**investigacion**

**Map.**

es una herramienta poderosa y flexible para transformar arreglos de forma sencilla y limpia.

**Every**

es perfecto para situaciones donde necesitas asegurarte de que todos los elementos de un arreglo cumplan con una regla específica.

**Split**

es una herramienta poderosa para trabajar con cadenas en JavaScript, permitiendo dividir texto en partes manejables según tus necesidades.

**Filter**

en JavaScript se utiliza para crear un nuevo arreglo con todos los elementos del arreglo original que cumplan con una condición especificada. Es muy útil para filtrar datos según ciertos criterios.

**Includes**

Como funcionan y como se aplican

se utiliza para determinar si un arreglo o una cadena contiene un elemento o una subcadena específica. Devuelve un valor booleano:

**true**: Si el elemento o subcadena está presente.

**false**: Si no está presente.

Es una forma sencilla y legible de verificar la existencia de un valor.

**Onchange**

El evento onchange en JavaScript se dispara cuando el valor de un elemento en el DOM cambia y se pierde el foco, es decir, cuando el usuario realiza una modificación y luego interactúa fuera del elemento (por ejemplo, al hacer clic en otra parte de la página o presionar "Enter").

**Aplicación común:**

El evento onchange se usa con frecuencia en:

Campos de entrada (<input>).

Listas desplegables (<select>).

Áreas de texto (<textarea>).

**Onclick**

El método **onclick en HTML** es sencillo y útil para ejemplos rápidos.

**JavaScript Vinculado** es más limpio y separado del HTML.

**addEventListener** es la mejor práctica, ya que permite agregar múltiples eventos al mismo elemento y facilita el mantenimiento del código.

**Onblur**

El evento onblur es ideal para validar formularios y proporcionar retroalimentación inmediata al usuario cuando deja un campo. Se recomienda usar addEventListener para mantener el código limpio y manejable.

Oncreate

El evento oncreate como tal no existe en JavaScript puro o HTML. Su implementación depende del framework o biblioteca que estés utilizando. En cualquier caso, se relaciona con ejecutar lógica en el momento de creación o inicialización de un elemento o componente. Adapta la solución a las herramientas que estés utilizando.

Archivo tareas

Va a tener un objeto que tiene varias tareas id texto un pendiente estado

Vamos a utilizar el método map

Hacer una especie de foreach

Siempre crea una copia por decirlo asi de nuestro arreglo como tal

Ideal para componentes repetitivos

Tenemos mi app.js limpio aun no tiene fragmentos

Creamos archivo tareas.js

Importar React

Crear una función tareas

Dentro de esta esta arreglo de objetos se va a llamar tareas va hacer un arreglos que tiene id, Nombre de la tarea un estado

Comprar frutas cuyo estado es false

Segundo objeto es el id 2 estudiar React false

Id 3 se va a llamar almorzar cuyo estado va hacer true

Vamos hacer un return dentro de este return vamos a colocar lo que lleva tareas

Dentro de un div colocamos un h2 lista de tareas

Vamos hacer una lista ul vamos a generar el map tareas.map

Hago referencia tarea

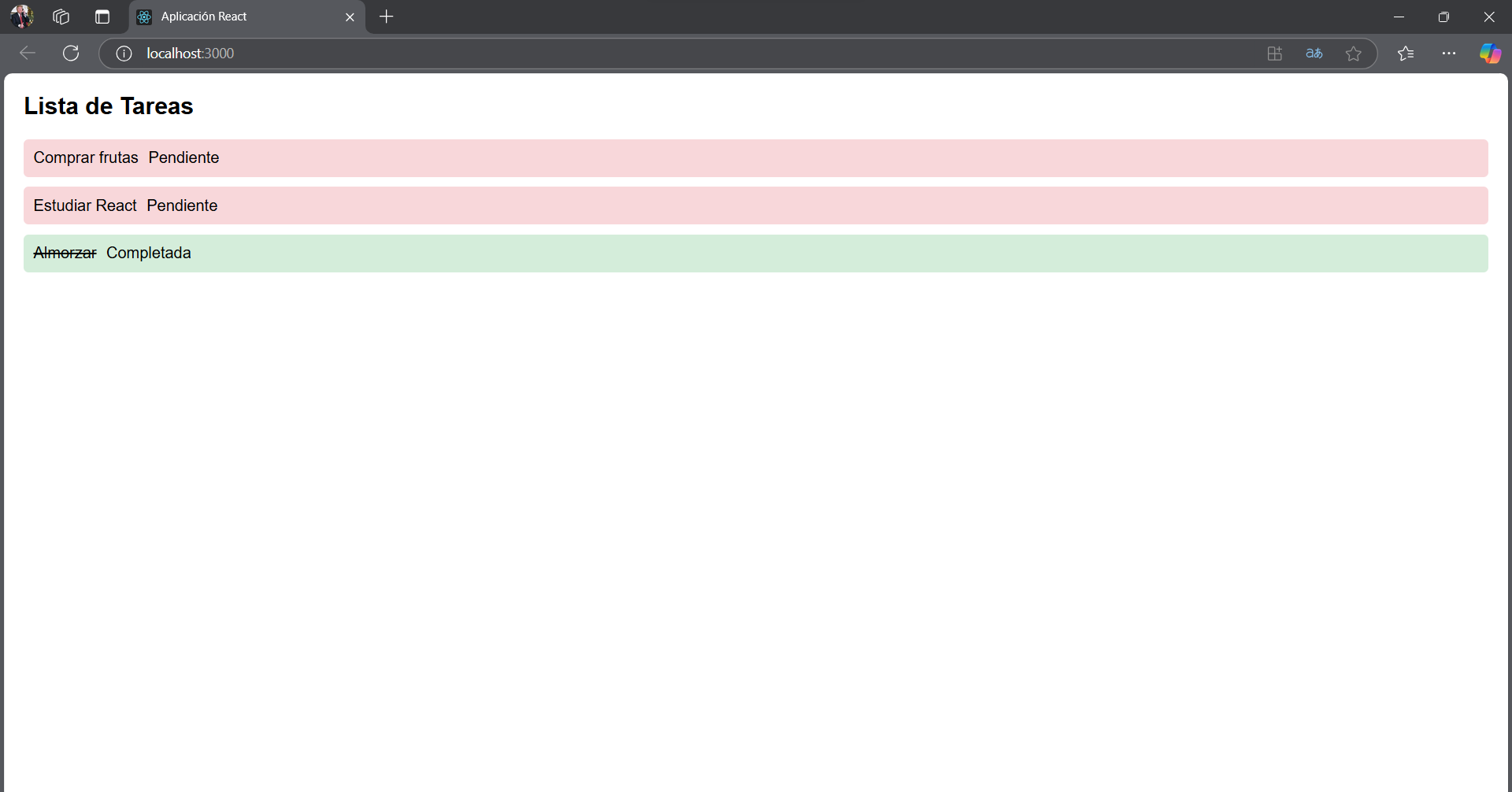
El ul lleva una lista por medio de li

Porque me va a repetir 3 veces

Export default tareas

Colocar estado – tarea.estado

Identificar cada uno de los li



Recorrer una lista mediante el dom

El ultimo punto de la actualizacion

Utilización de la api mostrar cualquier api que uno desee