**🏠 Hook useState**

**👋 Introducción**

En esta etapa, nos sumergimos en el uso del hook useState(), una herramienta fundamental en React para la gestión de estados en componentes funcionales. Exploraremos cómo useState() posibilita la creación de componentes dinámicos e interactivos, permitiéndoles mantener su propio estado a lo largo del tiempo. Descubrirás técnicas avanzadas y mejores prácticas para manejar el estado de manera eficiente, aspecto crucial en el desarrollo de aplicaciones modernas y reactivas.

El hook useState es una de las características más esenciales en React para gestionar estados en componentes funcionales. Este hook permite que los componentes recuerden información y reaccionen a los cambios en ella, lo cual es vital para la creación de interfaces dinámicas. Es una herramienta poderosa y flexible en React, permitiéndote agregar estados reactivos a tus componentes funcionales, convirtiéndolos en elementos interactivos capaces de responder a la entrada del usuario y a otros cambios a lo largo del tiempo.

**📖 Material Descargable**

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

👉[**Teoría Manejo de Estados II**](https://drive.google.com/file/d/1GCU4jLQGBbTk605YHWBm-oe0hxaPA-87/view?usp=drive_link)

# 👣 Manejar estados para cambiar la miniatura

Este paso se enfoca en la dinámica interacción con las miniaturas de los productos en una aplicación React. Mediante la gestión de estados, aprenderás a cambiar la imagen principal de un producto basándote en la miniatura seleccionada por el usuario. Esta funcionalidad enriquece la experiencia del usuario, permitiendo una visualización detallada y flexible de los productos.

## 📖 Material Audiovisual

Si lo deseas, te invitamos a mirar este video que amplía el concepto sobre el uso de hook useState.

<https://www.youtube.com/watch?v=TbegfYdyybA>

**❗*No es necesario que crees el ejemplo mostrado en el video, lo importante es que comprendas de manera clara lo aprendido para que  puedas aplicar lo visto en las actividades posteriores.***

### ✏️ Actividad:  Cambiar el estado de la foto ampliada del producto

En esta actividad, utilizarás el hook useState para controlar la visualización de la imagen principal de un producto basada en la selección de miniaturas.

1. **Definir un estado con un booleano**:
   * Importa el hook useState en el componente Thumbs.
   * Desestructura el hook con el estado y su función de actualización.
   * Define el estado inicial de thumb como la primera imagen del array images del producto, o una imagen predeterminada en caso de que no haya imágenes disponibles.

const [thumb, setThumb] = useState(product?.images[0] || /mock1.jpg");

1. **Configurar la imagen ampliada**:
   * Modifica el atributo src de la imagen principal para que muestre la imagen almacenada en el estado thumb.

<img id="selected-thumbnail" src={thumb} alt="" />

* Configura el evento para "setear" el estado de la foto. Cada click debe configurar el correspondiente cambio de estado de thumb para luego poder modificar la imagen ampliada del producto.

{product?.images.map((each, index) => (

   <img

     className={styles.thumbnail}

     key={index}

     src={each}

     alt={each}

     onClick={() => setThumb(each)}

   />

))}

1. **Verificar el correcto renderizado y cambio de estados.**

**👣 Manejar estados para cambiar sugerencias**

En este paso, te embarcarás en el desarrollo del componente Suggestions. Este componente será una parte vital de la interfaz de usuario ya que mostrará una lista de productos relacionados, ofreciendo a los usuarios opciones adicionales de compra.

**✏️ Actividad:  Crear un componente para las sugerencias**

En esta actividad, crearás el componente Suggestions, que mostrará una lista de productos relacionados. Este componente te permitirá practicar el manejo de estados y el renderizado de listas en React.

1. **Crear el componente Suggestions**: Crear los archivos Suggestions.module.css y Suggestions.tsx en la carpeta /src/components y definir la estructura básica de cualquier componente.
2. **Extraer y trasladar las sugerencias al componente ProductCard**:
   * Cortar el bloque de cógido HTML correspondiente a las sugerencias de la vista Product.
   * Pegar el contenido dentro del retorno return de Suggestions.
3. **Ajustar estilos**:
   * Mover los estilos relevantes desde /views/Product.module.css a /src/components/Suggestions.module.css.
   * Importar los estilos en el componente Suggestions.tsx y modificar las clases correspondientes.
4. **Identificar los datos**:
   * Buscar y copiar la variable products del curso JavaScript 1.
   * Pegar los datos en el componente.
5. **Reutilizar ProductCard**:
   * Importar ProductCard en Suggestions.
   * Implementar el método slice para "cortar" el array products de forma que sólo tenga cuatro elementos.
   * Mapear para generar las tarjetas de producto.
   * Agregar las flechas al bloque de código.

<div className="sales-box">

  <img src="/chevron-left-solid-60.png" />

  <div className={...}>

    {products.slice(from, to).map((each) => (

      <ProductCard

        key={each.id}

        id={each.id}

        image={each.images[0]}

        title={each.title}

        color={each.colors[0]}

        price={each.price}

      />

    ))}

  </div>

  <img src="/chevron-right-solid-60.png" />

</div>

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

**✏️ Actividad:  Cambiar el estado de las tarjetas de productos sugeridas**

1. **Definir los estados para el número "desde" y "hasta" dónde**:
   * Importa el hook useState en el componente Suggestions.
   * Utiliza el hook useState para manejar el rango de productos mostrados (from y to).
   * Desestructura el hook con el estado y su función de actualización.
   * Define el estado inicial de from y to para que muestre los primeros elementos del array de productos.

const [from, setFrom] = useState(0);

const [to, setTo] = useState(4);

1. **Configurar los límites con estados**:
   * Modifica los datos del slice por los estados definidos.

{products.slice(from, to).map(...)}

* Implementa las funciones nextSug y prevSug para actualizar los estados from y to, permitiendo la navegación entre diferentes "páginas" de productos.

const nextSug = ()=> {

   setFrom(from+4)

   setTo(to+4)

}

const prevSug = ()=> {

   setFrom(from-4)

   setTo(to-4)

}

* Configura los eventos para "setear" el estado de las páginas. Cada click debe configurar el correspondiente cambio de estado de from y to para cambiar los productos de las sugerencias.

<div className="sales-box">

  <img src="/chevron-left-solid-60.png" onClick={prevSug} />

  <div className={...}>

      ...

  </div>

  <img src="/chevron-right-solid-60.png" onClick={nextSug} />

</div>

1. **Verificar el correcto renderizado y cambio de estados.**