Contrôle Continu n°1 : Développement Front End
20234/2025

Filière	:	DEVOFST	Variante 2	Groupes : 203
Niveau	:	TS		
Durée	:	1 h 30		Barème : /20

Nom & Prénom :

Partie 1 : QCM (1x6 points)

Q1. Le Virtual DOM est :

a) Une représentation légère en mémoire du DOM réel
b) Une copie exacte du DOM réel
c) Un outil de débogage React
d) Un nouveau type de navigateur web

Q2. Pour créer un nouveau projet React, on utilise :

a) npm install react
b) npx create-react-app nom-projet
c) create react-app
d) npm start react-app

Q3. Les composants React peuvent être :

a) Uniquement des classes
b) Des classes ou des fonctions
c) Uniquement des fonctions
d) Ni des classes ni des fonctions

Q4. Les props sont :

a) Uniquement utilisables dans les composants de classe
b) Automatiquement mises à jour par React
c) Modifiables et gérées en interne du composant
d) Immuables et passées par le composant parent

Q5. Pour modifier le state dans un composant de classe, on utilise :

a) this.setState()
b) state.update()
c) this.state = newState
d) updateState()

Q6. Quelle est la nature exacte de create-react-app ?

a) Une commande native de Node.js
b) Un package npm qu'on doit installer globalement
c) Un module intégré à React
d) Un script npm qu'on exécute pour créer des projets React

Partie 2 : (14 points)

Exercice 1 : Attribution aléatoire de groupes aux sujets **(6 points)**

Développez une application **React** qui permet d'attribuer aléatoirement des groupes de stagiaires à des sujets selon les critères suivants :

1. Créez un composant de classe **React** qui affiche une liste non ordonnée. (1 point)
2. La liste doit contenir exactement 4 éléments. (1 point)
3. Chaque élément de la liste doit être affiché sous le format suivant :
   "Sujet i → Groupe n" (1 point)
4. Les sujets doivent être affichés dans l'ordre croissant (Sujet 1, Sujet 2, Sujet 3, Sujet 4). (1 point)
5. Les groupes (Groupe 1, Groupe 2, Groupe 3, Groupe 4) doivent être : (2 points)
   - Stockés dans un tableau
   - Attribués de manière aléatoire aux sujets
   - Chaque groupe ne peut être attribué qu'une seule fois

Exemple de sortie attendue :

- Sujet 1 → Groupe 4
- Sujet 2 → Groupe 1
- Sujet 3 → Groupe 2
- Sujet 4 → Groupe 3

Solution :

ListeGroupes.jsx

```jsx
import React, { Component } from 'react';

class ListeGroupes extends Component {
  constructor(props) {
    super(props);
    this.groupes = ['Groupe 1', 'Groupe 2', 'Groupe 3', 'Groupe 4'];
    this.state = {
      groupesAléatoires: this.melangerGroupes([...this.groupes])
    };
  }
  //const melangerGroupes = this.groupes.sort(() => Math.random() - 0.5);
  melangerGroupes(array) {
```

```
    for (let i = array.length - 1; i > 0; i--) {
      // Générer un index aléatoire entre 0 et i
      const j = Math.floor(Math.random() * (i + 1));
      // Échanger les éléments aux indices i et j
      [array[i], array[j]] = [array[j], array[i]];
    }
    return array;
  }

  render() {
    return (
      <div>
        <h2>Attribution des groupes aux sujets</h2>
        <ul>
          {[1, 2, 3, 4].map((numSujet, index) => (
            <li key={numSujet}>
              Sujet {numSujet} → {this.state.groupesAléatoires[index]}
            </li>
          ))}
        </ul>
      </div>
    );
  }
}

export default ListeGroupes;
```

App.js

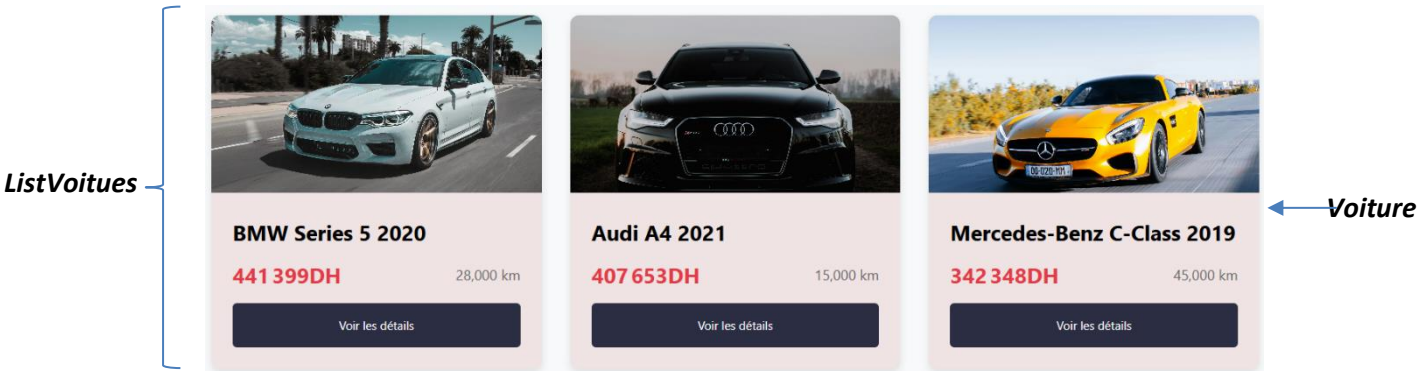
```
import ListeGroupes from './components/ListeGroupes';

function App() {
  return <ListeGroupes  />
}
export default App;
```

Exercice 2 : Gestion d'une liste de voitures (**8 points**)

Créer une application **React** permettant d'afficher une liste de voitures de luxe.

Consignes :

- Rendu de l'application final :



- La liste des voitures est disponible dans le fichier ***voitures.json*** qui a la structure suivante :

```
{
  "voitures": [
    {
      "id": 1,
      "marque": "Mercedes-Benz",
      "modele": "C-Class 2019",
      "prix": 342348,
      "kilométrage": "45,000 km",
      "image": "https://images.unsplash.com/photo-Mercedes-Benz?ixlib=rb-4.0.3"
    },
    ...
  ]
}
```

- On considère le fichier de style **App.css** déjà fourni.

Travail à réaliser :

1. Proposez une structure de dossiers appropriée pour l'application. (**2 points**)
2. Créez deux composants fonctionnels : (**4 points**)
   - ***Voiture*** : affiche les détails d'une voiture (marque, modèle, prix, kilométrage et image). (2 points)
   - ***ListeVoitures*** : affiche la liste des voitures passée comme attribut en utilisant le composant ***Voiture***. (2 points)
3. Dans le composant **App.js**, importez et utilisez le composant **ListeVoitures**. (**2 points**)

Solution :

1- Structure des dossiers de l'application :

```
ra-cars
|---public
|   |---index.html
|---src
|    |---components
|    |       |---Voiture.jsx
|    |       |--- ListeVoitures.jsx
|    |---data
|    |       |---voitures.json
|    |---App.js
|    |---index.js
```

2- Composants fonctionnels :

Voiture.jsx

```
// Les différents attributs 'className' sont optionnels

function VoitureCard({ voiture }) {

  return (
    <div className="voiture-card">
      <div className=" voiture -image">
        <img src={voiture.image} alt={`${voiture.marque} ${voiture.modele}`} />
      </div>
      <div className="voiture -details">
        <h2>{voiture.marque} {voiture.modele}</h2>
        <div className="voiture -info">
          <span className="prix">{voiture.prix.toLocaleString()}DH</span>
          <span className="kilométrage">{voiture.kilométrage}</span>
        </div>
        <button className="details-btn">Voir les détails</button>
      </div>
    </div>
  );
}

export default VoitureCard;
```

CarList.jsx

```
import VoitureCard from './ VoitureCard ';
```

```
function ListeVoitures({ voitures }) {
  return (
    <div className="car-list-container">
      <h1>Nos Voitures Premium</h1>
      <div className="voiture-list">
        { voitures.map(voiture => (
          <VoitureCard key={voiture.id} voiture={voiture} />
        ))}
      </div>
    </div>
  );
}

export default ListeVoitures;
```

3- Composant App.js

```
import ListeVoitures from './components/ ListeVoitures';

import voituresData from './data/voitures.json';

function App() {
  return (
    <div className="App">
        < ListeVoitures voitures={voituresData.voitures} />
    </div>
  );
}

export default App;
```