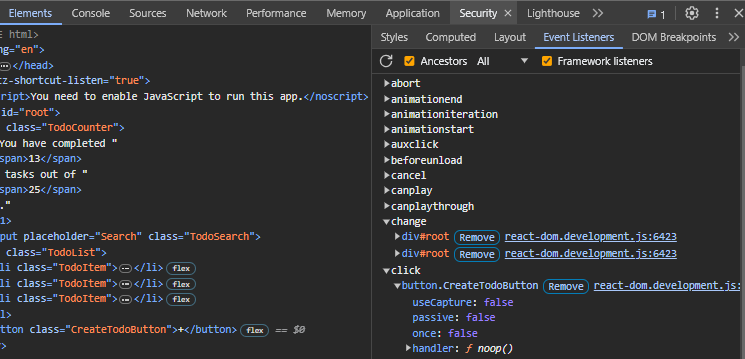


react coge todo lo que comience con On y lo va a transformar en un AddEventlistener, en un escuchador de evento de java script

Se usa el camel case para llamar el evento de java script, y toca encapsular lo de adentro en un fuction

Estos event listener se puede ver en inspeccionador hmtl 

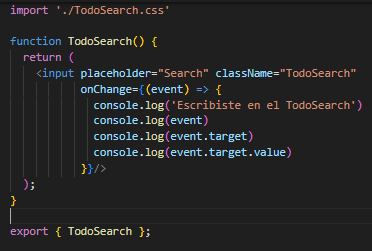
A computer screen shot of a program code

Description automatically generated

uno le puede pasar al fuction el event el cual tiene varios atributos, pero el que más se usa el target ya que este permite ingresar al elemento html

A screen shot of a computer

Description automatically generated

 este target como es el elemento como tal. Se puede acceder al Value

**ESTADOS**

Puede los estados que uno quiera, los estados no solo se consumen también se actualizan

El estado en react es imutable por eso se maneja como array

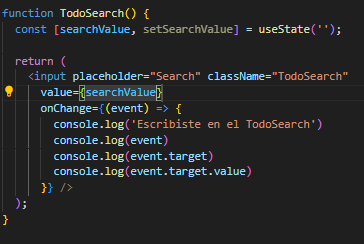


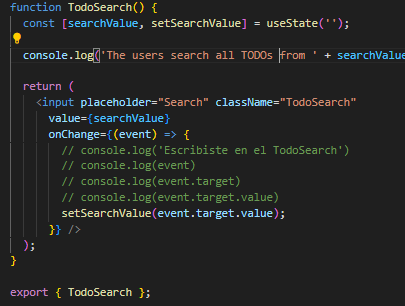
Se necesita una fucion actualizadora del estado controlada por REact Js para que pueda hacer todos sus procesos de inmutabilidad y no afecte al historial de estados de componente

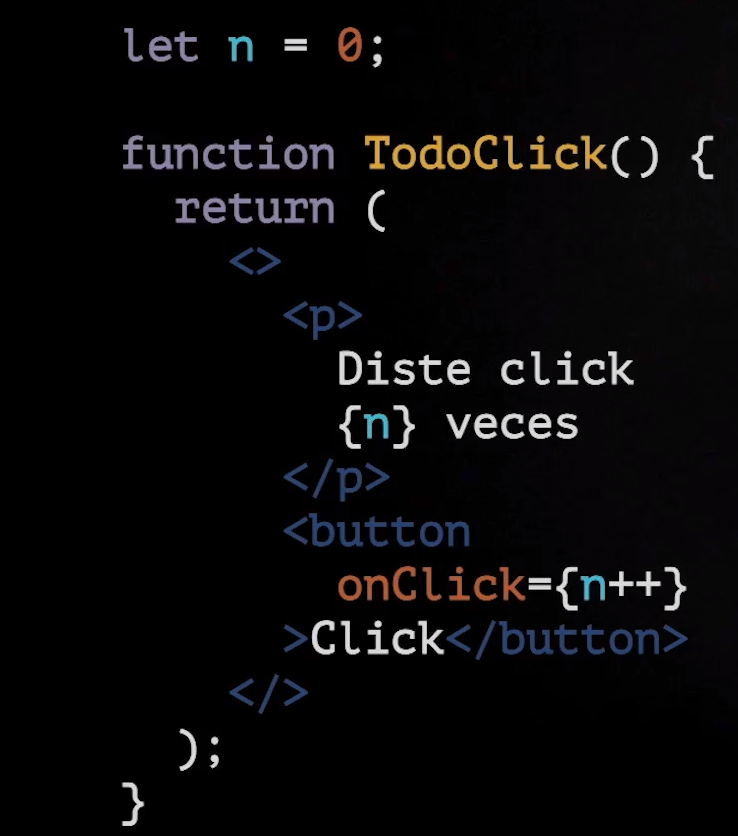
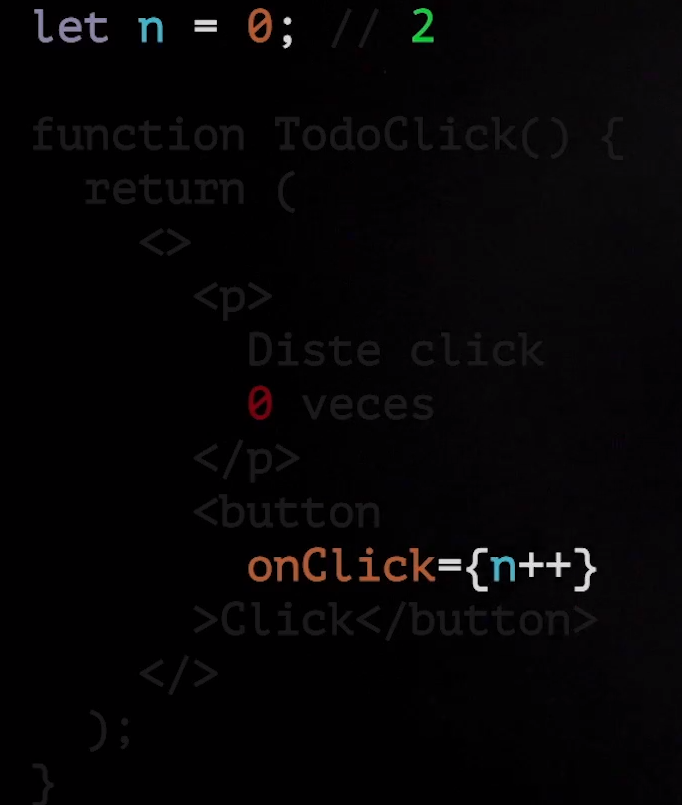


El estado se permite inicializar,

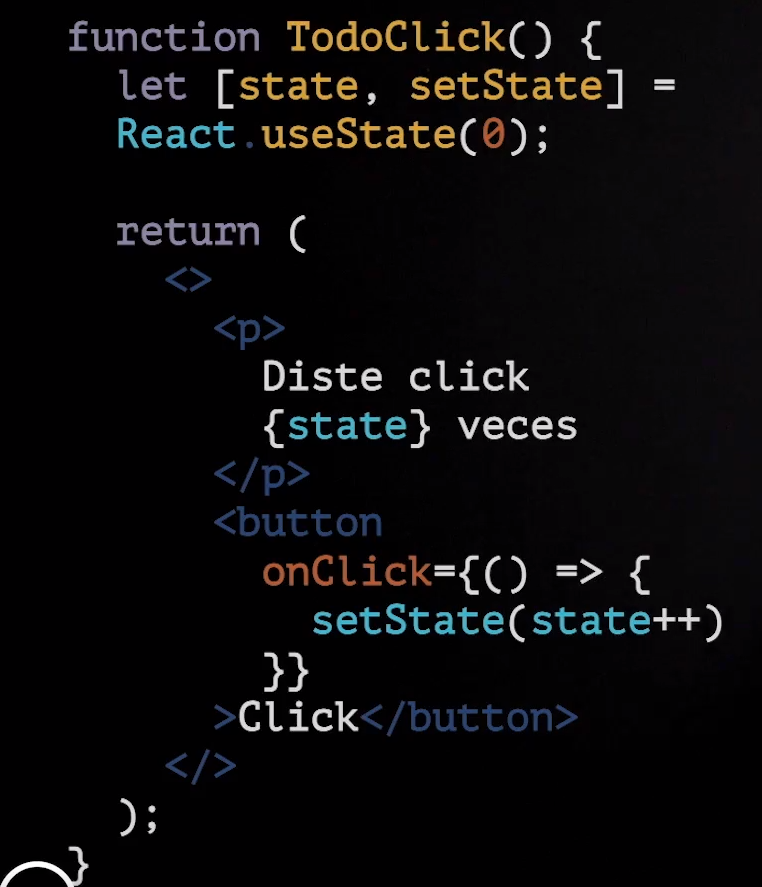
Se conecta el value del input con el estado creado

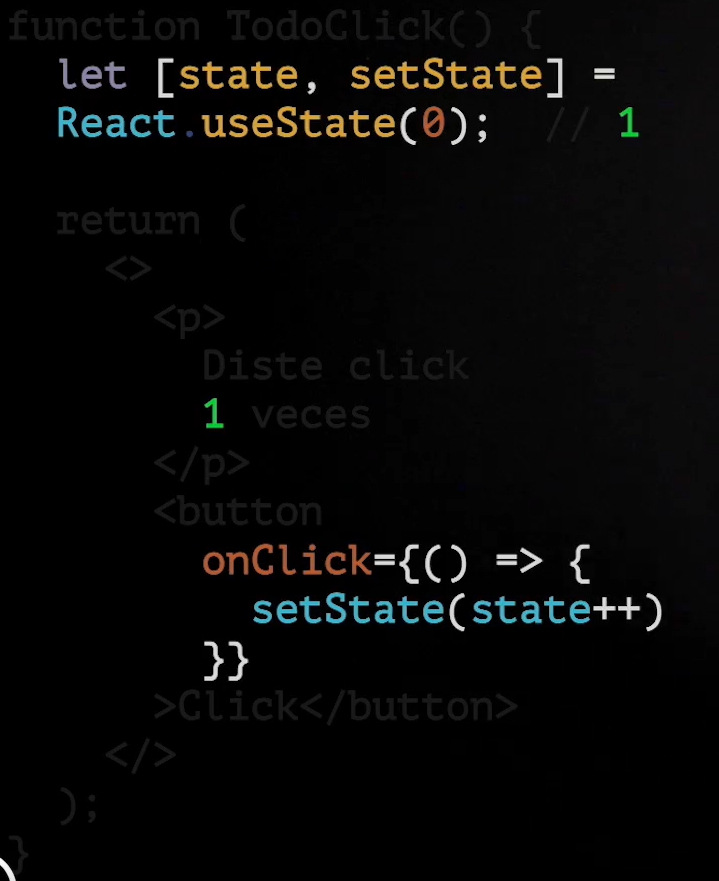


 ya que el value esta atado al estado toca usar el actualizador para que el value se actulice de forma correcta, así mismo se puede acceder al value del estado directamente con la variable creada

En este ejemplo cuando el usuario le di click la variable se actualiza mas no el texto en el html, esto debido a que React no se entera que tiene que renderizar de nuevo ese elemento

 Para esto se usaría estados

 Cada vez que se cambia el estado de un componente, está llamada a la fábrica de render y le pide que rendirize el componente de nuevo. Esto es con eventos lo cuales se encarguen de cambiar el estado para que se rendirize de nuevo

React maneja un virtual DOM, que es la copia del DOM principal, cada vez que renderiza algo de la apliacion, este aplica los cambios al virtual DOM y si algo que cambie hay si hace el cambio en el DOM principal. Esto le permite a REACT renderizar solo un elemento especifico que cambio y no todos

Solo se pueden pasar estado de padres a hijos y no de hijos a padres

Para esto toca combinar los props con lo estados y el estado tiene que se parte de padre no del hijo, para que este pueda pasarle es actualizador y el estado a los hijso

Estados derivados, estadoa que apartir de un estado hagan alguna operación y así poderlo pasar a otro componentes

A computer screen shot of a program code

Description automatically generated

La diferencia entre los métodos toLowerCase() y toLocaleLowerCase() en JavaScript está relacionada con la forma en que se procesan los caracteres en mayúsculas y acentuados en diferentes idiomas.

toLowerCase() convierte una cadena de texto en minúsculas, utilizando las reglas de conversión que se aplican a los caracteres ASCII (caracteres en inglés y otros idiomas europeos que no tienen acentos)

Por otro lado, toLocaleLowerCase() también convierte una cadena de texto en minúsculas, pero utiliza las reglas de conversión específicas del idioma y la ubicación (localización) en la que se está ejecutando el código. Esto significa que, en función de la localización, algunos caracteres con acentos o diacríticos (como la letra "á" en español) pueden ser convertidos a su equivalente en minúsculas, mientras que otros caracteres pueden permanecer sin cambios.

A computer code with colorful text

Description automatically generated with medium confidence

Fuctionception ember una fuction dentro de otra

Los eventos en react no esperan la fuction ya ejecutada si no una fuction a la que reat pueda ponerle los paréntesis cuando ocurra el evento o la interacción

A screen shot of a computer code

Description automatically generated

A diferencia de A screen shot of a computer code

Description automatically generated

Apuntes

Compañeros fijense que cuando usamos el useState(), ese arreglo que nos devuelve es una const. Y algunos diran: "pero si el estado va a estar cambiando por qué lo guardamos como una constante y no usamos "let"?

La respuesta es que en React el estado está diseñado para ser inmutable, lo que significa que no debemos modificar directamente el valor del estado. En su lugar, React proporciona una función específica para actualizar el estado y generar un nuevo valor.

En React se vuelve a renderizar un componente y sus hijos en 2 situaciones: cuando se cambia el estado (con la función set...) y cuando sus props cambian. . El renderizado en React vuelve a "ejecutar" TODA la función: hooks, variables, funciones y sus elementos. PEEEERO, el useState no se vuelve a ejecutar, es decir, el useState solo se crea una sola vez durante toda la vida útil del componente

<https://es.react.dev/learn/render-and-commit>

<https://es.react.dev/learn/state-a-components-memory#adding-a-state-variable>

<https://es.react.dev/learn/queueing-a-series-of-state-updates>

Qué es el Estado? .. El "Estado" es una de las principales herramientas que React nos proporciona para crear UI´s dinámicas y actualizadas en tiempo real. Podríamos definir al Estado como "Variables que almacenan información dinámica de un componente y se actualizan a medida que el usuario interactúa con los Componentes".

Aquella información utilizada en el estado puede ser utilizada por el componente para cambiar su apariencia y comportamiento en tiempo real. El estado también nos permite mantener la información actualizada de un componente sin necesidad de renderizar todo el componente, esto gracias al Virtual DOM que compara los cambios que se hicieron y los actualiza.