 **Initiation à Reactjs**

Dans ce tutoriel nous allons voir comment construire une application React en utilisant Typescript.

Je vais vous guider tout au long ce tutoriel étape par étape de la façon la plus simple et intelligente de développer une application React.

Nous verrons le concept React (TSX, components, props, state et lifecycle) à travers d’un exemple d’une application de gestion de contact.

Pour ce tuto nous allons utiliser l’api « **localStorage »** qui permet de stocker les données dans le navigateur sous forme de session (voir : <https://developer.mozilla.org/fr/docs/Web/API/Storage>)

**Les prérequis :**

Pour comprendre ce tuto, vous aurez besoin d'une connaissance de base en HTML, CSS et JavaScript.

**React : c’est quoi ?**

Reactjs est une librairie libre créée par Facebook en 2013 pour faciliter la création d’application monopage (SPA : Single Page Application). React n’est pas un vrai framework MVC (Model View Controller) comme Angular. Il fournit seulement la vue (V) du MVC.

Par défaut React utilise le langage javascript/ES6 (<http://www.ecma-international.org/ecma-262/6.0/>) pour développer un code orienté objet. Il peut être également utilisé avec le langage Typescript (<https://www.typescriptlang.org/docs/home.html>) qui est un superset de javascript.

React se démarque des autres concurrents sur beaucoup de points mais surtout au niveau de la performance car React utilise un virtual DOM (<https://la-cascade.io/comprendre-le-virtual-dom/>) basé sur l’algorithme « diffing » pour mettre efficacement à jour l’interface utilisateur (UI).

(<https://reactjs.org/docs/reconciliation.html>)

**Qui utilise React ?**

Facebook a développé pour leurs applications plus 50 milles composant en utilisant React. Bon nombre d’entreprise come **Twitter**, **Airbnb**, **Uber**, **Netflix**, **Instagram** etc… utilisent React.

Ceci explique la souplesse, la simplicité et surtout la performance que les entreprises ont optés pour développer leur application en React.

**Préparation de l'environnement de développement :**

Certaine préparation est requise pour le développement du React.

**Installation Node.js**

Le développement React se base sur cet outil.

Node.js a été créé en 2009 par Ryan Dahl, est une plate-forme libre en javascript orienté serveur.

Il est basé sur le moteur Javascript V8 libre qui est développé par Google.

(<https://nodejs.org/en/>)

Tester la version installée :

$ node -v

*$ : votre terminal de commande favorite*

Si tout s’est bien passé vous devez voire afficher la version du node, dans mon cas c’est la version suivante :

v10.16.3

Si ce n’est pas le cas, veuillez vérifier que Node est bien configuré dans le chemin (PATH) d’environnement de votre système.

Tester NPM :

$ npm -v

Dans mon cas c’est la version :

6.9.0

*Note* : NPM (Node Package Manager) est un gestionnaire de paquets pour Node.js et depuis la version 0.6.3 npm fait partie du Node.js est installé par défaut.

(<https://docs.npmjs.com/>)

**Installation Create-React-App**

Create-React-App est un outil standard qui facilite le développement d’application React.

Ce dernier permet de générer automatiquement la structure du projet et fournit bon nombre de scripts pour la gestion de l’application.

(<https://github.com/facebook/create-react-app>)

Pour l’installer :

$ npm install –global create-react-app

*Note* : Si pour quelque raison le npm n’est pas installé, vous avez toujours la possibilité de faire manuellement :

$ npm install --global npm

**Installation Git**

Git est nécessaire pour gérer les packages qui sont utilisés pour le développement React.

Pour l’installer, rendez sur le <https://git-scm.com/> et en fonction de votre système d’exploitation suivez les instructions. Dans mon cas, j’utilise mac os et la commande pour mac os est :

$ sudo apt-get install git

ou

$ brew install git

Vérification de la version :

$ git --version

**Installation Visual Studio Code**

VS Code est un éditeur de code développé par Microsoft.

(<https://code.visualstudio.com/>)

Pour ce tuto, nous allons utiliser cet éditeur de code pour développer notre application React.

Vous pouvez si vous le souhaitez, utiliser un autre éditeur de votre choix

**Projet création :**

Notre environnement de développement est prêt, on peut commencer à créer notre premier projet, pour cela, nous allons utiliser la commande suivante :

$ npx create-react-app react-typescript-starter --typescript

La commande **npx** fait partie de l’outil node/npm. Il est utilisé pour lancer les packages Node.js

*Note* : S’il est n’est dispo dans l’outil Node.js, installez en faisant :

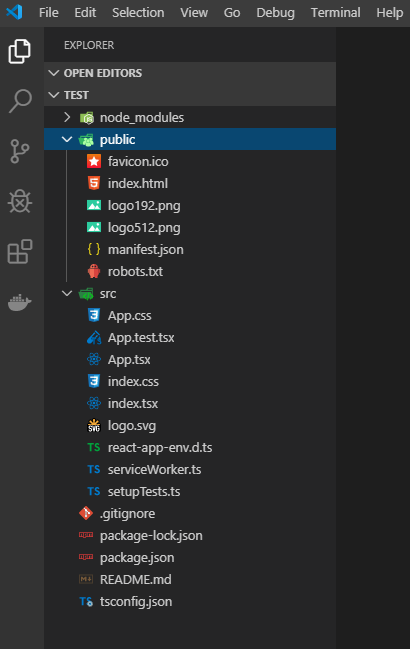
$ npm install --global npx

Le premier argument de la commande npx est le nom du package à lancer, ici c’est le package create-react-app, le deuxième argument est le nom du projet « react-typescript-starter » et le dernier argument est un flag qui indique l’utilisation du Typescript par défaut à la place du Javascript.

La génération du projet prend quelques minutes.

**Structure du projet :**

Ouvrez le dossier « react-typescript-starter » crée par l’outil create-react-app dans l’éditeur vs code.



|  |  |
| --- | --- |
| **Name** | **Description** |
| public/index.html | C’est le fichier qui est chargé dans le navigateur lorsque lancer l’application. L’élément « <div id="root"></div> » dans lequel le code React est exécuté. L’id « root » est utilisé par le fichier src.index.tsx pour le rendu html et javascript.  Plus d’info : <https://github.com/jantimon/html-webpack-plugin#options> |
| src/index.tsx | Le fichier typescript qui est responsable de démarrer l’application React. Il utilise le dom id « root » qui se trouve dans le fichier public/index.html |
| src/App.tsx | C’est le composant root qui contient le code html et typescript afficher. Il peut contenir d’autres composant React. |
| package.json | Le fichier de config pour la gestion des packages et dans celui-ci vous trouverez les scripts par défaut pour lancer l’application en local, publication, etc… |

**Lancer l’application :**

Pour tester l’application

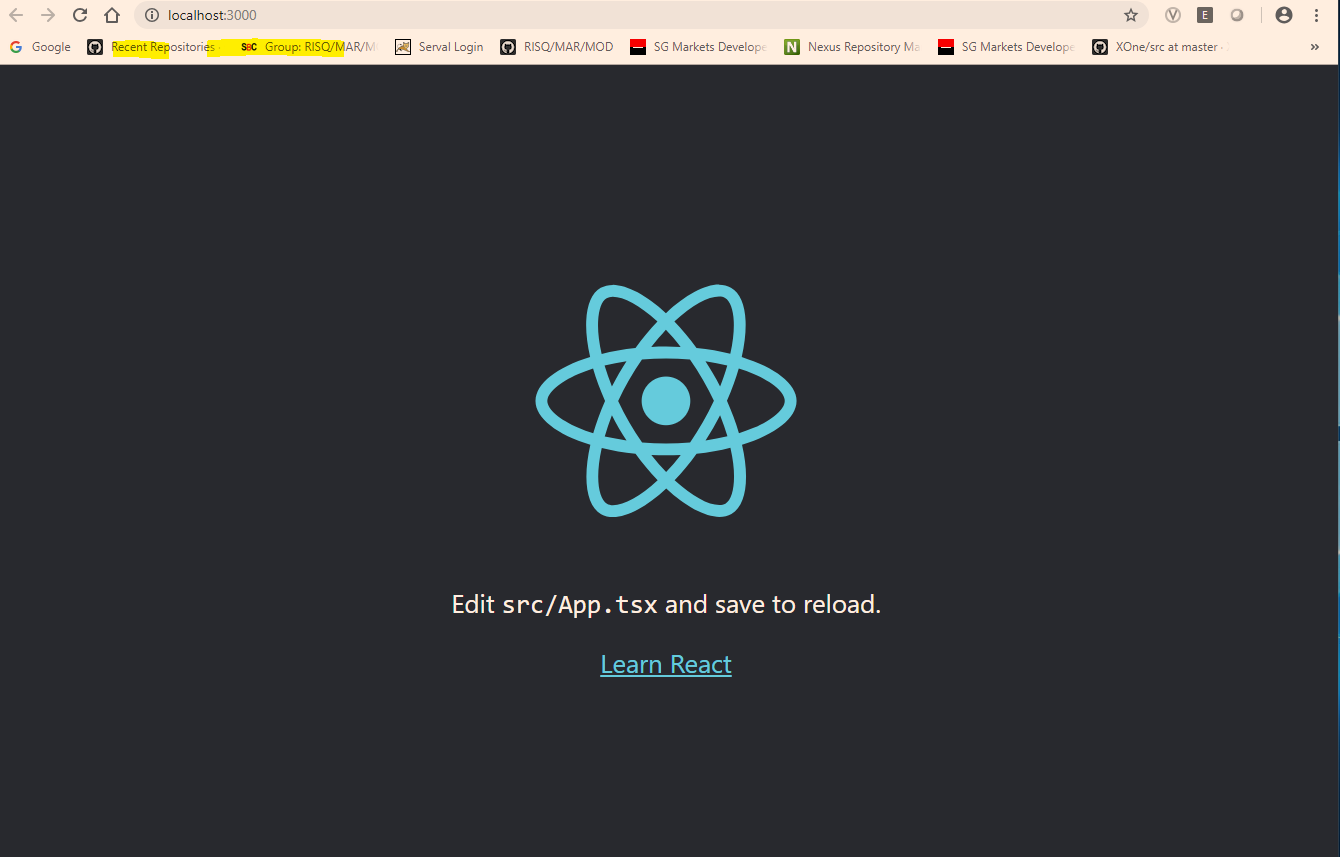
$ cd react-typescript-starter

$ npm start

Cette commande lance l’application et si tout se passe bien vous verrez la page ci-dessous affichée :

La page est ouvert sur <http://localhost:3000>, le port par défaut est 3000. Vous pouvez changer le port en créant un fichier .env dans la racine du projet, c’est-à-dire au même niveau que « package.json »

Dans ce fichier écrivez la ligne : « PORT=400 ». Pour voir le changement il suffit de relancer l’appli.



\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Start:

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Let's start now, we have installed the developpment tool and we are going to create our new project:

Install the react application developpment package Create-React-App

$ npm install -g create-react-app

(https://create-react-app.dev/)

This tool will help us to create the react app structure.

Open terminal and go to to your prefered workspace and follow the steps below:

$ npx create-react-app react-typescript-app --typescript

$ cd react-typescript-app

note:

/ react-typescript-app is name of the project

/ argument --typescript used to work with Typescript

the command npx execute npm package binairies, in our case it execute create-react-app library by

The comand generated the following structure :

- .git

- node\_modules

- public

--- favicon.ico

--- index.html

--- logo192.png

--- logo512.png

--- manifest.json

--- robots.txt

- src

--- App.css

--- App.test.tsx

--- App.tsx

--- index.css

--- index.tsx

--- logo.svg

--- react-app-env.d.ts

--- serviceWorker.ts

- .gitignore

- package.json

- package-lock.json

- README.md

- tsconfig.json

Exlanantion of each part:

TODO: Explain the structure

\* tsconfig.json declare the root of typescript project and specifies compiler parameters to compile the project.

\* public is the folder in where the static file reside such as the index.html, the following tag div in the file

`<div id="root"></div>` is a Reactjs entry point

\* src is a folder that will hold all app code: component, style, images etc...

the file `index.tsx` in the src foler is a react main program

the main component is executed using this code `ReactDOM.render(<App />, document.getElementById('root'));`

Everything in react is component and only one the root component here is App.

Note :

.ts are regular typescript file and .tsx are typescript syntax extension file

All the component we will create in this tutorial will have .tsx as extension

Inportant packages:

------------

Router:

We need a library to manage page routing.

$ npm install --save react-router-dom

# As we use typescript in our project we need also the @type package for react-router-dom.

We can say that @types are the interface between Javascript and Typescript

# We save @types in devDependencies

$ npm install --save-dev @types/react-router-dom

(http://definitelytyped.org/)

(https://reacttraining.com/react-router/web/guides/quick-start)

Bootstrap:

Boostrap is used to create the user interface:

$ npm install --save bootstrap

import 'bootstrap/dist/css/bootstrap.min.css';

(https://getbootstrap.com/)

icon:

$ npm install --save font-awesome

(https://fontawesome.com/icons?d=gallery)

note:

to import the css use following

import 'font-awesome/css/font-awesome.min.css';

Extra packages:

------------

Installing TSLint-React:

$ npm install --save-dev tslint tslint-react

Type the following command to generate a tslint.json file with default options.

$ npx tslint --init

\* tslint.json is the linter settings to be used by TSLint

Replace this file with the following:

{

"defaultSeverity": "error",

"extends": [

"tslint-react"

],

"jsRules": {

},

"rules": {

"member-access": false,

"ordered-imports": false,

"quotemark": false,

"no-console": false,

"semicolon": false,

"jsx-no-lambda": false

},

"rulesDirectory": [

],

"linterOptions": {

"exclude": [

"config/\*\*/\*.js",

"node\_modules/\*\*/\*.ts"

]

}

}

Note:

argument --save is used to save the package into "dependencies" section

and --save-dev save package in "devDependencies"

I will explain later, for now keep going on

The tool tslint checks typescript code for readability, maintainability, and functionality errors

(https://palantir.github.io/tslint/)

tslint-react: Lint rules related to React & JSX for TSLint.

(https://github.com/palantir/tslint-react)

We will add also Prettier:

Prettier is a fully automatic �style guide� worth adopting.

npm install prettier --save-dev

(https://prettier.io/docs/en/install.html)

Launch app:

-------------

Open the app in vs code and open a new terminal by clicking on menu "Terminal->New terminal"

So you can close your favorite terminal, we will vscode terminal to manage app.

Type the command:

$ npm start

to launch the application

If everything is going well, you will see the page with logo and theh text "Edit src/App.tsx and save to reload."

We will change the text. For that open the file "App.tsx" in src folder

Delete the text and add this "My first react app using typescript." and save.

The page is refreshed automatically.

Now we will create a CRUD component.

Create a new folder "components" in "/src" folder

+-- /src

| +-- /components

Delete code in App.tsx and we will create a class component instead, you can delete the other App.test.tsx and App.css files:

we will import the following files to build a beautify user interface:

import 'bootstrap/dist/css/bootstrap.min.css';

import 'font-awesome/css/font-awesome.min.css';

TODO: code here...

(https://getbootstrap.com/docs/4.4/examples/dashboard/)

Note:

component class must implement a render method which is used by reactjs to render the UI in the broswer

Reactjs Lifecyle :

-----------------

- init

- update

- destruct

-----------------------------------------

usefull links :

https://github.com/typescript-cheatsheets/react-typescript-cheatsheet#reacttypescript-cheatsheets

<https://code.visualstudio.com/docs/nodejs/reactjs-tutorial>

├── .git

├── node\_modules

├── public

│ ├── favicon.ico

│ ├── index.html

│ ├── logo192.png

│ ├── logo512.png

│ ├── manifest.json

│ ├── robots.txt

├── src

│ ├── App.css

│ ├── App.test.tsx

│ ├── App.tsx

│ ├── index.css

│ ├── index.tsx

│ ├── logo.svg

│ ├── react-app-env.d.ts

│ ├── serviceWorker.ts

│ └── setupTests.ts

├── .gitignore

├── package.json

├── package-lock.json

├── package.json

├── README.md

└── tsconfig.json