Chapitre 6 : Gestion des évènements et les états

Table des matières

I. Les évènements en React

a. Introduction

Le traitement des événements permet essentiellement à l'utilisateur d'interagir avec une page Web et de faire quelque chose de spécifique lorsqu'un certain événement, comme un clic ou un survol, se produit.

Lorsque l'utilisateur interagit avec l'application, des événements sont déclenchés, par exemple le passage de la souris, l'appui sur une touche, un événement de changement, etc. L'application doit gérer les événements et exécuter le code.

React a son propre système de gestion des événements qui est très similaire à la gestion des événements sur les éléments DOM. **Le système de gestion des événements de React est connu sous le nom d'événements synthétiques.**

Le tableau suivant montre la liste des évènements supportes par React v15.

Table 6.1. DOM events supported by React v15

Event group	Events supported by React
Mouse events	onClick, onContextMenu, onDoubleClick, onDrag, onDragEnd, onDragEnter, onDragExit, onDragLeave, onDragOver, onDragStart, onDrop, onMouseDown, onMouseEnter, onMouseLeave, onMouseMove, onMouseOut, onMouseOver, onMouseUp
Keyboard events	onKeyDown, onKeyPress, onKeyUp
Clipboard events	onCopy, onCut, onPaste
Form events	onChange, onInput, onSubmit
Focus events	onFocus, onBlur
Touch events	onTouchCancel, onTouchEnd, onTouchMove, onTouchStart
UI events	onScroll
Wheel events	onWheel
Selection events	onSelect
Image events	onLoad, onError
Animation events	onAnimationStart, onAnimationEnd, onAnimationIteration
Transition events	onTransitionEnd

b. Ajouter un évènement

La gestion des événements avec les éléments React est très similaire à la gestion des événements sur les éléments DOM. Il y a quelques différences de syntaxe :

- Les événements React sont nommés en utilisant **camelCase**, plutôt que des minuscules, par exemple, OnClick au lieu de onclick en HTML
- Avec JSX, vous passez une fonction comme gestionnaire d'événement, plutôt qu'une chaîne.

Exemple 1 : Dans cet exemple, Le composant Exemple1Event va afficher un bouton dans lequel nous avons associe l'évènement onClick avec la fonction affiche.

```
class Exemple1Event extends React.Component {
  constructor(props){
      super(props)
      this.affiche = this.affiche.bind(this);
  }

  affiche(){
    alert("Clique sur button...");
  }

  render(){
    return <button onClick={this.affiche}>Cliquer ici</button>
  }
}
```

Exemple 2 : Cet exemple montre appel d'une fonction fléchés dans dans un évènement.

```
class Exemple1Event extends React.Component {
  render(){
    const affiche = () => {
      alert("Clique sur button...");
    }
    return <button onClick={affiche}>Cliquer ici</button>
  }
}
```

Exemple 3 :

```
class Exemple1Event extends React.Component {
  render(){
    return (
<button onClick={() => {alert('clique declenche ....')}}>
      Cliquer ici
</button> );
```

```
  }
}
```

c. SyntheticEvent

Les gestionnaires d'événements transmettent des instances de SyntheticEvent, **une enveloppe multi-navigateurs autour de l'événement déclenché par le navigateur.** Il possède la même interface que l'événement natif du navigateur, y compris stopPropagation() et preventDefault(), sauf que les événements fonctionnent de manière identique dans tous les navigateurs.

Exemple :

```
function EventHandler(e){
  e.preventDefault();
  e.target.style.color = 'red';
  console.log(e.target.innerHTML);
}

function Exemple() {
  return <button onClick={EventHandler}>
    Cliquer ici
  </button>
}
```

d. Passer des arguments a un évènement

Nous pouvons utiliser les fonctions fléchées pour passer des paramètres comme le montre l'exemple suivant :

```
const handleClick(event, arg1) {
  console.log(arg1)
}

return (
  <button onClick={(event)=>handleClick(event, 'hello')}> Click me </button>
);
```

II. Etat interne d'un composant

a. Introduction

L'état local (State) est une fonctionnalité fondamentale de React pour créer des composants dynamiques. L'état local n'est disponible que sur les composants de classe, et chaque composant gère son état. Vous pouvez définir la valeur initiale de l'état sur le constructeur du composant, et lorsque vous mettez à jour la valeur de l'état, le composant sera re-rendu lui-même.

State est un objet React intégré qui est utilisé pour contenir des données ou des informations sur le composant.

L'état d'un composant peut changer au fil du temps ; chaque fois qu'il change, le composant se présente à nouveau. Le changement d'état peut se produire en réponse à une action de l'utilisateur ou à des événements générés par le système et ces changements déterminent le comportement du composant et la façon dont il sera rendu.

Le tableau suivant montre une comparaison entre les props et state :

	State	Props
Utilisation	L'état est utilisé pour stocker les données des composants qui doivent être rendus dans la vue.	Les props sont utilisés pour transmettre des données aux composants enfants.
Mutabilité	State détient les données et peut changer au fil du temps	Les props sont immuables : une fois définis, ils ne peuvent plus être modifiés.
Composant	Utilise seulement dans les composants de types classe.	Les props peuvent etre utilise dans les classes et les fonctionnels composants.
Changement	Les gestionnaires d'événements mettent généralement à jour l'état.	Le composant parent définit les props pour les composants enfants.

b. Déclaration des States

L'objet state est initialise dans le constructeur du composant. L'exemple suivant montre la déclaration d'une valeur dans l'objet et son utilisation dans la méthode render :

```
class Voiture extends React.Component {
  constructor(props) {
    super(props);

    this.state = {
      marque:"Golf"
    };
  }

  render() {
    return <h1>Marque de la voiture : {this.state.marque}</h1>
  }
}
```

c. Changement des States

Pour modifier une valeur dans l'objet State, nous utilisons la méthode `this.setState()`. Lorsqu'une valeur de l'objet d'état change, le composant est rendu à nouveau, ce qui signifie que la sortie change en fonction de la ou des nouvelles valeurs.

Voici un exemple d'utilisation de la méthode this.setState :

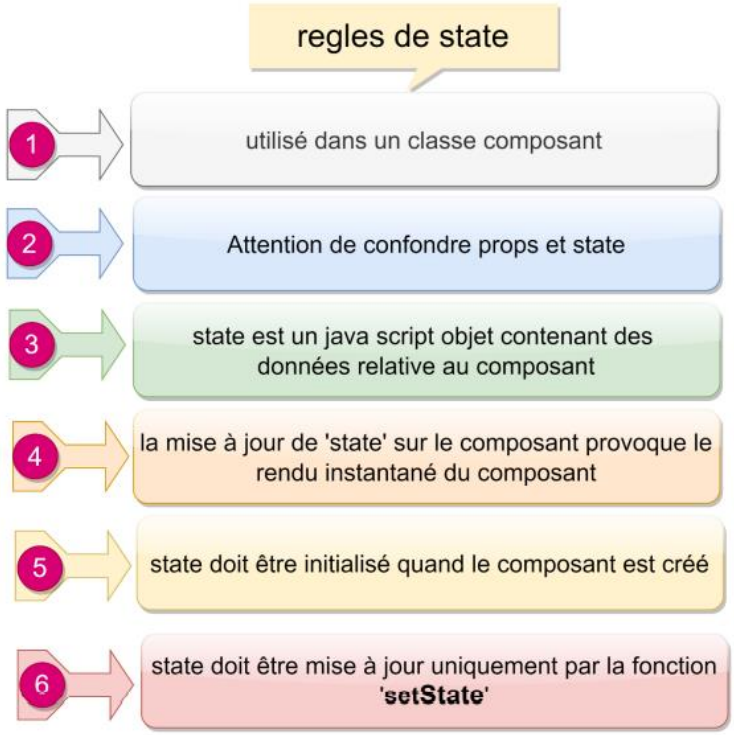
```
class Voiture extends React.Component {
  constructor(props){
    super(props);

    this.state = {
      marque:"Golf"
    };
    this.changerMarque = this.changerMarque.bind(this);
  }

  changerMarque(){
    this.setState({marque:"Ford"})
  }
  render() {
    return <>
      <h1>Marque de la voiture : {this.state.marque}</h1>
      <button onClick={this.changerMarque}>Changer marque</button>
    </>
  }
```

```
}
```

d. Les règles de state



e. Exemples

Voici un exemple d'utilisation de l'objet State dans un composant classe :

Exemple 1 :

```
class Exemple1 extends Component {
  constructor(props){
    super(props)
    this.state = {
      texte:'Mon texte'
    }
  }

  render() {
    return (
      <div>
        <p> {this.state.texte} </p>
        <button onClick={() => {this.setState({texte:'Texte Modified'})}
}>Cliquer ici</button>
      </div>
```

```
    );
  }
}
```

Exemple 2 :

```
class ClassComponentExample extends React.Component {
  constructor(props){
      super(props);
      this.state = {
          messageB1:"Default Message B - 1",
          messageB2:"Default Message B - 2",
      }
      this.b1OnClick = this.b1OnClick.bind(this);
      this.b2OnClick = this.b2OnClick.bind(this);
  }
  b1OnClick() {
      this.setState({
          messageB1:"B-1 Button Clicked :)"
      })
  }
  b2OnClick() {
      this.setState({
          messageB2:"B-2 Button Clicked :)"
      })
  }

  render() {
      return <div><h1>{this.props.message}</h1>
          <h1>{this.state.messageB1}</h1>
          <h1>{this.state.messageB2}</h1>
          <button onClick={this.b1OnClick}>B1 Button</button>
          <button onClick={this.b2OnClick}>B2 Button</button>
      </div>
  }
}
```