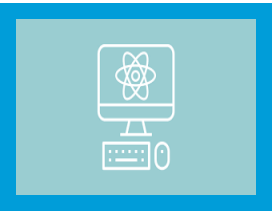


Gestion des formulaires

Comment React gère les formulaires ?

- La gestion des formulaires concerne la façon dont on gère les données lorsqu'elles changent de valeur ou sont soumises.
- En HTML, les données du formulaire sont généralement gérées par le DOM.
- Dans React, les données de formulaire sont généralement gérées par les **composants.**
- Lorsque les données sont gérées par les composants, toutes les données sont stockées dans le composant **state.**
- On peut contrôler les modifications en ajoutant des gestionnaires d'événements.

Gestion des événements

- Tout comme HTML, React peut effectuer des actions en fonction des événements utilisateur.
- React a les mêmes événements que HTML : onClick, onChange, onMouseOver, etc.

Exemple :

```
<button onClick={this.handleClick}>Cliquer</button>

handleClick(){
 alert("Bonjour à toutes et à tous");
}
```

NB : En React, l'appel de la fonction se fait plutôt comme suit : **{this.handleClick}**

Et non pas : ~~{this.handleClick()}~~

Composants de formulaires en React

- Tout comme en HTML, React utilise des formulaires pour permettre aux utilisateurs d'interagir avec la page Web.
- Au niveau des formulaires en React, on utilise simplement value= quel que soit le type du composant.
- La soumission de formulaire dans React.js est assez différente des soumissions de formulaire habituelles en HTML.
- React.js donne un contrôle total des valeurs qu'on passe à l'élément exploitable suivant, permettant de formater des structures de données plus complexes.
- Toutes les valeurs sont conservées par l'objet d'état d'un composant et sont propagées dans tous les éléments rendus, tels que ceux d'une entrée.

Ajouter un formulaire avec React

- Composant de type classe

```
import React,{ Component }  from "
react";
class MyForms extends Component{
render()
    {
        return(
            <form>
       <h1>Donner Votre nom:</h1>
       <input type="text"/>

            </form>
        );
    }
}
export default MyForms;
```

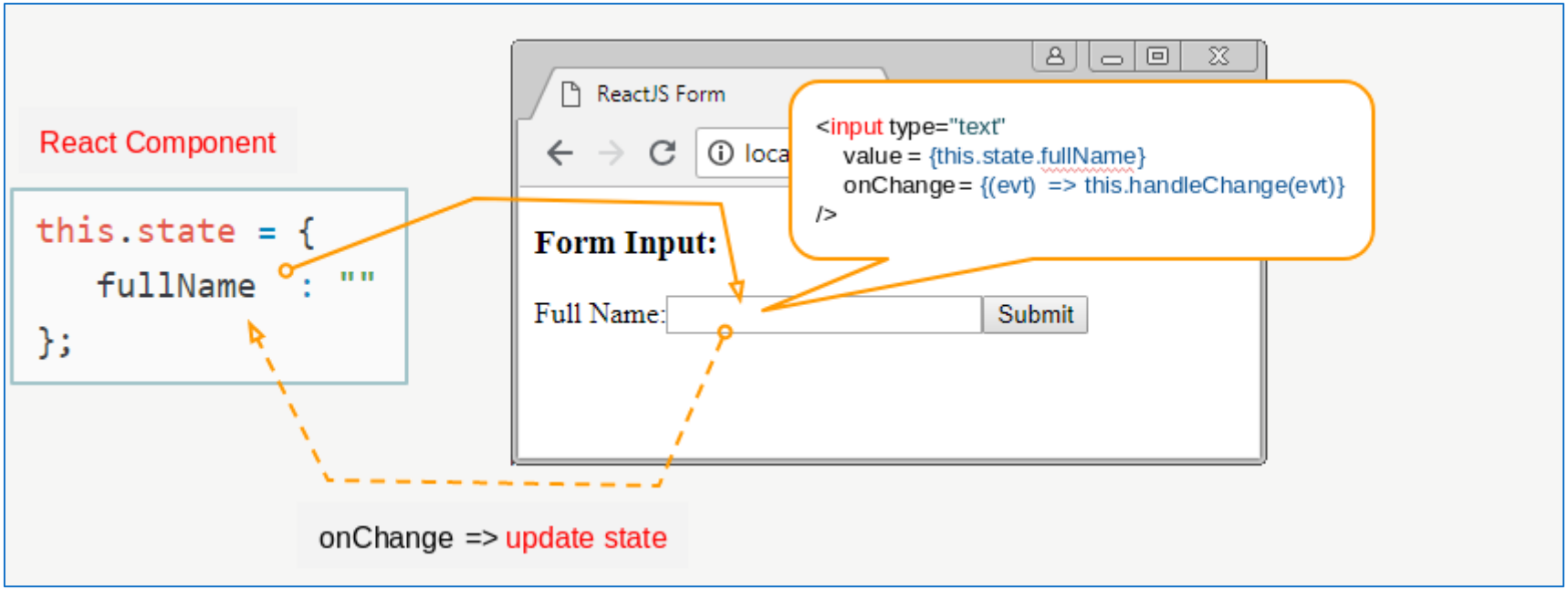
- Composant fonctionnel

```
const MyForms=()=> {
        return(
            <form>
     <h1>Donner Votre nom:</h1>
     <input type="text"/>

            </form>
        );
    }
export default MyForms;
```

```jsx
import IdentForms from  './Components/MyForms'
function App() {
  return (
    <div>
      <IdentForms/>
      </div>
  );
}

export default App;
```

Ajouter un champ input avec les classes

- Ci-dessous, l'exemple simple avec l'élément <input>. La valeur de cet élément est assignée de this.state.fullName. Lorsque les utilisateurs changent la valeur de <input>, cette valeur a besoin d'être mise à jour pour this.state.fullName via la méthode setState().

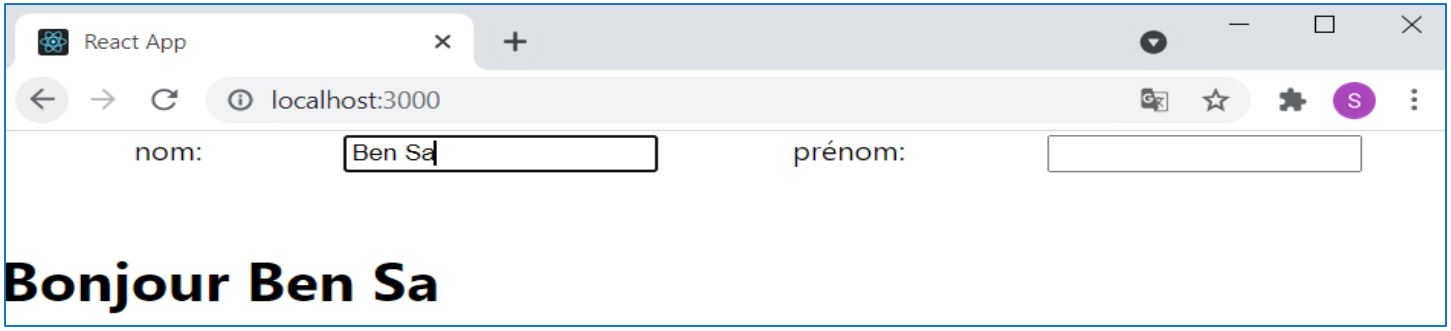


Gestionnaires d'événements dans le formulaire

```
import React,{ Component }  from "react";
class MyForms extends Component{
    constructor(props){
        super(props);
            this.state =
            { nom: '' ,
              prenom:''
            };
                }
changerStateNom = (event) => {
    this.setState({nom: event.target.value});
  }
  changerStatePrenom = (event) => {
    this.setState({prenom: event.target.value});
  }
```

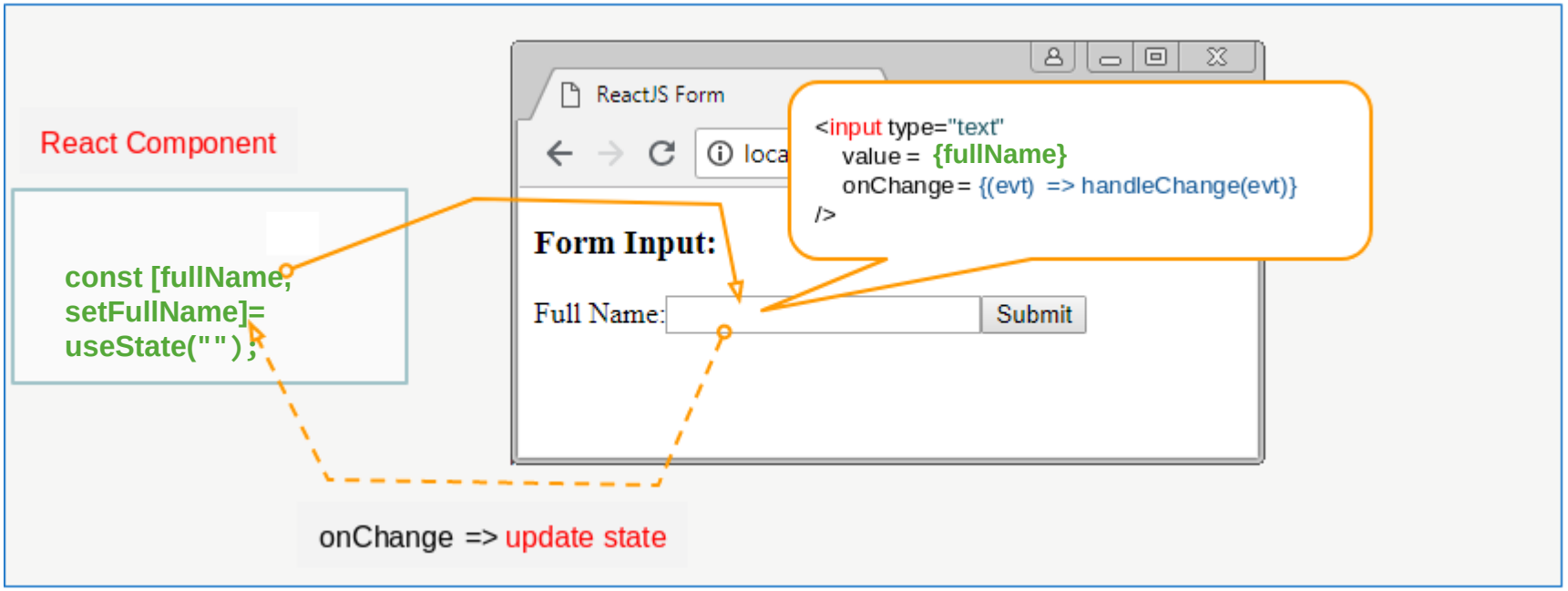
Accès à la valeur du champ : ***event.target.value***

```
render()
    {
        return(
            <form>
                nom:
                <input type="text"
                onChange={this.changerStateNom}/>
                prénom:
                <input type="text"
                onChange={this.changerStatePrenom}/>
               <h1>Bonjour {this.state.nom} {this.state.prenom} </h1>
            </form>
        );
    }
}
export default MyForms;
```



Ajouter un champ input avec les Hooks

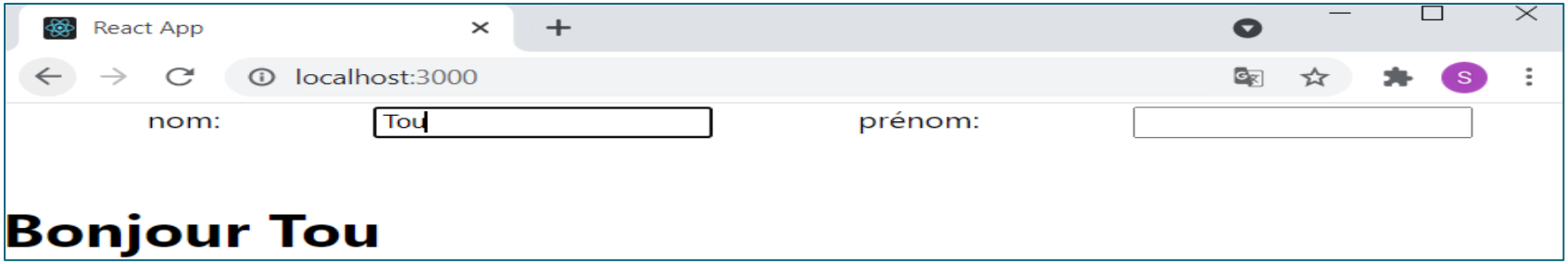
- Ci-dessous, l'exemple simple avec l'élément <input>. La valeur de cet élément est assignée de **fullName**. Lorsque les utilisateurs changent la valeur de <input>, cette valeur a besoin d'être mise à jour pour **setFullName**.



Gestionnaires d'événements avec les Hooks

```
import React,{useState}  from "react";
const MyForms=()=> {
   const[nom,setNom]=useState('');
   const[prenom,setPrenom]=useState('');

const changerStateNom = (event) => {
        setNom(event.target.value);
  }
const changerStatePrenom = (event) => {
        setPrenom(event.target.value);
  }
```

```
return(
            <form>
                nom:
                <input type="text"
                onChange={changerStateNom}/>
                prénom:
                <input type="text"
                onChange={changerStatePrenom}/>
                <h1>Bonjour {nom} {prenom} </h1>
            </form>
        );
 }
export default MyForms;
```



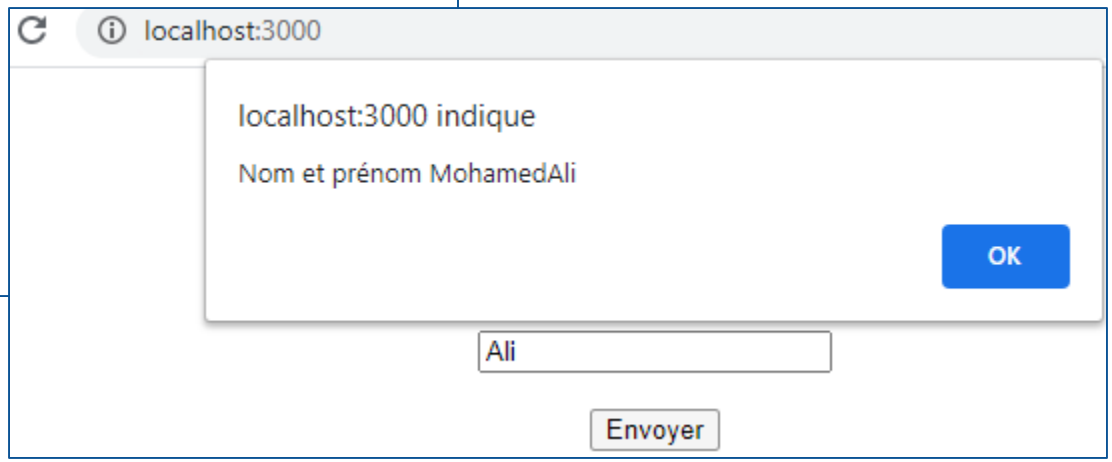
Soumettre un formulaire

- Contrôler l'action de soumission en ajoutant un gestionnaire d'événements dans l'attribut onSubmit.

```
import React,{ Component }  from "react";
class MyForms extends Component{
    constructor(props){
        super(props);
            this.state =
            { nom: '' ,
              prenom:''
            };
                }
changerStateNom = (event) => {
    this.setState({nom: event.target.value});
  }
  changerStatePrenom = (event) => {
    this.setState({prenom: event.target.value});
  }
```

```
mySubmitHandler = (event) => {
    event.preventDefault();
    alert("Nom et prénom " +
          this.state.nom +
         this.state.prenom);
  }
  render()
    {
   return(
     <form onSubmit={this.mySubmitHandler}>
        <p>nom:</p>
        <input type="text"
        onChange={this.changerStateNom}/>
        <p>prénom:</p>
        <input type="text"
        onChange={this.changerStatePrenom}/>
        <p> <input type='submit' /></p>
        </form>
    );
    }
}
export default MyForms;
```

event.preventDefault :
Empêcher le rechargement / rafraîchissement du navigateur



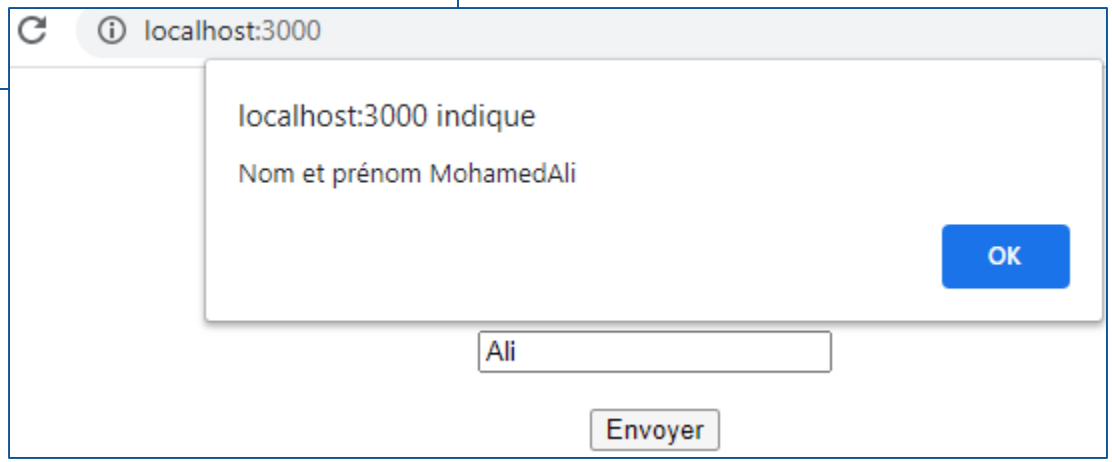
Soumettre un formulaire avec les Hooks

- Contrôler l'action de soumission en ajoutant un gestionnaire d'événements dans l'attribut onSubmit.

```
import React,{useState}  from "react";
const MyForms=()=> {
   const[nom,setNom]=useState('');
   const[prenom,setPrenom]=useState('');

const changerStateNom = (event) => {
        setNom(event.target.value);
  }
const changerStatePrenom = (event) => {
        setPrenom( event.target.value);
  }
```

```
const mySubmitHandler = (event) => {
    event.preventDefault();
    alert("Nom et prénom " +
        nom + prenom);
  }
   return(
     <form onSubmit={mySubmitHandler}>
        <p>nom:</p>
        <input type="text"
        onChange={changerStateNom}/>
        <p>prénom:</p>
        <input type="text"
        onChange={changerStatePrenom}/>
        <p> <input type='submit' /></p>
        </form>
    );
   }
export default MyForms;
```

event.preventDefault :
Empêcher le rechargement / rafraîchissement du navigateur

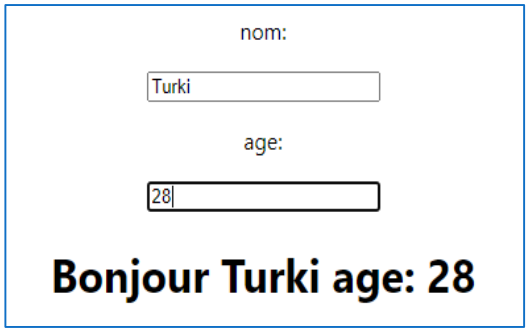


Plusieurs champs dans un formulaire React

- Vous pouvez contrôler les valeurs de plusieurs champs d'entrée en ajoutant un attribut de nom à chaque élément.
- Lorsque vous initialisez l'état dans le constructeur, utilisez les « names » des champs.
- Pour accéder aux champs du gestionnaire d'événements, utilisez la syntaxe **event.target.name** et **event.target.value**.
- Pour mettre à jour l'état, utilisez des crochets [notation entre crochets] autour du nom de la propriété.

```
import React,{ Component }  from "react";
class MyForms extends Component{
    constructor(props){
        super(props);
            this.state =
            { nom: '' ,
              age:''
            };
                }
changerState = (event) => {
    let namefields=event.target.name;
     let valfields=event.target.value;
    this.setState({ [namefields] : valfields});
  }
 render()
    {
 return(
 <form >
     <p>nom:</p>
      <input type="text"
       name="nom"
       onChange={this.changerState}/>
     <p>age:</p>
      <input type="text"
       name="age"
       onChange={this.changerState}/>
 <h1>Bonjour {this.state.nom} age: {this.state.age}</h1>
 </form>
   );
    }
}export default MyForms;
```

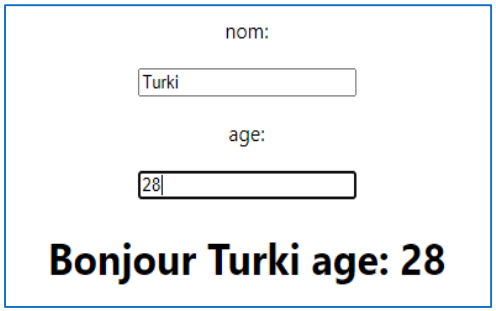
Avec les classes



```jsx
import React,{useState}  from "react";
const MyForms=()=> {

const [{ nom, age},setState] = useState('');

const changerState = (event) => {
 let namefields=event.target.name;
 let valfields=event.target.value;
 setState((prevState) => ({ ...prevState, [namefields] : valfields }));
       }
return(
<form >
 <p>nom:</p>
 <input
    type="text"
    name="nom"
    onChange={changerState}/>
 <p>age:</p>
  <input
    type="text"
    name="age"
    onChange={changerState}/>
 <h1>Bonjour {nom} age: {age}</h1>
 </form>
    );
       }
export default MyForms;
```

Avec les Hooks

prevState est un nom donné à l'argument passé à la fonction de rappel setState. Ce qu'il contient est la valeur de state avant que le setState ne soit déclenché par React



TextArea

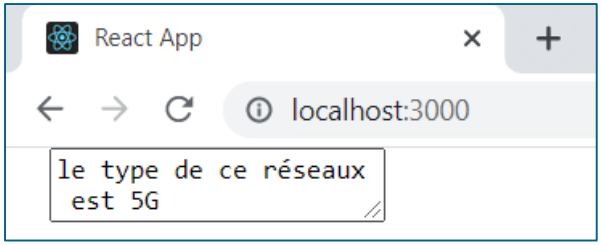
- Une zone de texte simple avec du contenu initialisé dans le constructeur :

- Composant de type classe

```
import React,{ Component }  from "react";
class MyForms extends Component{
  constructor(props){
    super(props);
      this.state =
        {
description:'le type de ce réseaux est 5G';
        };
}
render()    {
  return(
   <form >
     <textarea
      value={this.state.description} />
   </form>
       );
    }
}
export default MyForms;
```

- Composant fonctionnel

```
import React,{useState}  from "react";
const MyForms=()=> {

const[description,setDescription]=
useState('le type de ce réseaux est 5G');

  return(
   <form >
     <textarea value={description} />
   </form>
       );
}
export default MyForms;
```



Select

- Sélectionner Une liste déroulante, ou une zone de sélection, dans React est également un peu différente du HTML.

- Composant de type classe

```
import React,{ Component }  from "react";
class MyForms extends Component{
    constructor(props){
        super(props);
          this.state =
            {mavoiture: 'Volvo'};
}
render()    {
 return(
   <form >
   <select
   value={this.state.mavoiture}>
     <option value="Ford">Ford</option>
     <option value="Volvo">Volvo</option>
     <option value="Fiat">Fiat</option>
    </select>
    </form>
        );
    }
} export default MyForms;
```

- Composant fonctionnel

```
import React,{useState}  from "react";
const MyForms=()=> {

const [mavoiture,setMavoiture]=useState('Volvo')
 return(
   <form >
   <select
   value={mavoiture}>
     <option value="Ford">Ford</option>
     <option value="Volvo">Volvo</option>
     <option value="Fiat">Fiat</option>
    </select>
    </form>
        );
}
export default MyForms;
```