM virtuel.
- Le DOM virtuel est décrit avec la syntaxe JSX, car c'est la méthode la plus recommandée.

Les Hooks

- Les composants de Classe comportent la notion de State et de Cycle de vie de l'objet contrairement aux composants de fonctions.
- Les Hooks permettent de pallier aux limites aux composants de fonction, en leur ajoutant des fonctionnalités supplémentaires.
- Le Hook d'état permet de définir un *state* à nos composants.
- Le Hook d'effet permet d'intervenir sur le cycle de vie de nos composants.

Hooks Cycle de Vie



Le DOM Virtuel avec JSX

- La syntaxe {} de JSX permet d'interpréter du code JavaScript dans vos templates, ce qui permet notamment d'afficher les données de nos composants dans les templates.
- On utilise une condition avec l'opérateur logique « `&&` » pour conditionner l'affichage d'un template.
- On utilise l'opérateur ternaire pour afficher une portion ou une autre d'un template.
- `condition ?  true : false`
- La méthode JavaScript ***map*** permet d'afficher un tableau de données dans un template.
- Lorsqu'on affiche une liste dans du code JSX, il est important d'appliquer la propriété ***key***, sous peine de lever une erreur.
- On peut gérer les interactions d'un utilisateur avec les événements React.
- La syntaxe des événements React n'est jamais identique à celle des événements du DOM.
- Exemple: *onclick* devient *onClick* en React.
- On peut récupérer l'événement natif du DOM dans un gestionnaire d'événement, à condition de le passer en dernier paramètre.

Les props

- Les *props* sont le moyen le plus simple de faire communiquer deux composants entre eux.
- On passe des *props* depuis un composant parent vers un composant fils.
- Les *props* permettent de mieux découper notre application en composants plus petits et plus autonomes.
- On peut ajouter un type à nos *props*, afin de s'assurer que la nature des données reçues par le composant fils.
- On peut définir des *props* comme facultatives. Il faut alors associer une valeur par défaut à ces props.
- Les *props* permettent beaucoup de souplesse quant au fonctionnement de nos composants.

Les Hooks Personnalisés

- Les propriétés calculées sont simplement des fonctions qui permettent d'effectuer un traitement sur les données d'un composant avant de les afficher à l'utilisateur.
- Il est recommandé de factoriser la logique commune à plusieurs composants dans une fonction à part.
- Si la logique à factoriser implique d'utiliser les Hooks d'un composant, *useState* et *useEffect* par exemple, alors il faut utiliser un Hook personnalisé.
- Un Hook personnalisé est une fonction JavaScript, dont le nom commence par "*use*" , et qui peut appeler d'autre Hooks.
- Un Hook personnalisé peut ensuite être utilisé dans plusieurs composants.

Les Routes

- React utilise *react-router-dom* pour mettre en place un système de navigation.
- React simule la navigation de l'utilisateur auprès du navigateur, sans que nous n'ayons rien à faire.
- On construit un système de navigation en associant une *url* et un *composant*.
- Le routeur de React interprète les routes du haut vers le bas. Il faut donc être prudent quant à l'ordre de déclaration des routes.
- Il est possible de faire passer des paramètres depuis les urls vers un composant, et de typer ces paramètres avec TypeScript.
- La balise <*Switch*> permet d'injecter le template d'un composant en fonction de l'url demandée par l'utilisateur.
- La balise <*Route*> permet de définir une route au sein de notre application.
- Pour gérer les erreurs 404 dans notre application, il faut déclarer en dernier un élément *Route* qui ne prend pas de chemin. Ainsi, toutes les urls seront interceptés par cet élément.

Les Formulaires

- Il y a deux manières différentes de développer des formulaires avec React: avec les composants contrôlés, ou avec les composants non-contrôlés.
- React recommande fortement d'utiliser les composants contrôlés car ils sont beaucoup moins limités.
- On utilise l'état d'un composant pour sauvegarder la structure et l'état de notre formulaire.
- À chaque modification du formulaire de la part de l'utilisateur, il faut prévoir un traitement spécifique. Autrement dit, il est nécessaire de définir un gestionnaire d'événement pour chacun de nos champs.
- On doit appliquer des règles de validation sur un formulaire par l'utilisation d'expressions régulières.
- Si un champ n'est pas valide, il faut prévoir un indicateur visuel à afficher à l'utilisateur. Ce sera plus agréable pour lui, de remplir le formulaire en cas d'erreur.
- Il faut toujours effectuer une validation côté serveur en complément de la validation côté client, si vous avez prévu de stocker des données depuis votre application.

Effectuer des requêtes HTTP

- Nous avons utilisé l'API Fetch pour effectuer des requêtes HTTP sur le réseau. Il existe bien sûr d'autres clients HTTP.
- Une requête HTTP est constitué d'une url à appeler, et éventuellement d'une en-tête et d'un corps.
- Il est possible de mettre en place une API Rest de démonstration au sein de votre application. Cela vous permettra d'interagir avec un jeu de données configuré à l'avance.
- Il est rapidement indispensable de maîtriser les quatre opérations de base nécessaire pour interagir avec votre API Rest : ajout, récupération, modification et suppression.
- Les quatre opérations de base pour interagir avec une API Rest sont appelées les opérations CRUD.

Un Système d'Autocomplétion

- Les API Rest mettent parfois à notre disposition des urls plus avancées que juste les requêtes classiques.
- La souplesse permise par React à travers les composants, le *state* et les services, permet de développer toutes les fonctionnalités.
- Les API Rest ne renvoient jamais un résultat instantanément, la requête envoyé au serveur prend toujours un certains temps avant de retourner une réponse.
- Il faut donc prévoir que notre application n'aura pas toujours des données à disposition immédiatement, et anticiper le temps d'attente des requêtes.

L’Authentification

- L’authentification permet de restreindre l’accès à certaines fonctionnalités de notre application.
- Par défaut, toutes les routes de notre application sont publiques. Il faut bien faire attention à cela.
- Le formulaire d’authentification est un formulaire comme les autres.
- Il faut sauvegarder le fait que l’utilisateur est connecté ou non dans notre application. Dans notre cas, on a stocké cette information dans un service.
- L’authentification nécessite la mise en place d’un système fiable : on utilise pour cela un service dédié et un composant *PrivateRoute*.

Le Déploiement

- Le déploiement est une étape dans un projet qui consiste à faire fonctionner une application sur l'environnement de production.
- Avant de déployer un projet local sur une machine de développement, il est nécessaire d'effectuer quelques étapes.
- React propose une commande permettant de générer une version du projet spécialement optimisée pour l'environnement de production.
- Firebase propose un utilitaire en ligne de commande nommé *Firebase CLI*, afin de déployer en ligne facilement des applications web statiques. Cela fonctionne aussi bien pour les applications qui ne sont pas développées avec React.

ES5 -> 6

- JavaScript profite de la nouvelle spécification standardisée *ECMAScript 6*, également nommée *ES6*.
- On peut développer en *ES6* dès aujourd'hui, et utiliser un transpilateur pour convertir notre code d'*ES6* vers *ES5*, afin qu'il soit compréhensible par tous les navigateurs.
- *ES6* nous permet d'utiliser les classes et l'héritage en JavaScript.
- *ES6* introduit deux nouveaux mots-clés : *let* et *const*. Le premier permet de déclarer des variables et tend à remplacer *var*, et *const* permet de déclarer des constantes.
- Les Promesses offrent une syntaxe plus efficace que les *callbacks*, et tendent à les remplacer, surtout pour les développements relatifs à la programmation asynchrone.

Débogage

- Maintenant que vous savez développer une application avec React, vous aurez sûrement envie de développer vos propres applications.
- Pour vous faciliter la tâche, l'éco-système de React propose un outil pratique pour faciliter vos développements.
- *React Developer Tools*
- L'objectif de cet outil est de vous permettre de mieux comprendre le fonctionnement de vos application lors du développent.
- Plus besoin de placer les fameux *console.log* partout dans votre code, pour savoir si tel ou tel donnée à été prise en compte dans le *state* de votre composant, ou la valeur de telle *prop* en entrée d'un composant.
- La solution est donc d'installer l'extension *React Developer Tools* dans votre navigateur Chrome

Urlographie

- https://fr.reactjs.org/
- https://www.typescriptlang.org/
- **Editeurs en ligne**
- https://www.typescriptlang.org/play
- https://codepen.io
- https://babeljs.io/repl
- https://materializecss.com/

Logiciels à installer

- Installation Visual Studio Code
  - https://code.visualstudio.com/
  - Eslint
  - Material Icon
  - Path Intellisense
- Installation node.js
  - https://nodejs.org
- Installation Typescript
  - http://www.typescriptlang.org
  - npm install –global typescript ou npm i –g typescript

Courses

- Glitch: React Starter Kit - 5-part video course with interactive code examples that will help you learn React.
- https://blog.glitch.com/post/react-starter-kit
- Codecademy: React 101 - Codecademy's introductory course for React.
- https://www.codecademy.com/learn/react-101
- React Bootcamp - Recordings from three days of a free online React bootcamp.
- https://ui.dev/react/
- Scrimba: Learn React - 48 hands-on video tutorials building react apps.
- https://scrimba.com/learn/learnreact
- University of Helsinki: Full Stack Open MOOC - Learn to build web applications with React.
- https://fullstackopen.com/en/

Contenu des travaux pratiques

- Hello World avec React
- Les Composants
- Le DOM virtuel avec JSX
- Les Props
- Les Hooks personnalisés
- Les Routes
- Les Formulaires
- Effectuer des requêtes HTTP
- Un système d’autocomplétion
- Authentification
- Déployer votre application