

Solution EFM React Js

Exercice I : QCM

- 1) Comment incorporer de style inline qui spécifie background-color rouge font-size 18px dans l'élément <p style=...../> (une seule bonne réponse) (2 pts)
- A. Style={{font-size :18;background-color :'red'}}
 - B. Style={{fontSize :'18px',backgroundColor :'red'}}
 - C. Style={{fontsize :'18px',backgroundcolor :'red'}}
 - D. Style={fontSize :'18px',backgroundColor :'red'}

Réponse B : style={{ fontSize: '18px',backgroundColor: 'red' }}

- 2) Dans un classe composant quelle est la fonction pour modifier de la variable d'état quantité à 15 (une seule bonne réponse) (2pts)
- A. this.state.quantite=15
 - B. this.setState({quantite=15})
 - C. this.setState({quantite :15})
 - D. this.getState({quantite :15})

En React, pour modifier l'état (state) d'un composant, on utilise la fonction `setState()`. Si vous souhaitez modifier l'état "prix" à 230, vous pouvez utiliser la syntaxe suivante dans votre composant :

Réponse C : `this.setState({ quantite: 15 });`

-et si on travaille avec la fonction `component` on utilise le hook `useState`:

```
const [prix, setPrix] = useState(0);  
  
setPrix(230);
```

```
import React, { useState } from 'react';
```

```
function MonComposant() {
```

```
  const [prix, setPrix] = useState(0);
```

```
  function handleClick() {
```

```
    setPrix(230);
```

```
  }
```

```
  return ( <div>
```

```
    <p>Le prix est: {prix} </p>
```

```
    <button onClick={handleClick}>Modifier le prix</button>
```

```
  </div>);}
```

Function
Component

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

```
class MonComposant extends Component {
```

```
  constructor(props) {
```

```
    super(props);
```

```
    this.state = { prix: 0 };
```

```
    this.handleClick = this.handleClick.bind(this);
```

```
  }
```

```
  handleClick() {
```

```
    this.setState({ prix: 230 });
```

```
  } render() {
```

```
    return (
```

```
      <div>
```

```
        <p>Le prix est: {this.state.prix} </p>
```

```
        <button onClick={this.handleClick}>Modifier le prix</button>
```

```
      </div> );
```

```
    } }
```

Class
Component

Solution EFM React Js

3) dans un composant créer par fonction comment appeler en jsx une fonction myFunction à partir d'un élément button déclenché par click (choisir les deux bonnes réponses)(2pts)

- A. `<button onClick={()=>MyFunction()}>Update</button>`
- B. `<button onClick={{MyFunction()}}>Update</button>`
- C. `<button onClick={MyFunction()}>Update</button>`
- D. `<button onClick={MyFunction}>Update</button>`

Réponse D: `<button onClick={Myfunction}>Update</button>`

ou bien vous pouvez l'appeler en utilisant une arrowfunction :

Réponse A: `<button onClick={() => Myfunction ()}>Update</button>`

Cela peut être utile si vous devez passer des arguments.

4) Soit la liste `livres=[{id :10,titre : "REACT REDUX"},{id :11,titre : "Agile"}]`
créer `livres2` qui contient est la copie de `livres` en lui ajoute le livre `{id :12,titre : "HTML"}`
(choisir la bonne réponse)(2pts)

- A. `const livres2=[...livres,{id :12,titre : "HTML"}]`
- B. `const livres2 ;
livres.push({id :12,titre : "HTML"}) ;livres2=livres`
- C. `const livres2=[livres,{id :12,titre : "HTML"}]`

on va utiliser le spread operator :

Réponse A :-`const livres2 = [...livres, {id: 12, titre: "HTML"}];`

5) `useEffect(()=>{ document.title="cours react" },[])` est invoqué quand
(choisir la bonne réponse)(2pts)

- A. Le composant est supprimer
- B. Le composant est retirer du dom
- C. Après le premier render
- D. Après chaque render

Réponse C: Après le premier Render.

```
5-useEffect(()=>{  
  
document.title=title  
  
},[title])
```

Réponse :-Le code passé à `useEffect` sera invoqué après le premier render et à chaque fois que le state **title** est mis à jour.

Solution EFM React Js

6) En redux donner l'action qui est valide (choisir la bonne réponse)(2pts)

- A. {type : 'ADD_ARTICLE', payload : {id : 10, nom : 'article1'}}
- B. {action : 'ADD_ARTICLE' payload : {id : 10, nom : 'article1'}}
- C. {ADD_ARTICLE : {id : 10, nom : 'article1'}}
- D. {action : 'ADD_ARTICLE' , id : 10, nom : 'article1'}

Dans Redux, les reducers sont des fonctions qui prennent en entrée l'état actuel de l'application et une action, et renvoient un nouvel état en fonction de l'action. Les actions sont des objets qui décrivent ce qui s'est passé dans l'application, et qui contiennent souvent un type et un payload.

-l'action valide est :

{type:'nom_de_l'action',payload:'parametre(s)'}

Réponse A: {type:'ADD_ARTICLE',payload:{id:10,nom : 'article 1'}}

Exercice 2 :

On souhaite récupérer les données à partir d'un API endpoint : lien <https://freejson.com/api/livres>
Donner le code qui permet de récupérer la réponse de l'API (utiliser fetch ou axios) puis stocker la réponse dans la variable d'état livres

-Axios est une bibliothèque JavaScript populaire qui permet d'effectuer des requêtes HTTP . Elle fournit une interface simple pour effectuer des appels HTTP asynchrones et facilite la manipulation des données de réponse.

// npm install axios

AXIOS

```
import axios from 'axios';
import React, { useEffect, useState } from 'react'
import axios from 'axios';
export default function Api() {
  // declaration du state :
  const [livres, setLivres] = useState([]);
  useEffect(() => {
    axios.get('http://127.0.0.1:8000/api/livres')
      .then(response => setLivres(response.data));
  }, []);
```

FETCH

```
import React, { useEffect, useState } from 'react'
export default function Api() {
  // declaration du state :
  const [livres, setLivres] = useState([]);
  useEffect(() => {
    fetch('http://127.0.0.1:8000/api/livres')
      .then(response => response.json())
      .then(data => setLivres(data));
  }, []);
```

Solution EFM React Js

un IMC normal doit se situer entre 18,5 à 24,5.

Indice de masse corporelle

Poids en (Kg)
71

Taille en (mètre)
1 75

calculer

IMC:23.18 Normale

$IMC = r = \text{poids} / (\text{taille} * \text{taille})$

1-cr  er l'application :

`npx create-react-app partie2`

2-executer l'application :

`npm start`

```
import React, { useState } from 'react'

export default function CalculIMC() {

  // declaration des states:

  const[poids,setPoids]=useState();const[taille,setTaille]=useState();

  const[imc,setImc]=useState();const[msg,setMsg]=useState("");

  const calculimc=()=>{

    let r=poids/(taille*taille);          setImc(r.toFixed(2));

    if (r >= 18.5 && r <= 24.5) {

      setMsg("Normale"); } else if (r < 18.5) { setMsg("Sous-poids"); } else { setMsg("Surpoids"); } }

  return (

    <div>

      <h1> Indice de masse corporelle</h1>    <h3>Poids en Kg:</h3>

      <input type="text" value={poids} onChange={(e)=>setPoids(e.target.value)} />

      <h3>Taille en M:</h3>

      <input type="text" value={taille} onChange={(e)=>setTaille(e.target.value)} />

      <br /><br />

      <input type="button" value="Calculer" onClick={calculimc} />

      <div>

        <h4>IMC {imc} : {msg}</h4>

      </div> </div>

    ) }
```