PARCOURS FRONT-END

Framework JS

Ateliers React

Exercice « Layout »

- Créer un nouveau projet React.
- L'objectif est de créer une mise en page (ou layout) pour votre futur site web fait avec React.
- Vous allez devoir créer plusieurs composants, suivant la hiérarchie suivante :
  - App
    - Header
      - Logo
      - Navigation
    - Main
    - Footer
- Comme vous l'aurez compris, un 1er composant (App) est chargé de contenir l'ensemble de l'UI, elle-même divisée en 3 grandes parties :
  - La barre de navigation, séparée néanmoins du logo pour plus de sémantique et tous deux regroupés dans un composant Header.
  - Un bloc de contenu principal, dont le contenu serait amené à changer, en fonction de la navigation par exemple.
  - Un bas de page, où l'on retrouvera quelques liens utiles, un copyright, …
- Définir le rendu (à l'aide de JSX) de chacun de ces composant afin de représenter les parties demandées.
- Penser à bien insérer les composants là où cela est nécessaire !
- Libre à vous d'utiliser du CSS pure ou bien n'importe quel API CSS (bootstrap, …)
- Concernant l'arborescence et l'organisation des sources, entre JS et CSS, libre à vous d'utiliser les conventions que vous souhaitez. Une bonne approche serait, pour chaque composant, de créer un dossier portant le nom du composant et contenant au moins le fichier JS du composant, ainsi qu'un fichier CSS pour ce composant (tous 2 ayant le même nom avec les extensions .js et .css).

Exercice « Layout children »

- Créer un nouveau projet React, basé sur le précédent.
- Faire en sorte d'obtenir le même rendu, en transformant votre JSX de manière à utiliser la propriété « children ».

Exercice « Profile »

- Créer un nouveau projet React
- Dans le composant principal « App », définir un objet contenant les informations personnelles d'un utilisateur :
  - Infos générales :
    - Nom
    - Prénom
    - Age
  - Infos de l'adresse :
    - Rue
    - Code postal
    - Ville
    - Pays
- Créer 3 nouveaux composants :
  - « ProfileGeneral » : ce composant doit présenter le nom, prénom et l'age sous la forme d'une liste.
  - « ProfileAddress » : ce composant doit présenter les différentes informations de l'adresse sous la forme d'une liste avec un encadré
  - « Profile » : ce composant doit afficher les composants « ProfileGeneral » et « ProfileAddress ». Il doit aussi recevoir de « App » les informations de l'utilisateur et transmettre, parmi ces informations, celles qui sont nécessaires aux composants « ProfileGeneral » et « ProfileAddress ».

Exercice « Clock »

- Créer un nouveau projet React
- L'objectif de cet exercice est de créer un composant « Clock » affichant l'heure.
- Le composant doit être fait de manière à ce que, chaque seconde, l'heure se mette à jour.

Exercice « Counter »

- Créer un nouveau projet React
- L'objectif de cet exercice est de créer un composant « Counter » affichant :
  - Un nombre (valeur par défaut « 0 »).
  - Un bouton « + » qui, au clic, incrémentera de 1 le compteur
  - Un bouton « - » qui, au clic, décrémentera de 1 le compteur
  - Un bouton « reset » qui, au clic, réinitialisera le compteur à 0

Exercice « Apple basket »

- Créer un nouveau projet React
- Créer 2 composants :
  - AppleBasket :
    - Ce composant doit maintenir une liste de pomme (une pomme est un objet avec une propriété « id ») vide à l'état initiale.
    - Au niveau de la vue, il doit :
      - Afficher un bouton permettant d'ajouter une nouvelle pomme dans la liste de pomme (générer un nouvel id).
      - Afficher un texte indiquant qu'il faut cliquer sur une pomme pour la supprimer
      - Afficher un séparateur <hr> pour plus de clarté
      - Afficher autant de composant Apple qu'il y a d'objet de type Apple dans la liste de pomme.
  - Apple :
    - Ce composant prend en entrée une propriété « apple » (un objet pomme).
    - Au niveau de la vue, il doit :
      - Afficher l'image d'une pomme (image pouvant être fournie par le formateur).
      - Au survol de la pomme, un titre s'affiche (attribut title) indiquant l'id de la pomme.
      - Au clic sur la pomme, cette dernière doit être retirée de la liste des pommes maintenue dans AppleBasket.

Exercice « Contact form »

- Créer un nouveau projet React
- Créer un composant « ContactForm » contenant :
  - Un champ input de type texte « Prénom »
  - Un champ input de type texte « Nom »
  - Un champ textarea « Motif »
- Un bouton « Envoyer » qui, au clic, soumettra le formulaire, ce qui affichera les données des champs dans une alerte.

Exercice « Tasklist »

- Créer un nouveau projet React
- Créer une classe JavaScript « TaskModel », représentant une tâche, avec les propriétés suivantes :
  - « id » représentant l'id de la tâche dans la liste
  - « title » représentant le titre de la tâche
- Créer un composant « Task » contenant :
  - Une propriété en entrée « task » de type TaskModel
  - Concernant la vue de « Task », il faut afficher :
    - Une ligne contenant le titre de la tâche (ce titre est modifiable)
    - Un bouton permettant de supprimer la tâche de la liste des tâches (voir après)
- Créer un composant « TaskList » contenant :
  - Une propriété « tasks » qui contiendra des instances de TaskModel
  - Concernant la vue de « TaskList », il faut afficher :
    - Un bouton « Ajouter une tâche » qui ajouter une nouvelle instance de TaskModel dans la propriété « tasks » (générer un nouvel id)
    - Une liste de « Task »

Exercice « Tasklist redux »

- Reprendre l'exercice Tasklist
- Procéder à se refonte à l'aide de Redux

Exercice « Ping pong »

- Créer un nouveau projet
- Le but est de réaliser un petit jeu de ping pong aléatoire
- L'interface doit contenir :
  - Un bouton avec le titre « Let's play » :
  - Un paragraphe affichant un message variable « Message : … »
  - 2 boites (carrés) contenant respectivement les textes « Player 1 » et « Player 2 »
  - Une image représentant une raquette de Ping Pong qui se positionnera sous les 2 boites.
- L'état de votre application doit être géré par redux.
- Les informations à gérer sont les suivantes :
  - **isPlaying** : booléen déterminant si une partie est en cours ou non. Cette valeur permettra l'affichage de la raquette (si vraie)
  - **message** : chaîne contenant les différents messages qui s'afficheront dans le paragraphe en fonction des actions envoyées au store
  - **player** : numéro du joueur en train de jouer (1 ou 2)
- 4 actions pourront être dispatchées au store :
  - **GAMESTART** : change le message en « New game ! » et initialise player à 1
  - **PING** : met isPlaying à true, change la valeur de player de 1 à 2 ou 2 à 1 et change le message en « PING ! » si le joueur 1 joue ou « PONG ! » si c'est le joueur 2
  - **VICTORY** : change le message en « Player X has won ! » où X est le numéro du joueur ayant gagné.
  - **GAMEOVER** : met isPlaying à false et change le message en « Game over ! »
- Créer une watcher saga (avec redux-saga) pouvant capter une action de type « PLAY » et la déléguant à une worker saga.
- Cette worker saga devra :
  - Générer un nombre aléatoire X entre 1 et 10
  - Créer un effet de dispatch de l'action « **GAMESTART** »
  - Créer un effet d'un délai de 2sc
  - Faire une boucle qui itèrera X fois et dans laquelle vous devez :
    - Créer un effet de dispatch de l'action « **PING** »
    - Créer un effet d'un délai de 1sc
  - Créer un effet de dispatch de l'action « **VICTORY** »
  - Créer de nouveau un effet d'un délai de 2sc
  - Créer un effet de dispatch de l'action « **GAMEOVER** »
- Réaliser tous les branchements nécessaires dans votre application pour que cette dernière fonctionne.
- **BONUS :** A l'affichage, faire en sorte que l'image de la raquette se positionne sous la boite correspondant au joueur en train de jouer.

Exercice « Webmarket »

- Créer un nouveau projet
- Le but est de réaliser une mini plateforme présentant des articles en ligne
- Le site doit présenter une navigation de 4 liens redirigeant vers des pages :
  - Accueil (par défaut) :
    - Accessible via « / »
    - Cette page affiche simplement un message de bienvenue
  - Livres :
    - Accessible via « /books »
    - Cette page affiche une liste de titres de livre
    - Chaque titre est cliquable et redirige vers une page de détail (titre + description) accessible via « /books/id », où id est l'identifiant du livre.
  - Repas :
    - Même choses que pour Livres mais avec des données liées à des repas
    - Accessible via « /food »
    - Page de détail accessible via « /food/id »
  - Clothes :
    - Même choses que pour Livres mais avec des données liées à des vêtements
    - Accessible via « /clothes »
    - Page de détail accessible via « /clothes/id »
- Les données en question peuvent être enregistrées en dur dans des fichiers JSON présents dans le projet.
- Libre à vous de choisir les informations qui vous plairont.
- La structure recommandée pour ces fichiers JSON est la suivante :

```
{
   "title": "Category",
   "list": [
      {
         "id": 1,
         "title": "Item 1",
         "description": "..."
      },
      {
         "id": 2,
         "title": "Item 2",
         "description": "..."
      },
      {
         "id": 3,
         "title": "Item 3",
         "description": "..."
      }
   ]
}
```