Framework front-end ReactJS

Membre 1 : Pierre Bastien

Membre 2 : Ruell Arnaud

# Signalétique

|  |  |
| --- | --- |
| Version | 16.5.0 |
| Langage (Javascript, typescript, etc.) | Javascript |
| Divers | Utilisation de Redux et de React-router |
| Site officiel | https://reactjs.org/ |

# Installation

Responsable membre 1

## Prérequis techniques

## - Il faut que node packet manager (npm) soit installé dans sa dernière version

- Il faut que le paquet « *create-react-app* » soit installé en global (*npm install -g create-react-app*)

## Téléchargement

Url de téléchargement ou instructions pour télécharger le framework

## Installation

- Lancer la commande create-react-app nomduprojet

### Environnement de développement

### Environnement de production

## Organisation des fichiers

## 

- package.json va regrouper les dépendances de notre application ainsi que des informations comme la version, le nom ou les scripts qui nous permettront d’exécuter l’application.

Dans le dossier *src*  
- *index.js* va être le point d’entrer de l’application, il va juste rendre *App.js* dans la page *index.html*.  
- *App.js* est le component de départ, c’est donc là que nous allons commencer à ajouter des components.  
- *index.css* sera le css qui sera partagé à l’application entière  
- *App.test.js* va nous permettre de faire des tests sur le component *App*  
- *App.css* est le css qui sera appliqué uniquement au component *App* par convention  
- *serviceWoker.js* nous permet d’enregistrer un service worker mais nous ne l’utiliseront pas ici

## Modules supplémentaires

- Déplacez-vous dans le dans le dossier du projet et lancer le terminal dans celui-ci  
 - Lancer la commande npm install redux react-redux redux-thunk react-router-dom  
 (Si vous avez une erreur dans l’installation avec redux-thunk relancer l’installation en tant qu’administrateur)

- Redux sera la librairie qui permet d’avoir une achitecture flux, ce qui facilitera le partage de données et leur cohérence entre les vues  
 - React-redux va permettre d’avoir des outils pour nous faciliter l’utilisation de Redux avec un projet React  
 - Redux-thunk va nous permettre de faire des fonctions asynchrones dans la libraire Redux  
 - React-router-dom va nous permettre d’avoir un système de routage

# Création de composants web

Responsable membre 2

## Description d’un composant web

Décrivez comment votre framework vous permet de programmer un composant web (vue, modèle de vue, liaison de données, etc.). Dans quel dossier sont rangés les composants web ? Donnez le lien vers la documentation correspondante à votre framework.

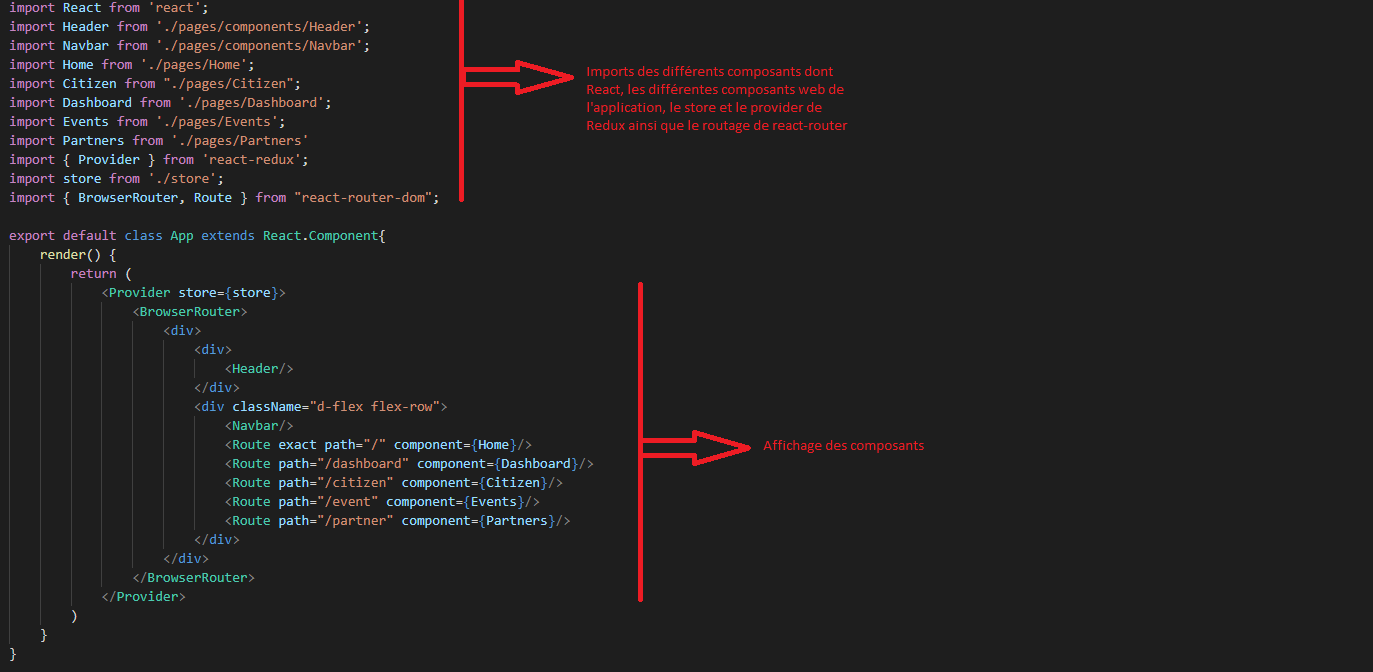
React permet de créer des composants web héritant de la classe « Component » de React. Ces composants sont des classes Javascript qui possèdent la méthode « render » qui est appelée pour effectuer l’affichage de la page. Il suffit ensuite de retourner depuis cette fonction l’affichage qui est décrit comme une page HTML. Il y a ensuite les « reducers » et les « actions » de Redux qui vont permettre d’interagir avec le back-end. Il suffit d’appeler les méthodes dans les « actions » qui vont-elles même appeler le « reducer » qui mettra à jour les informations de l’application web et qui actualisera la vue. Les informations de l’application étant récupérées depuis le back-end sont des objets Javascript qui peuvent donc être directement utilisés dans la vue.

Le composants web sont rangés dans le dossier « pages ».

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

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

## Exemple de composant de top niveau ou destiné au routing

Présentez et expliquez le code correspond à votre composant de top niveau, susceptible d’accueillir de le composant à afficher après routage. 

Provider : Permet de donner le « store » qui contient les données à tous les composants de l’application.

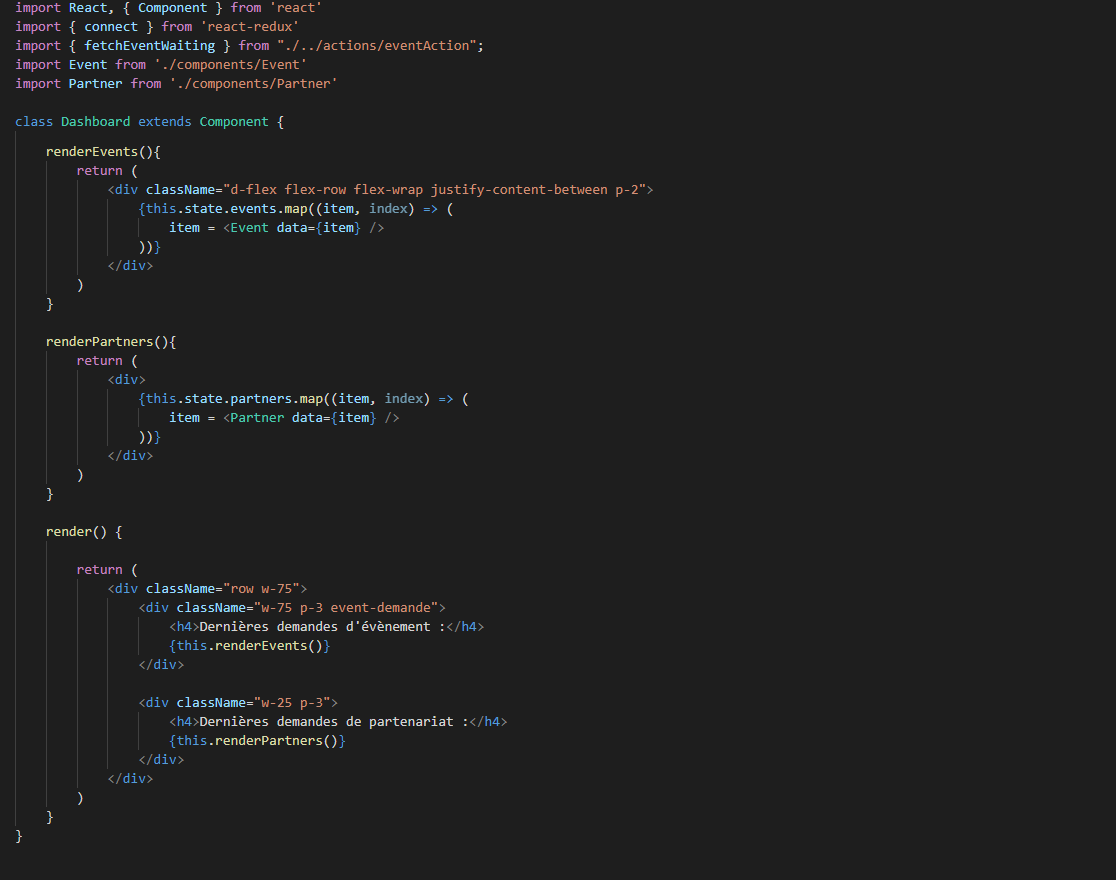
BrowserRouter : Permet de faire le routage de l’application.

Header : Représente le header qui va apparaitre sur chaque page.

Navbar : Représente la barre de navigation qui va apparaitre sur chaque page.

Route : Permet d’afficher le composant correspondant à la route utilisée.

## Exemple de composants de haut niveau

Présentez et expliquez le code du composant dashboard. Ce composant de haut niveau comporte plusieurs sous-composant. Comment sont-ils invoqués ? 

La méthode « render » permet d’afficher la page. On y retrouve des balises avec des « className » qui permettent de styliser l’affichage et on trouve aussi des appels aux méthodes « renderEvents » et « renderPartners » qui permettent de générer l’affichage de chaque objet évènement et partenaire. L’affichage de ces objets est géré par les sous-composants « Event » et « Partner ». Après avoir importé ces composants (qui agissent de la même manière que le composant « Dashboard ») on peut simplement les appeler là où on souhaite les afficher. Chaque objet est passé en propriété via « data={item} » au sous-composant correspondant ce qui permet à ce dernier d’utiliser les données de l’objet.

## Exemple de composant avec formulaire

Présentez et expliquez le code du composant d’édition d’une nouvelle invitation. Ce composant contient un formulaire à soumettre. Comment votre framework lie les données du formulaire à son modèle de vue ?

# Routage frontend

Responsable membre 1

## Introduction

Expliquez brièvement le fonctionnement du routage (fichiers concernés, liens vers documentation) Indiquez également s’il y a des facilités pour valider les formulaires ou valider certaines demandes de navigation (utilisateur authentifié par exemple)

## Routage simple

Sur base du projet AskMo, présentez un exemple de syntaxe vers une route sans paramètre

## Routage avec paramètres explicites

Sur base du projet AskMo, présentez une route avec paramètres explicitement nommés

## Routage avec paramètres implicites (wildcard)

Sur base du projet AskMo, présentez une route avec paramètres non nommés

Framework backend

# Signalétique

|  |  |
| --- | --- |
| Version |  |
| Langage (Javascript, typescript, etc.) |  |
| Divers |  |
| Site officiel |  |

# Installation

Responsable membre 2

## Prérequis techniques

Listez les prérequis nécessaires sur votre poste de travail pour héberger le framework.

## Téléchargement

Url de téléchargement ou instructions pour télécharger le framework

## Installation

Expliquez brièvement les étapes de l’installation ; donnez vos conseils pour l’installation sur les serveurs de l’école si besoin ; citez des ressources utiles (tutoriel, pages de documentation liées à l’installation).

### Environnement de développement

### Environnement de production

## Organisation des fichiers

Montrez et expliquez l’organisation des fichiers du framework (cf. document annexe).

## Configuration

Expliquez les différentes étapes de configuration lorsque le framework est installé (cf. document annexe).

## Modules supplémentaires

Listez et expliquez brièvement les modules supplémentaires que vous conseillez d’installer (ex : module de routage).

## Générateur de code

Expliquez s’il existe un générateur de code et quelles sont les possibilités.

# Routage backend

Responsable membre 2

## Introduction

Expliquez brièvement le fonctionnement du routage (fichiers concernés, liens vers documentation) Indiquez également s’il y a des facilités pour valider les formulaires ou valider certaines demandes de navigation (utilisateur authentifié par exemple)

## Routage simple

Sur base du projet AskMo, présentez un exemple de syntaxe vers une route sans paramètre

## Routage avec paramètres explicites

Sur base du projet AskMo, présentez une route avec paramètres explicitement nommés

## Routage avec paramètres implicites (wildcard)

Sur base du projet AskMo, présentez une route avec paramètres non nommés