**Stateless Functional Components**

Dans l'éditeur de code, jetez un œil à Example.js. Le premier composant Example est défini comme une classe JavaScript, mais ce n'est pas obligatoire ! Dans React, nous pouvons également définir des composants comme des fonctions JavaScript — nous les appelons des composants de fonction pour les différencier des composants de classe.

Dans les dernières versions de React, les composants de fonction peuvent faire tout ce que les composants de classe peuvent faire. Dans la plupart des cas, cependant, les composants fonctionnels offrent un moyen plus élégant et concis de créer des composants React. Cette leçon se concentrera sur la conversion d'un composant de classe en un composant de fonction et sur l'ajout des props, qui sont disponibles dans toutes les versions de React.

Comparez le composant de classe Example et le composant de fonction Example. Pour les composants de fonction les plus basiques, tout ce que vous avez à faire est de supprimer le début render() { et la fin } de la méthode render() :

render() { // Delete this  
  return <h1>Hello</h1>  
} // Delete this

En d'autres termes : le composant de fonction doit renvoyer le même JSX que celui renvoyé à l'origine par la méthode render().

Question

In the context of this [exercise 3](https://www.codecademy.com/paths/web-development/tracks/front-end-applications-with-react/modules/react-102-advanced-react-u/lessons/stateless-functional-components/exercises/stateless-functional-component-intro), is a stateless functional component the same as a stateless component?

Answer

Although similar in some aspects, a stateless functional component is not the same as a stateless component. Some of the differences are as follows:

A stateless functional component can be seen just as a function which returns JSX based on the prop values that are passed to it. It is not a class and does not extend the React.Component class. It also has no state.

The general structure typically looks like the following

const Example = (props) => {

//

}

A stateless component is a class and extends the React.Component class. It has no state and no life-cycle unlike regular stateful components. These are typically used by a parent component for rendering something, like text or information, without the need for it to perform any actions.

The general structure is similar to a regular stateful component, but usually just has a render() method,

class Example extends React.Component {

render() {

//

}

}

**Function Components and Props**

Comme tout composant, les composants de fonction peuvent recevoir des informations via des props.

Pour accéder à ces props, attribuez à votre composant de fonction un paramètre nommé props. Dans le corps de la fonction, vous pouvez accéder aux props à l'aide de ce modèle : props.propertyName. Vous n'avez pas besoin d'utiliser le mot-clé this.

export function YesNoQuestion (props) {  
  return (  
    <div>  
      <p>{props.prompt}</p>  
      <input value="Yes" />  
      <input value="No" />  
    </div>  
  );  
}  
  
ReactDOM.render(  
  <YesNoQuestion prompt="Have you eaten an apple today?" />,  
  document.getElementById('app');  
);

Dans l'exemple ci-dessus, nous passons une valeur de " Have you eaten an apple today?" comme props d'invite lors du rendu de YesNoQuestion.

### Question

In the context of this [exercise 21](https://www.codecademy.com/paths/web-development/tracks/front-end-applications-with-react/modules/react-102-advanced-react-u/lessons/stateless-functional-components/exercises/stateless-functional-component-props), can a stateless functional component have more parameters, other than props and state?

### Answer

No, stateless functional components in React can only accept props and state as optional parameters. As a result, you cannot do something like the following, which adds other parameters other than props or state,

export const MyComponentClass = (props, param2, param3) => {

//

}

If you need to pass any values to the component, you should always do so by passing them as props, without the need for adding other parameters.

**Review**

Bien joué! Vous avez écrit votre premier composant de fonction. Voici un récapitulatif :

* **Les composants de fonction** sont des composants React définis comme des fonctions JavaScript.
* **Les composants de la fonction** doivent retourner JSX.
* **Les composants de fonction** peuvent accepter un paramètre props. Attendez-vous à ce que ce soit un objet JavaScript.

Bien que les composants de fonction et les composants de classe puissent faire les mêmes choses, vous verrez beaucoup de composants de fonction dans la documentation de React et des exemples d'applications. Certains développeurs les préfèrent aux composants de classe pour leur simplicité et leurs fonctionnalités simples, comme Hooks, que vous apprendrez plus tard dans votre parcours de codage.

Regardez l'exemple dans l'éditeur de code. Assurez-vous que le code a du sens avant de continuer !

// A component class written in the usual way:

export class MyComponentClass extends React.Component {

  render() {

    return <h1>Hello world</h1>;

  }

}

// The same component class, written as a stateless functional component:

export const MyComponentClass = () => {

  return <h1>Hello world</h1>;

}

// Works the same either way:

ReactDOM.render(

  <MyComponentClass />,

  document.getElementById('app')

);

import React from 'react';

import ReactDOM from 'react-dom';

// <Friend /> as function component

export const Friend = () => {

  return <img src='https://content.codecademy.com/courses/React/react\_photo-octopus.jpg' />;

}

// <Friend /> as class component

// export class Friend extends React.Component {

//  render() {

//    return <img src='https://content.codecademy.com/courses/React/react\_photo-octopus.jpg' />;

//  }

// }

ReactDOM.render(

  <Friend />,

  document.getElementById('app')

);