Contrôle Continu n°1 : Développement Front End
2024/2025

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

Nom & Prénom :

Partie 1 : QCM (1x6 points)

Q1. Le Virtual DOM est :

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

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

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

Q3. Les composants React peuvent être :

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

Q4. Les props sont :

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

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

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

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

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

Partie 2 : (14 points)

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

Développez une application React qui permet d'attribuer aléatoirement des sujets à des groupes de stagiaires 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 : "Groupe i → Sujet n" (1 point)
4. Les groupes doivent être affichés dans l'ordre croissant (Groupe 1, Groupe 2, Groupe 3, Groupe 4). (1 point)
5. Les sujets (Sujet 1, Sujet 2, Sujet 3, Sujet 4) doivent être : (2 points)
    - Stockés dans un tableau
    - Attribués de manière aléatoire aux groupes
    - Chaque sujet ne peut être attribué qu'une seule fois

Exemple de sortie attendue :

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

Solution :

ListeSujets.jsx

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

class ListeSujets extends Component {
  constructor(props) {
    super(props);
    this.sujets = ['Sujet 1', 'Sujet 2', 'Sujet 3', 'Sujet 4'];
    this.state = {
      sujetsAléatoires: this.melangerSujets([...this.sujets])
    };
  }
  //const melangerSujets = this.sujets.sort(() => Math.random() - 0.5);
  melangerSujets(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 sujets aux groupes</h2>
        <ul>
          {[1, 2, 3, 4].map((numGroupe, index) => (
            <li key={numGroupe}>
              Groupe {numGroupe} → {this.state.sujetsAléatoires[index]}
            </li>
          ))}
        </ul>
      </div>
    );
  }
}

export default ListeSujets;
```

App.js

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

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

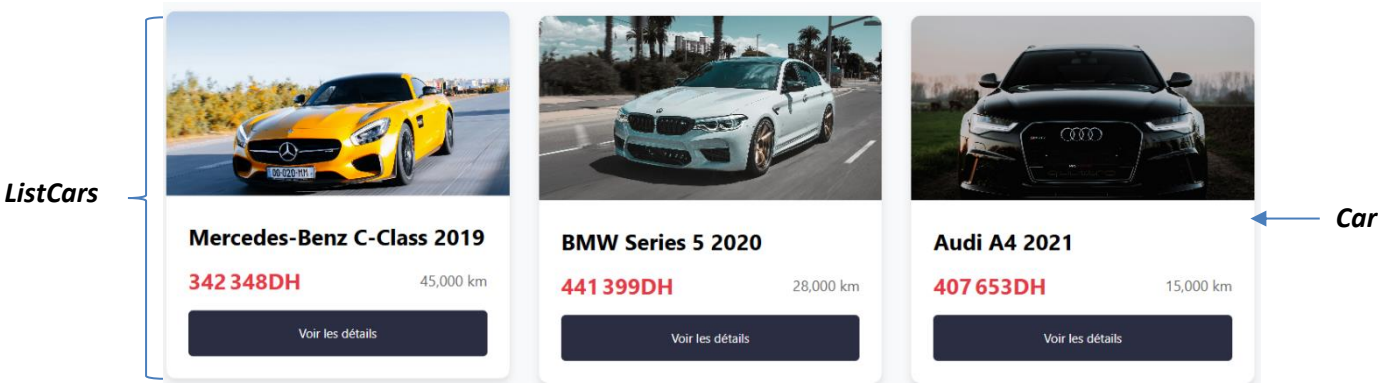
Exercice 2 : Gestion d'une liste de voitures (**8 points**)

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

Consignes :

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

- Rendu de l'application final :



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

```
{
  "cars": [
    {
      "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"
    },
    ...
  ]
}
```

Travail à réaliser :

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

Solution :

1- Structure des dossiers de l'application :

```
ra-cars
|---public
|    |---index.html
|---src
|     |---components
|     |        |---Car.jsx
|     |        |--- ListeCars.jsx
|     |---data
```

```
|    |       |---cars.json
|    |---App.js
|    |---index.js
```

2- Composants fonctionnels :

Car.jsx

```
// Les différents attributs ‘className’ sont optionnels

function CarCard({ car }) {

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

export default CarCard;
```

ListeCars.jsx

```
import CarCard from './CarCard';


function ListeCars({ cars }) {
  return (
    <div className="car-list-container">
      <h1>Nos Véhicules Premium</h1>
      <div className="car-list">
        {cars.map(car => (
          <CarCard key={car.id} car={car} />
        ))}
      </div>
    </div>
  );
}
```

```
export default ListeCars;
```

3- Composant App.js

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

import carsData from './data/cars.json';

function App() {
  return (
    <div className="App">
        < ListeCars cars={carsData.cars} />
    </div>
  );
}

export default App;
```