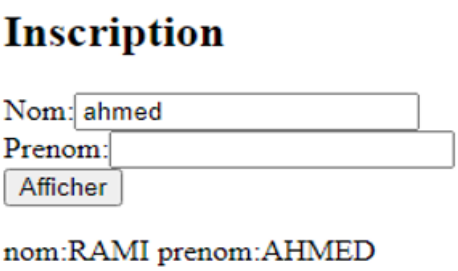
Exercice d'application 1



Quand l'utilisateur clique sur Afficher s'affiche le message contenant le nom et le prenom

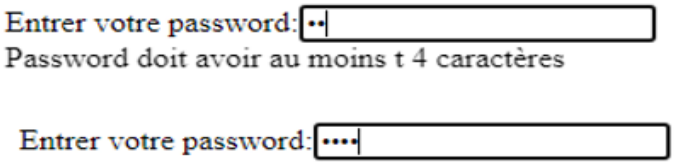
-ajouter si vous voulez la verfication que les nom, prenom ne contient que des caracteres, en utilisant les expressions reguliere ,et un messageError qui s'affiche si on click sur le button d'envoi

Exercice Application 2 :

On peut valider les éléments input facilement en utilisant le Système gestion d'état de Rect.

Essayons ça en créant un simple validateur de mot de passe. Ce dernier va être un input text qui nécessite l'utilisateur d'entrer un mot de passe qui contient au moins 4 caractères.

Si l'utilisateur entre un mot de passe moins de 4 caractères. un message d'erreur va être affiché «**Password doit avoir au moins de 4 caractères**"



Exercice d'application 3:

-Le but et de créer un composant <Select/> pour manipuler la liste déroulante que que vous aviez utiliser en HTML

1- Commençant par index.js ,options est un tableau de valeur recuperer dans le composant avec **props.options**(vous l'appelez comme vous voulez):

Index.js

```
const root = ReactDOM.createRoot(document.getElementById('root'));
root.render(
  <div>
   <Select options={["Element1", "Element2", "Element3", "Element4", "Element5"]}/>
</div>

);
```

2-Créer Le composant Select dans un fichier select.js on a le choix d'utiliser format class , ou format function :

Composant class	Composant fonctionnel

class Select extends React.Component
{
constructor(props) {
super(props); }
handlerChange(event) {
let i =event.target.value
let text=event.target.options[i].text
console.log('indice='+i+" text de option="+text);
}
render() {
return (
<select onChange={this.handlerChange}>
{this.props.options.map((v, i) =>{
return <option key={i} value={i}>{v}</option>
})
}
</select>
) }
}
export default Select;
	const Select = (props) => {
	const handlerChange=(event)=>{
	let i =event.target.value
	let text=event.target.options[i].text
	console.log('indice='+i+" text de option="+text+" ");
	}
	return (<select onChange={handlerChange}>
	{
	props.options.map((x, i)=><option key={i}
	value={i}>{x}</option>
	})
	}
	</select>
	); };
	
	export default Select;

3-la fonction HandleChange permet de récupérer la valeur sélectionné ainsi que son indice dans tableau options[]

```
const handlerChange=(event)=>{
        let i =event.target.value
        let text=event.target.options[i].text
        console.log('indice='+i+" text de option="+text+" ");
    }
```

Exercice d'application 3 :

-Composant pour Manipuler une Textarea

```
function FTextArea(props) {
    const [message, setMessage] = useState(props.value);
    const handlerChange=(event)=>{
        setMessage(event.target.value);
    }
  const handlerFocus=()=>{
   setMessage("");      }
  return (<>
  message=:{message}
  <textarea cols={props.cols}
   rows={props.rows}
   value={message}
       onFocus={handlerFocus}
   onChange={handlerChange.bind(this)} />    </>
 );     };
```

Exercice d'application 4 :

1-Créer un composant type function Checkbox simple qui return un checkbox avec sa valeur et text associe , et état coché ou non :

```
<CheckBox   text='francais' value='fr' checked={true}/>
```

-Exemple de Composant avec class

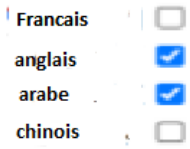
```
//<CheckBox  value="ar" text="arabe" checked={true} />
class CheckBox extends React.Component {
    constructor(props) {
        super(props);
        this.state = {
            checked : props.checked || false };
        }
   handlerChange(event) {
       this.setState({checked : event.target.checked});
   }
   render() {
       return (<label><span>{this.props.text}</span>
       <input type="checkbox" value={this.props.value}
       checked={this.state.checked}
       onChange={this.handlerChange.bind(this)} />
    <br/>
    --etat={this.state.checked?"true":"false"}</label>
   )
       }
   }
```

2-soit le tableau T

```
let  T = [
    { value : 1, text : "francais",checked:false },
    { value : 2, text : "anglais", checked : true },
    { value : 3, text : "arabe", checked : true },
    { value : 4, text : "chinois" ,checked:true}   ];
```

-Créer le composant function <CheckBoxGroup /> qui prend le tableau T en Propriété et afficher l'ensemble des checkbox(voir image) , Utiliser le Composant function Question2 <Checkbox /> dans le map du tableau T, vous avez une solution de code proposé

-appel du composant sera comme :<CheckBoxGroup T={T} />
-résultat voulu :



Code de `CheckBoxGroup.js`

```
function CheckBoxGroup(props){
    let [LangueChoix,setLangueChoix]=useState([])
    let AfficherCaseCoche=(event)=>{event.preventDefault()
    //ici remplir le tableau des langues choisi dans LangueChoix
    }
    return (<form onSubmit={AfficherCaseCoche}    >
        {props.T.map((x, i) =><CheckBox key={i} text={x.text} value={x.value}
                checked={x.checked}/>)
        }
        <button>Envoyer</button>
        {LangueChoix.map((x,i)=><span key={i}>{x}</span>)}
        </form> )
}
```

3 -Dans le form on a `<form onSubmit={AfficherCaseCoche}  >` Completez function AfficherCaseCoche pour remplir le tableau des langues choisie et aussi afficher les dans l écran

Exercice d'application 5: Gérer les boutons radio

1-Dans notre index.js on aura remarquer le passage par **props** du tableau qui contient les données nécessaire au radiobutton group

```
import RadioGroup from './RadioGroup';
let T = [   { value : 1, text : "radio1" },
            { value : 2, text : "radio2" },
            { value : 3, text : "radio3", checked : true },
            { value : 4, text : "radio4" } ];


const root = ReactDOM.createRoot(document.getElementById('root'));
root.render(<FRadioGroup T={radios} name="group1" />);
```

2-le Composant <FRadioGroup …/>

```
//RadioGroup.js
 function FRadioGroup(props) {
    return (
        <div>
            {
            props.T.map((radio, index) => {
                        return (
                                <label key={index+1}><span>{x.text}</span>
                                <input type="radio" value={x.value} name={props.name} />
                                </label>
                            )
                            })
            }
</div>
);
};
 export default FRadioGroup;
```