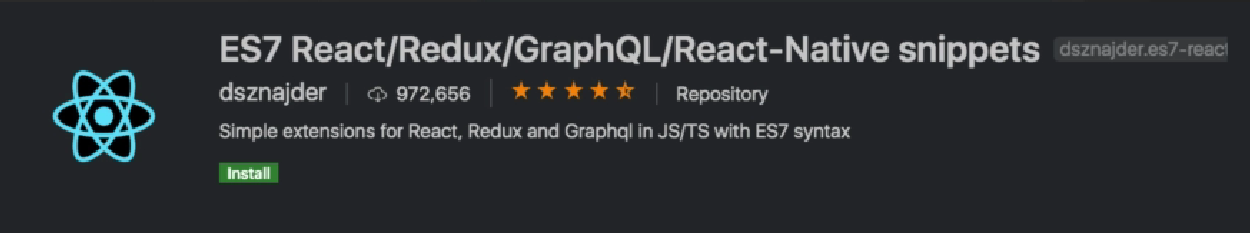
**Notas Curso La guía de React.**

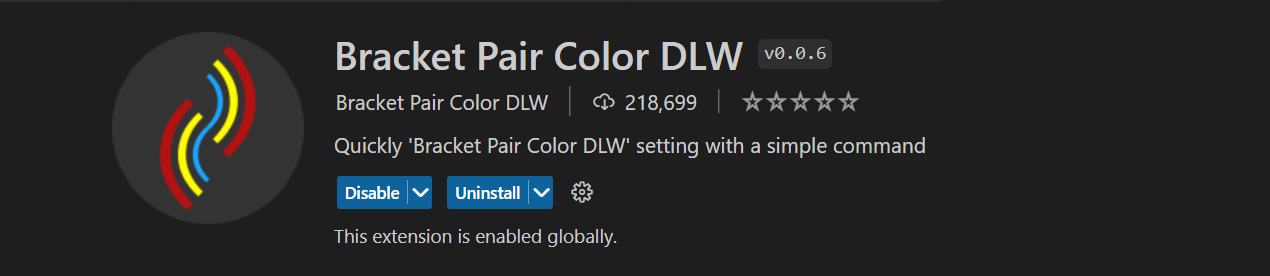
**Parte: Básicos.**

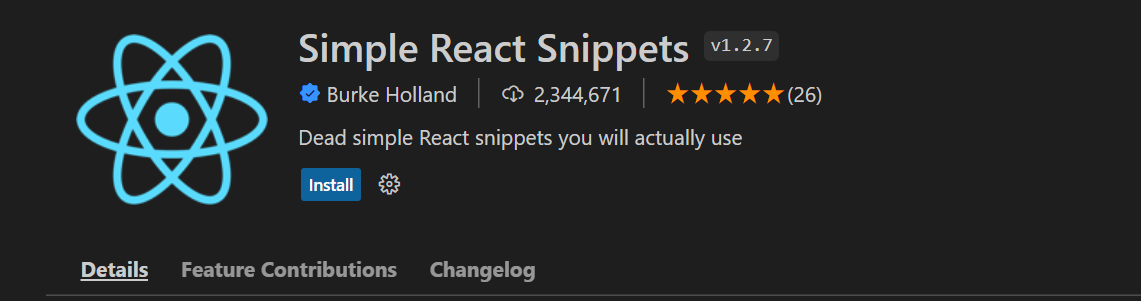
**Para no estar escribiendo constantemente la importación de React:**

**Import React from ‘react’**

**Descargar siguientes extensiones:**

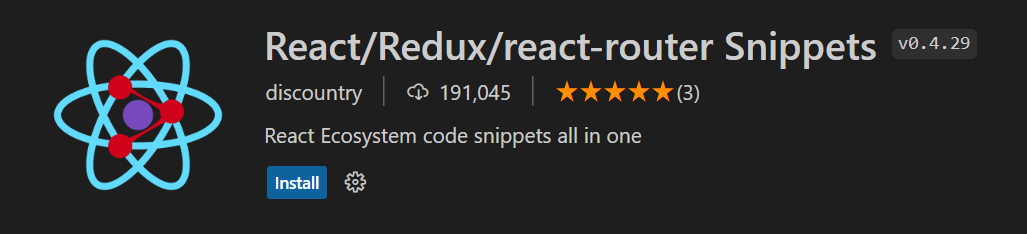
****

****

****

****

**Imr : *import React from ‘react’* sfc*: const = ()=>***

****

**Para no tener div que dentro de inspeccionar se ve como código html de más, pero para poder retornar se necesita un contenedor se puede usar <fragment></fragment>**

import React, {Fragment} from 'react';

  return (

    <Fragment>

      <h1>Básicos React</h1>

      <h2>{saludo}</h2>

      <h2>{mensaje}</h2>

      <Header/>

      <Footer/>

    </Fragment>

  );

**La diferencia entre una FUNCTION DECLARATION Y FUNCTION EXPRESSION es que está última se puede borrar el return porque react lo toma como implícito, en cambio la sintaxis de la funtion expresión no lo permite.**

**Esta es una Function declaration que no se puede obviar el RETURN.**

// se crea componente con function declaration

function Header() {

   return(

       <h3 id="cabezera" className="encabezado">Hola desde elHeader</h3>

    )

}

export default Header;

**Esta es una FUNCTION EXPRESSION.**

// se crea componente con function expresión

const Footer = () => {

    return (

        <footer>

          <p> Todos los derechos reservados &copy;</p>

        </footer>

    );

}

**De esto, podemos pasar a acortar código y no nos va a tirar error, ya que le return queda implícito en la FUNCTION EXPRESSION.**

import React from 'react'

// se crea componente con function expression

const Footer = () => (

    <footer>

        <p> Todos los derechos reservados &copy;</p>

    </footer>

    );

export default Footer

**Props: Los datos en React siempre fluyen desde el componente principal hacía los componentes internos.**

**Si queremos pasar al datos como booleans, strings, números o funciones se utilizan las PROPS.**

**Así se pasa un props desde el componente principal: ej “Tienda virtual” O un número.**

<Header titulo="Tienda virtual" numero={1} />

**En el componente Header llamaría a las props así:**

import React from "react";

// se crea componente con function declaration

function Header(props) {

    return(

        <h3 className="encabezado">Esta es la {props.titulo}

         Número:{props.numero}</h3>

    )

}

export default Header;

**De igual forma la manera adecuada es no utilizando la nomenclatura props sino desestructurar (destructuring) de la siguiente manera:**

function Header({titulo,numero}) {

    return(

        <h3 className="encabezado">Esta es la {titulo}

         Número:{numero}</h3>

    )

}

**Se puede pasar como props código js**

function App() {

const fecha = new Date().getFullYear();

  return (

    <Fragment>

      <h1>Básicos React</h1>

      <h2>{saludo}</h2>

      <h2>{mensaje}</h2>

      <Header titulo="Tienda virtual" numero={1} />

      <Footer fecha={fecha}/>

    </Fragment>

  );

}

export default App;

**En el componente Footer quedaría:**

import React from 'react'

const Footer = ({fecha}) => (

    <footer>

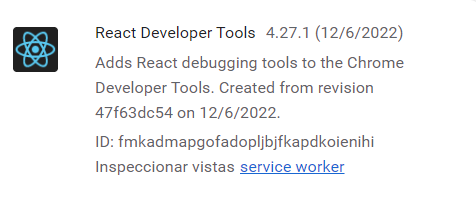
        <p> Todos los derechos reservados &copy; {fecha} </p>

    </footer>

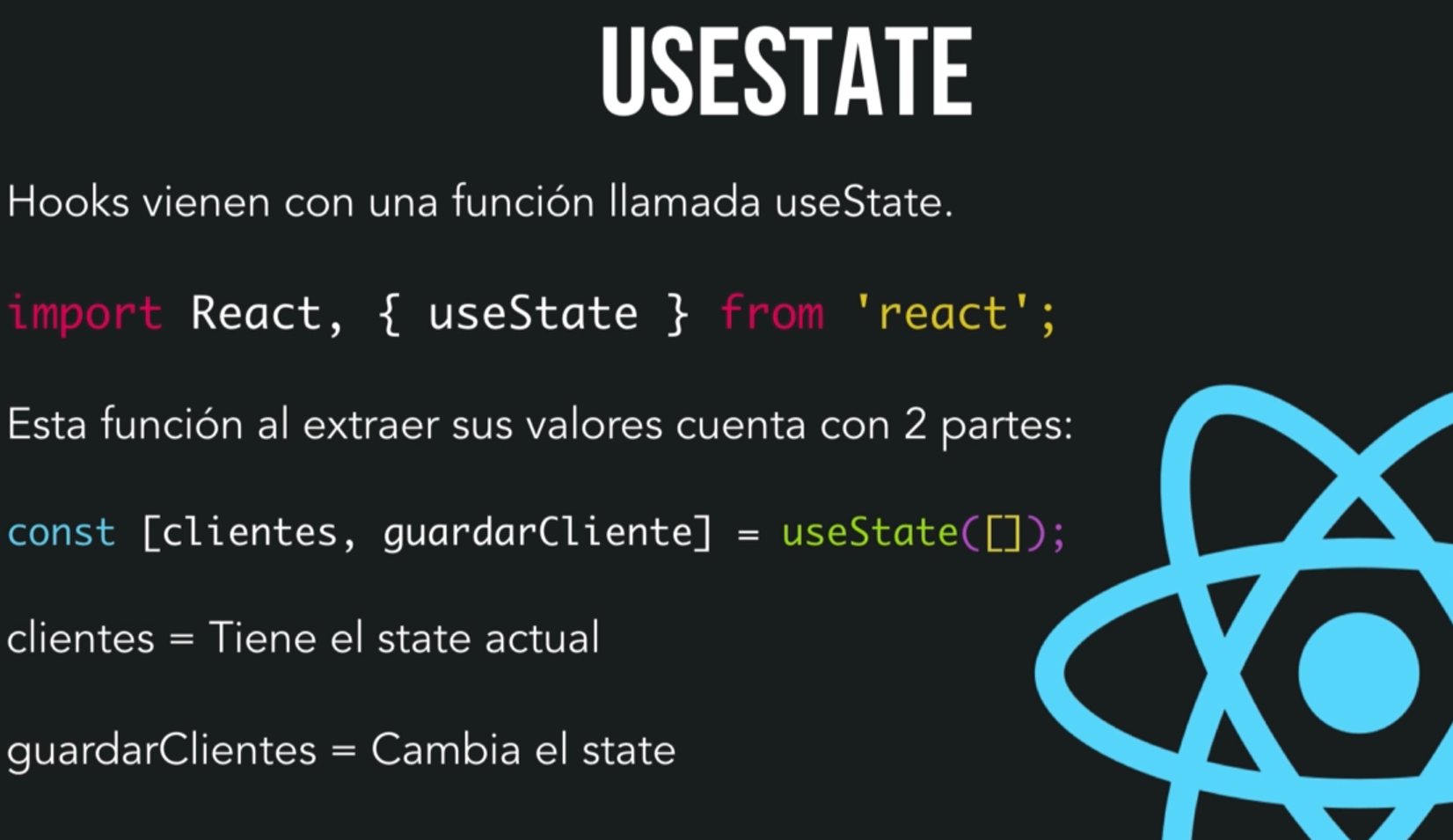
    );

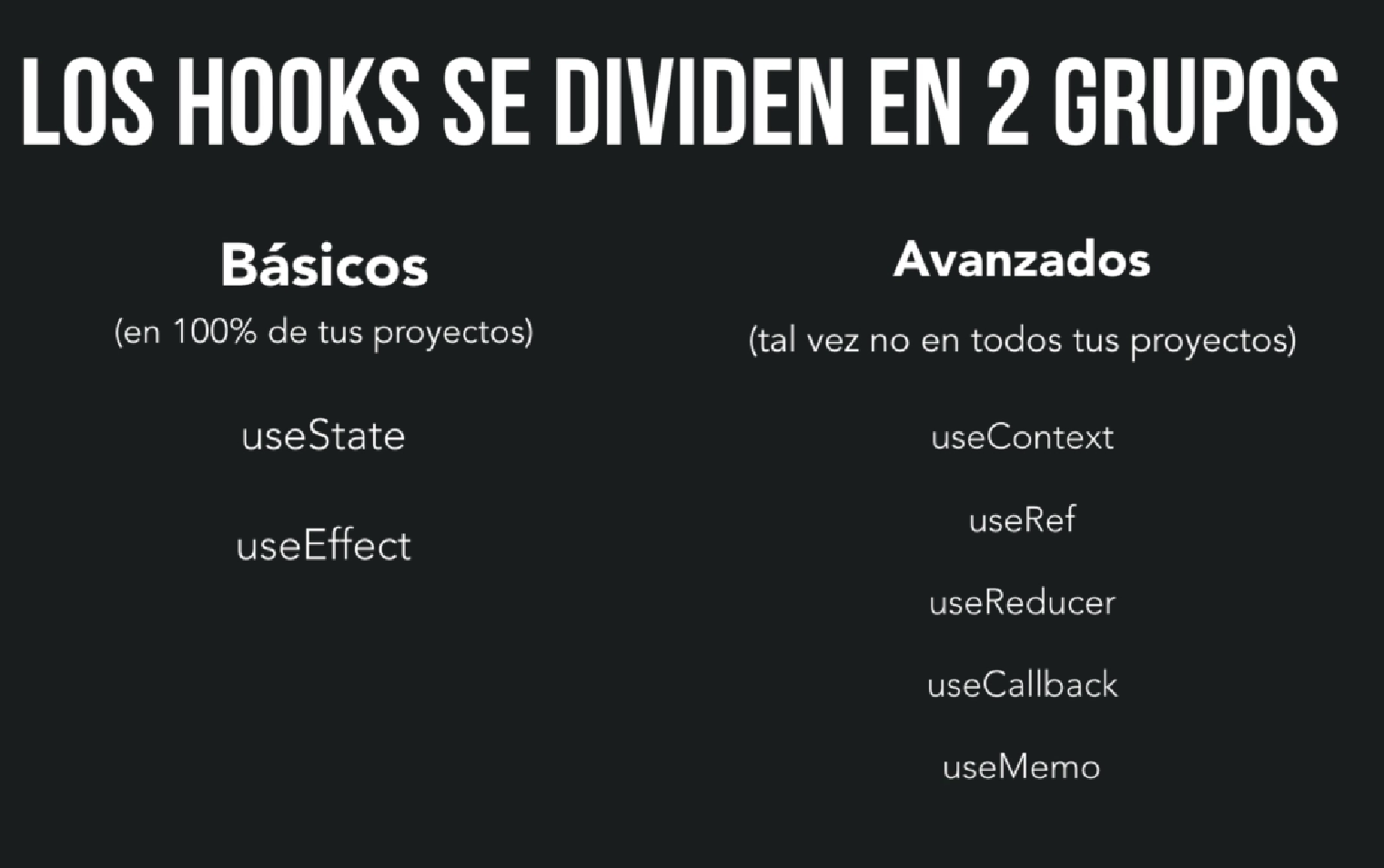
export default Footer

**Siempre tener en Chrome o Firefox la siguiente extensión para hacer visibles las props al inspeccionar los componentes.**

****

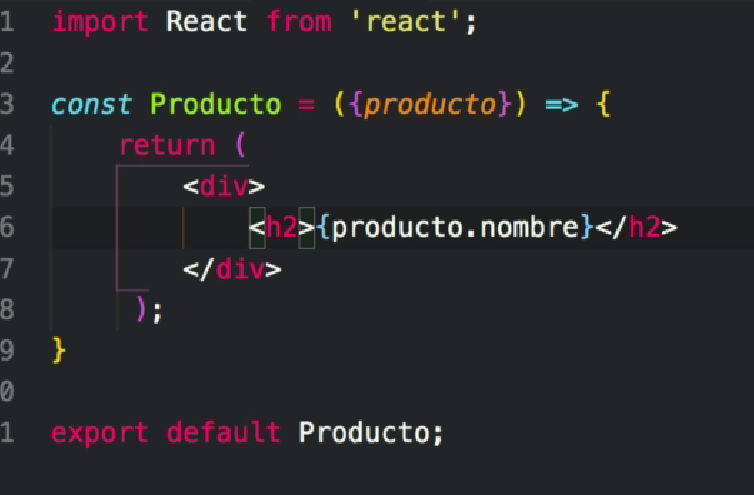
**Introducción a React HOOKS.**

****

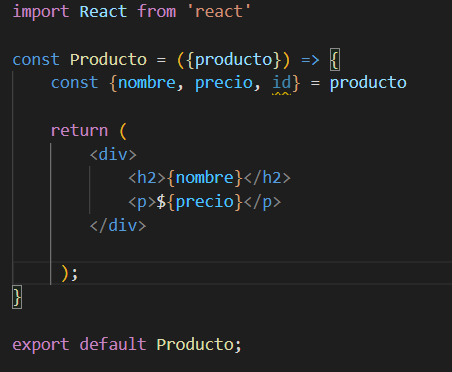
****

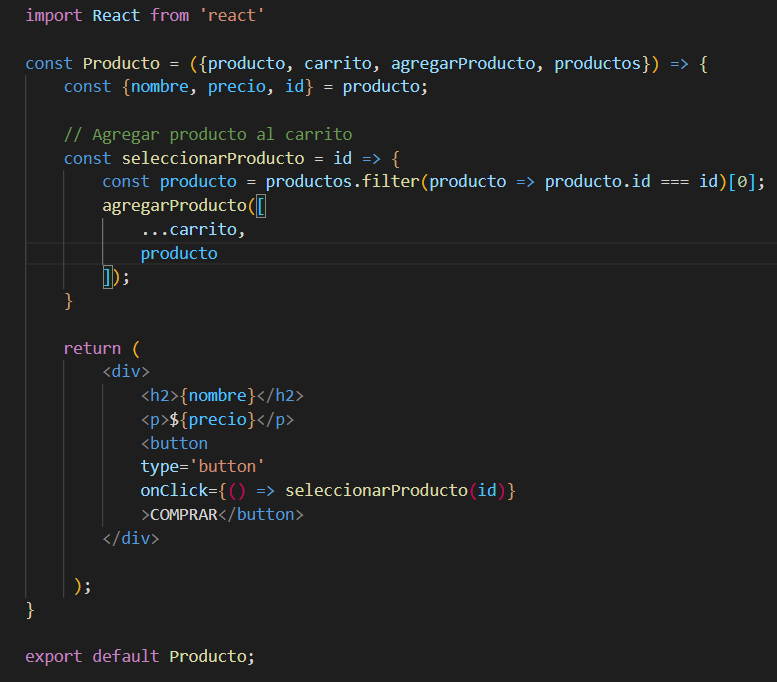
**En vez de pasar las props y llamarlas de la siguiente manera**

**producto.nombre**

****

**Se debe extraer los valores para llamarlos directamente**

****



**Spread Operator (…)**

**Permite clonar un objeto o un arreglo.**

**…carrito**

**En el caso de [0] que se encuentra al final**

**De productos.filter. Este permite que en**

**Consola al hacer click y nos muestre el**

**Producto no desde un nuevo array,**

**Sino desde el propio objeto.**

**Al agregar el segundo valor productos, luego de la copia de …carrito, este permite que se muestre cada producto clickeado. Es decir, se guarda la copia de carrito y se muestra los productos clickeados todos.**

**Viéndolo más de cerca:**

    // Agregar producto al carrito

    const seleccionarProducto = id => {

        const producto = productos.filter(producto => producto.id === id)[0];

        agregarProducto([

            ...carrito,

            producto

        ]);

    }

AgregarProductos, que se encuentra en el componente principal y es la función de useState, cambia el estado es decir el state. La copia del carrito permite no perder esa información cada vez que se actualiza, y el segundo parámetro, que es el producto permite que se muestre el producto agregado, de lo contario no agregaría nada.

Nota: En los Return no se puede usar un IF pero si un TERNARIO