**🏠 Renderizado dinámico**

**👋 Introducción**

**¡Hola!👋 Bienvenido a un nuevo paso del curso de React I.** En esta sección vamos a enfocarnos  en el concepto de renderizado dinámico en React, una técnica esencial para construir interfaces de usuario interactivas y reactivas.

A lo largo de esta sección, exploraremos cómo el concepto de componentización y el hábil manejo de eventos se convierten en las piedras angulares para crear experiencias de usuario dinámicas y personalizadas en aplicaciones React. Descubriremos cómo los componentes modulares pueden adaptarse y actualizarse de manera eficiente, permitiéndonos construir interfaces que respondan de manera ágil a las acciones del usuario.

**📖 Material Descargable**

Aquí encontrarás el material descargable correspondiente que te dará las bases para tus próximos pasos.

👉[**Teoría Manejo de Componentes**](https://drive.google.com/file/d/190GyPxWHWtZ0JRGepZYvxe1_SWzq6fRy/view?usp=drive_link)

# 👣 Renderización dinámica de productos

En esta sección, definiremos enlaces dinámicos para vincular correctamente las tarjetas de cada producto con sus respectivas vistas de detalle. Durante este proceso, también repasaremos conceptos previamente abordados.

### ✏️ Actividad: Crear un componente de la tarjeta de producto

En esta actividad, crearás un componente ProductCard que muestra información de un producto. Este ejercicio te ayudará a entender cómo pasar datos a los componentes y cómo se renderizan en la interfaz de usuario.

1. **Crear el componente ProductCard**:
   * Crear el archivo ProductCard.tsx en la carpeta /src/components
   * Definir la estructura básica de cualquier componente.
   * Cortar el elemento <article> del componente Home.
   * Pegar el contenido dentro del retorno return de ProductCard.
2. **Definir los estilos de ProductCard**:
   * Crear el archivo ProductCard.module.css en la carpeta /src/components
   * Mover los estilos relevantes desde /Home.css a /src/components/ProductCard.module.css.
   * Importar los estilos en el componente ProductCard.tsx y modificar las clases correspondientes.

/\* /src/components/ProductCard.tsx \*/

import styles from "./ProductCard.module.css"

function ProductCard() {

   return <article> ... </article>

}

export default ProductCard;

1. **Implementar el componente ProductCard en la vista**:
   * Importar ProductCard en Home.
   * Llamar al componente <ProductCard /> donde estaba el <article>.

/\* /src/views/Index.tsx \*/

import styles from "./Home.module.css";

import NavBar from "../components/NavBar";

import ProductCard from "../components/NavBar";

function Home() {

  return <>

      ...

      <ProductCard />

      <ProductCard />

      <ProductCard />

      <ProductCard />

      <ProductCard />

      ...

  </>;

}

export default Home;

1. **Verificar el correcto renderizado de la vista.**

### ✏️ Actividad:  Reutilizar Componentes

Podríamos reutilizar el componente ProductCard tal y como lo hicimos con NavButton (uno debajo del otro), pero esta práctica es poco práctica y manual. Además, para este caso en particular, la cantidad de productos a renderizar es variable por lo que debemos implementar otra forma.

1. **Identificar los datos**:
   * Buscar y copiar la variable products del curso JavaScript 1. Cada objeto del array products representa un producto con sus propiedades.
   * Pegar la variable products en el archivo /src/views/Home
2. **Analizar los datos dinámicos de la tarjeta**:
   * En este caso, los datos que cambian son: id, title, color price e image. Estos datos se pasarán al componente como atributos de la etiqueta.
3. **Programar los atributos de una tarjeta**.
   * Para generar n tarjetas de producto es necesario implementar el método para iterar map(). Cada elemento de la iteración necesita la configuración del atributo key, necesario para que React identifique de manera eficiente cada elemento y realice actualizaciones de manera más efectiva cuando la lista cambia.
   * El uso de map en lugar de foreach se debe a que en una aplicación de React, es necesario que cada iteración retorne lo que se va a renderizar en la vista:

/\* /src/views/Index.tsx \*/

const products = /\* pegar array \*/

function Index() {

  ...

  return (

    ...

    <div className={...}>

      {products.map((each) => (

        <ProductCard

          id={each.id}

          title={each.title}

          price={each.price}

          color={each.colors[0]}

          image={each.images[0]}

        />

      ))}

    </div>

    ...

  )

}

export default Index;

1. **Interpolar las propiedades en la tarjeta**:
   * Con esta configuración de atributos, el componente recibe un objeto con estas propiedades, por lo tanto, es necesario desestructurarlo y luego, cambiar los datos estáticos haciendo uso de la interpolación de variables con {}. A modo de ejemplo:

/\* /src/components/ProductCard.tsx \*/

import styles from "./ProductCard.module.css"

function ProductCard(props) {

  const { id, title, color, price, image } = props

  return (

    <article className={...}>

      <img className={...} src={ image } alt={ id } />

      <div className={...}>

        <span className={...}>{ title }</span>

        ...

    </article>

  )

}

export default ProductCard;

1. **Generar un enlace dinámico**:
   * En el archivo /src/components/ProductCard importar el componente Link del módulo de RRD.
   * Cambiar la etiqueta article por el componente Link.
   * Configurar el atributo to concatenando la ruta de detalle con el id de las props

/\* /src/components/ProductCard.tsx \*/

import styles from "./ProductCard.module.css"

import { Link } from "react-router-dom";

function ProductCard(props) {

  const { id, title, color, price, image } = props

  return (

    <Link to={"/product/"+id} className={...}>

        ...

    </Link>

  )

}

export default ProductCard;

1. **Verificar el correcto renderizado de la vista y los links.**

# 👣 Creación de vistas de detalle de producto

En este paso, aprenderás a crear una vista dinámica de React para mostrar el detalle de productos utilizando useParams de react-router-dom.

Este componente buscará un producto específico basado en un ID de la URL y lo mostrará. Si el producto no se encuentra, mostrará un mensaje indicando que el producto no se encontró.

### ✏️ Actividad:  Crear vista dinámica

1. **Identificar los datos**:
   * Buscar y copiar la variable products del curso JavaScript 1. Cada objeto del array products representa un producto con sus propiedades.
   * Pegar la variable products en el archivo /src/views/Product
2. **Implementar el hook useParams**:
   * Importar useParams de react-router-dom\*\*. Esta función te permitirá acceder a los parámetros dinámicos de la URL.
   * El hook devuelve un objeto con todos los parámetros definidos en la configuración del browserRouter. En este caso definimos sólo el parámetro id.
3. **Buscar el producto correspondiente**:
   * Utiliza el método find en el array products para encontrar el producto que coincide con el id obtenido.
4. **Renderizar el producto o un mensaje de error**:
   * Si el producto existe, la vista renderiza los detalles del producto encontrado. Analizar la vista y en cada dato estático del producto utilizar la interpolación con {} para renderizar correctamente cada propiedad del objeto encontrado.
   * Las miniaturas y los colores requieren un mapeo ya que la cantidad de elementos del array es variable de un producto a otro.
   * En caso contrario, muestra un mensaje indicando que el producto no se encontró.

/\* /src/views/Product.tsx \*/

import styles from "./Product.module.css";

import { useParams } from "react-router-dom";

const products = /\* pegar array \*/

function Product() {

   const { id } = useParams();

   const product = products.find((each) => each.id === id);

   if (!product) {

      return <>

         ...

         <span className={...}>{product.title</span>

         ...

         <select type="text" placeholder="Selecciona un color">

            {product.colors.map(each=><option value={each}>{each}</option>)}

         </select>

         ...

      </>

   }

   return <> ... </>

}

export default Product;

1. **Verificar el correcto renderizado de la vista y los links.**
   * Acceder a las tarjetas de Home para verificar que se renderiza la vista Product con los detalles del producto.
   * Verificar que las miniaturas y los colores se renderizan correctamente.
   * Acceder a /product/id222 para verificar que se renderiza el mensaje de error.