

Objectifs :

- Valider un formulaire en utilisant les composants de class.
- Valider un formulaire en utilisant les composants fonctionnels.

Le seconde paramètre de la méthode setState

Si vous avez besoin d'exécuter une fonction, ou de vérifier si l'état a bien été mis à jour correctement, vous pouvez passer une fonction comme deuxième argument de l'appel setState, la fonction sera exécutée une fois l'état mis à jour.

Exemple :

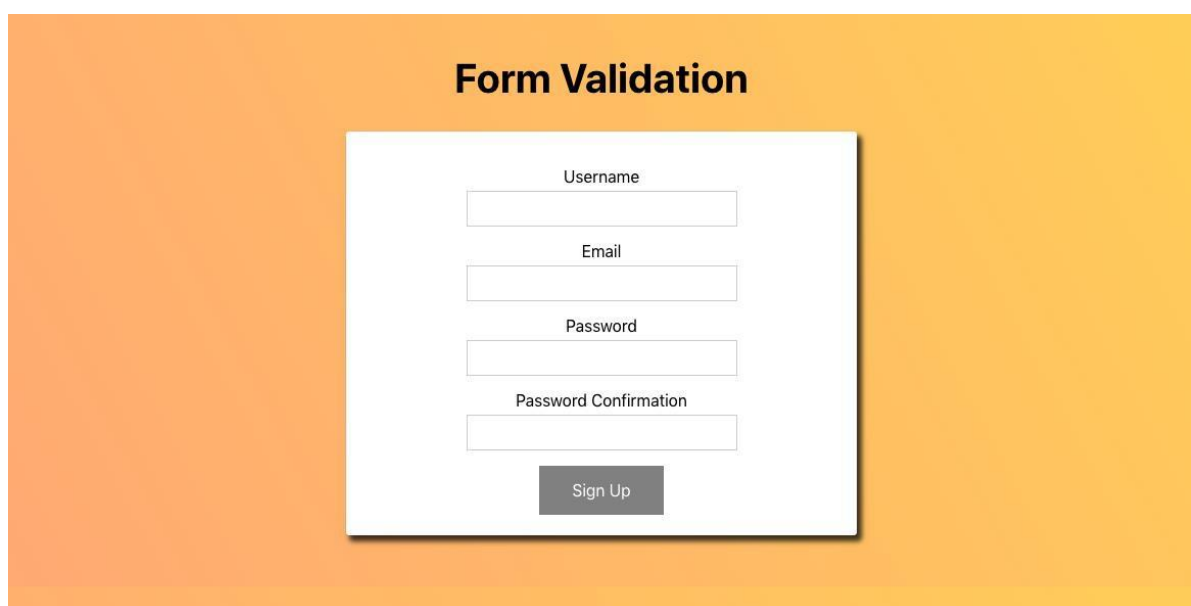
```
class App extends Component {
  this.state = {
    sampleItem: 'test',
  }

  handleChange = (event) => { console.log(this.state.sampleItem)

    this.setState({ sampleItem: event.target.value}, () =>
      console.log(this.state.sampleItem))
  };
}
```

Exercice 1 : Valider un formulaire en utilisant les composants de class

1. Créer un composant sous classe classe qui retourne le formulaire suivant :



The image shows a web form titled "Form Validation" centered on an orange background. The form itself is a white box with a shadow. It contains four text input fields stacked vertically, each with a label above it: "Username", "Email", "Password", and "Password Confirmation". Below these fields is a grey button with the text "Sign Up".

2. Compléter le code de ce formulaire pour réaliser une validation :



```
import React from "react";
import "./app1.css";

function ValidationMessage(props) {
  if (!props.valid) {
    return <div className="error-msg">{props.message}</div>;
  }
  return null;
}

class MyForm extends React.Component {
  state = {
    username: "",
    usernameValid: false,
    email: "",
    emailValid: false,
    password: "",
    passwordValid: false,
    passwordConfirm: "",
    passwordConfirmValid: false,
    formValid: false,
    errorMsg: {},
  };

  validateForm = () => {
    const { usernameValid, emailValid, passwordValid, passwordConfirmValid } = this.state;
    this.setState({ formValid:
      usernameValid && emailValid && passwordValid && passwordConfirmValid,
    });
  };

  updateUsername = (username) => {
    this.setState({ username }, this.validateUsername);
  };

  validateUsername = () => {
    const { username } = this.state;
    let usernameValid = true;
    let errorMsg = { ...this.state.errorMsg };

    if (username.length < 3) {
      usernameValid = false;
      errorMsg.username = "Must be at least 3 characters long";
    }
  };
}
```



```
this.setState({ usernameValid, errorMsg }, this.validateForm);
};

updateEmail = (email) => {
  this.setState({ email }, this.validateEmail);
};

validateEmail = () => {
  const { email } = this.state;
  let emailValid = true;
  let errorMsg = { ...this.state.errorMsg };

  // checks for format _@._
  if (!/^[^\s@]+@[^\s@]+\.[^\s@]+$/.test(email)) {
    emailValid = false;
    errorMsg.email = "Invalid email format";
  }

  this.setState({ emailValid, errorMsg }, this.validateForm);
};

updatePassword = (password) => {
  this.setState({ password }, this.validatePassword);
};

validatePassword = () => {
  const { password } = this.state;
  let passwordValid = true;
  let errorMsg = { ...this.state.errorMsg };

  // must be 6 chars
  // must contain a number
  // must contain a special character

  if (password.length < 6) {
    passwordValid = false;
    errorMsg.password = "Password must be at least 6 characters long";
  } else if (!/\d/.test(password)) {
    passwordValid = false;
    errorMsg.password = "Password must contain a digit";
  } else if (!/[!@#$%^&*]/.test(password)) {
    passwordValid = false;
    errorMsg.password = "Password must contain special character: !@#$%^&*";
  }
}
```



```
this.setState({ passwordValid, errorMsg }, this.validateForm);
};

updatePasswordConfirm = (passwordConfirm) => {
  this.setState({ passwordConfirm }, this.validatePasswordConfirm);
};

validatePasswordConfirm = () => {
  const { passwordConfirm, password } = this.state;
  let passwordConfirmValid = true;
  let errorMsg = { ...this.state.errorMsg };

  if (password !== passwordConfirm) {
    passwordConfirmValid = false;
    errorMsg.passwordConfirm = "Passwords do not match";
  }

  this.setState({ passwordConfirmValid, errorMsg }, this.validateForm);
};

render() {
  return (
    <div className="App">
      <main role="main">
        <header>Form Validation</header>
        <form action="#" id="js-form">
          <div className="form-group">
            <label htmlFor="username">Username</label>
            <ValidationMessage
              valid={this.state.usernameValid}
              message={this.state.errorMsg.username}
            />
            <input
              type="text"
              id="username"
              name="username"
              className="form-field"
              value={this.state.username}
              onChange={(e) => this.updateUsername(e.target.value)}
            />
          </div>
          <div className="form-group">
            <label htmlFor="email">Email</label>
```



```
<ValidationMessage
  valid={this.state.emailValid}
  message={this.state.errorMsg.email}
/>
<input
  type="email"
  id="email"
  name="email"
  className="form-field"
  value={this.state.email}
  onChange={(e) => this.updateEmail(e.target.value)}
/>
</div>
<div className="form-group">
  <label htmlFor="password">Password</label>
  <ValidationMessage
    valid={this.state.passwordValid}
    message={this.state.errorMsg.password}
  />
  <input
    type="password"
    id="password"
    name="password"
    className="form-field"
    value={this.state.password}
    onChange={(e) => this.updatePassword(e.target.value)}
  />
</div>
<div className="form-group">
  <label htmlFor="password-confirmation">
    Password Confirmation
  </label>
  <ValidationMessage
    valid={this.state.passwordConfirmValid}
    message={this.state.errorMsg.passwordConfirm}
  />
  <input
    type="password"
    id="password-confirmation"
    name="password-confirmation"
    className="form-field"
    value={this.state.passwordConfirm}
    onChange={(e) => this.updatePasswordConfirm(e.target.value)}
  />
</div>
```



```
        </div>
        <div className="form-controls">
          <button
            className="button"
            type="submit"
            disabled={!this.state.formValid}
          >
            Sign Up
          </button>
        </div>
      </form>
    </main>
  </div>
);
}
}

export default MyForm;
```

Exercice 2 : Valider un formulaire en utilisant les composants fonctionnels

Proposez un composant fonctionnel de validation du formulaire précédents en utilisant les Hooks `useState()` et `useEffect()` ;