Componente:

Es un trozo de pantalla o la parte lógica que controla un trozo de pantalla, podríamos

Tener un componente que represente cada parte de la página, es al final del día un trozo o parte lógica de la aplicación, todo lo que podemos ver es controlado y gestionado por los componentes, lo que hace es recibir unos parámetros o props, lo que hace el componente es procesar una lógica hacer una funcionalidad.

El componente da soporte a la UI o a la vista dicho más sencillo, interactuaremos con la plantilla que representa por medio de cierta lógica que el componente va a estar gestionando, el componente hace de mediador entre la vista y la lógica de la aplicación, pide datos los trabaja y se los devuelve a la plantilla o a la vista.

Como crear un componente desde 0 con React.

Una forma prolija de crear los componentes es crear una carpeta de components y una carpeta por cada componente, lo cual es más organizado, lo mejor es tenerlo separado.

Y crear una carpeta assets en la cual vayamos colocando en subcarpetas los css, las imágenes, entre otros.

Los componentes se ponen en mayúscula, utilizando la convención camel case.

Importamos el objeto React.

Lo que hacemos es:

Escribimos class nombreDelComponente extends React.Component, para crear un nuevo

Componente.

Es necesario crear el método render, que es justamente el que se va a encargar de mostrar la vista en pantalla de renderizarla, de mostrar el HTML que se generará.

El método debe llevar un return y dentro del mismo estará el JSX que va a mostrar este componente.

Usamos el Export Default para poder exportar el componente.

Luego si queremos verlo por ejemplo en APP.js debemos importarlo.

Otra forma mas limpia es utilizar el structuring

Import {Component}…

Además es buena idea no utilizar etiquetas html para retornar, podemos utilizar un fragmento React

Y devolverlo, no crea ninguna etiqueta extra, simplemente es para saltarnos la restricción.

    render() {

        return (

            <React.Fragment>

                < h1 > Hola, soy el componente MiComponente</h1 >

            </React.Fragment>

        )

    }

También se puede utilizar un div, la diferencia es que en este caso no tenemos ninguna etiqueta, depende de que es lo que queremos devolver.

Podemos tener un objeto dentro y luego imprimirlo.

  render() {

        let receta = {

            nombre: "Pizza",

            ingredientes: ["Tomate", "Queso", "Harina"],

            calorias: 400

        };

        return (

            <React.Fragment>

                < h1 > Hola, soy el componente MiComponente</h1 >

            </React.Fragment>

        )

    }

Podemos iterar la lista de ingredientes

        return (

            <React.Fragment>

                <h2>{'Receta'+ receta.nombre + 'Ingredientes' + receta.ingredientes}</h2>

                < h1 > Hola, soy el componente MiComponente</h1 >

                {receta.ingredientes.map((ingrediente, i)=>{

                    return (

                        <li>{ingrediente}</li>

                    )

                })}

            </React.Fragment>

        )

Podemos ponerle la key, en realidad debemos por si por ejemplo cambiamos dinámicamente esas propiedades, con la key si se actualiza alguno de los valores, que se puedan actualizar automáticamente de forma reactiva y dinámica en la vista, que se modiique el dom.

        return (

            <React.Fragment>

                <h2>{'Receta'+ receta.nombre + 'Ingredientes' + receta.ingredientes}</h2>

                < h1 > Hola, soy el componente MiComponente</h1 >

                {receta.ingredientes.map((ingrediente, i)=>{

                    return (

                        <li key="{i}">{ingrediente}</li>

                    )

                })}

            </React.Fragment>

        )

Puede haber componentes dentro de otros componentes

Se pueden crear componentes estáticos, sin estados, funcionales.

import React, {Component} from 'react';

const MensajeEstatico = () => {

    return(<h1>Hola soy el mensaje estatico</h1>)

}

export default MensajeEstatico;

Los comentarios se escriben {/\*\*/}

Importamos las imágenes como de costumbre.