Desarrollo de Aplicaciones Web

Tema Nº11:REACT JS – FUNDAMENTOS

Indicador de logro Nº11:Construye interfaces de usuario por medio de la librería de JavaScript a través de React JS desarrollada por Facebook para optimizar la perfomance de las aplicaciones web.

**TEMA 01 Teoría de los**

Imagen que contiene Icono

Descripción generada automáticamente

**TEMA Nº11:**

REACT JS – FUNDAMENTOS

**Subtema 11.1:**

¿Qué es React?

React surge de una necesidad, la de resolver los problemas en los que se encontraba Facebook para realizar el mantenimiento del código de los anuncios dentro de su plataforma Facebook Ads Org. Con el crecimiento de la compañía, surgieron nuevas necesidades/requerimientos que se transformaron en un código no paraba de aumentar en cantidad y complejidad cada día más y más.

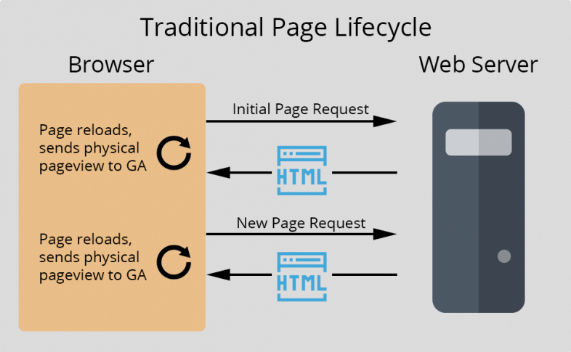
Todo esto se traduce en invertir más dinero para cubrir sus necesidades de más ingenieros para poder realizar el mantenimiento del producto.

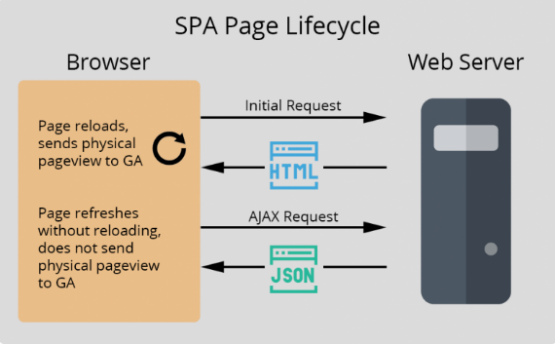


Estos problemas provocaron que se buscará una solución urgente. Este escenario es ideal para la gestación de React y aquí es donde entra en juego Jordan Walke al que se él encarga el desarrollar una solución. Por tanto, el desarrollo de ReactJS se inicia en 2011, en un proyecto de Facebook liderado como no por el ingeniero de software Jordan Walke, el cuál podemos afirmar que es el padre de React.

En los orígenes de la creación React, se inició con la creación de un prototipo que recibió el nombre de FaxJS. React buscaba optimizar y otorgarle cierta rapidez a la carga de contenidos a la red social de Facebook. El objetivo principal de React es organizar los elementos sobre una interfaz dividiendo dicha interfaz de usuario (UI) en una colección/conjunto de componentes.

Los componentes nos permiten dividir una aplicación web en pequeñas partes donde cada parte es independiente y aunque estás partes sean independientes las podemos trabajar juntas o unirlas para realizar una aplicación web. React, además, destaca por ser SPA (Single Page Application). Con SPA, nos referimos a aplicaciones de una sola página.



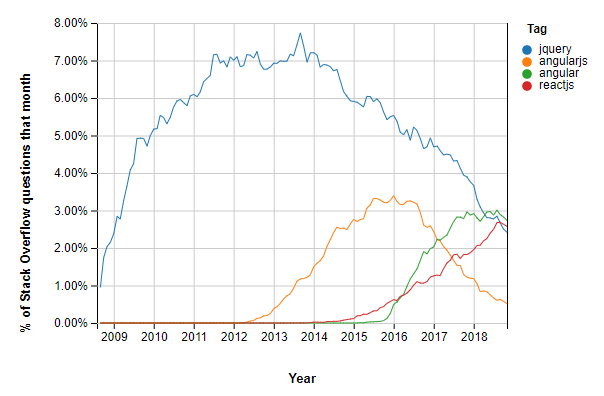


**Subtema 11.2:**

React en la actualidad

Cuando hablamos de React, estamos hablando de una de las librerías más importantes y populares de todo JavaScript. De código abierto licencia MIT (una licencia que se caracteriza por ser bastante blanda debido a que es una licencia libre, de código abierto y sin copyleft, por todo ello, podemos afirmar que nos encontramos ante una licencia bastante blandita).

Muestra de esa importancia y para ayudarnos a aterrizar y coger un poco de contexto sobre la importancia de esta librería en el desarrollo Web/Front actual, vamos a ver una imagen que hemos obtenido de divante.com.



En la gráfica, podemos ver el total de consultas que se lanzan por mes en Stack Overflow sobre jQuery, AngularJS, Angular y ReactJS durante el transcurso de casi la última década desde 2009 hasta 2019.

En este gráfico, podemos ver un cambio de tendencia a favor de dos puntales como son: ReactJS y Angular que literalmente se están comiendo a jQuery con patatas.

**Subtema 11.3:**

Tipos de React

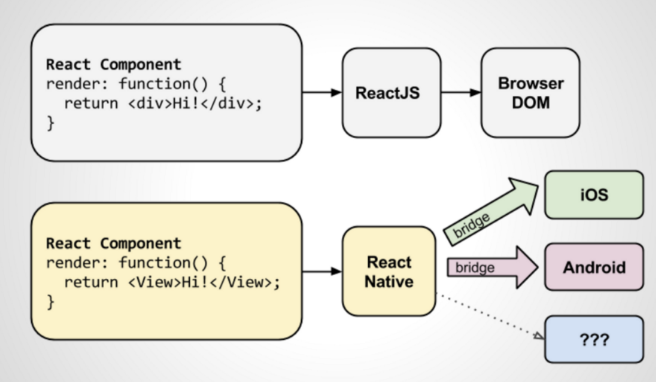
Normalmente cuando hablamos de React, nos referimos a React.js o ReactJS, pero de dentro del mismo troncal (de React) salen tres ramas diferentes: ReactJS, React Native y React VR. ¿Sobre cuál de ella estamos trabajando?

**React JS VS React Native VS React VR**

Tres vertientes que nacen con el fin de cubrir necesidades distintas y que nos permitirán diferenciar dentro del troncal principal de React según las necesidades que cubren. Aunque las tres son hijas del mismo padre, saber situar sobre cuál de ellas estamos trabajando es importante, para ello vamos a explicar un poquito sobre las 3 versiones de React:

* **React JS**: está destinado a la creación de interfaces de usuario para aplicaciones web.
* **React Native**: está destinado a la creación aplicaciones móviles nativas con JavaScript en: Android, iOS, multiplataforma (híbridas).
* **React VR**: está biblioteca de JavaScript, está destinada al desarrollo de aplicaciones web en realidad virtual. Tales como: escenas en 3D (3 dimensiones) como también al desarrollo de interfaces de usuario interactivas mediante al uso de la realidad virtual.

React JS usa HTML en cambio React Native utiliza su propia sintaxis. Hay que matizar que, pese a sus pequeñas peculiaridades, las tres bibliotecas de React son proyectos de código abierto desarrollados por Facebook.



**Compañías que usan React**

Algunas de las grandes compañías que usan React son: Facebook (como no, no podía faltar la empresa creadora de React), Instagram, Netflix, Skype, Airbnb, WhatsApp, Wix, Uber, Dropox… La lista como se podrá ver se podría extender mucho más.

**Subtema 11.4:**

Ventajas de React

* **Veloz y reutilizable**: La carga de páginas con ReactJS son más rápidas debido a que crea interfaces de usuario basándose en componentes como unidad primaria. Estos componentes se pueden escribir una sola vez, para posteriormente reutilizarlos. Esto, se realiza escribiendo componentes que posteriormente iremos reutilizando añadiéndolo dicho componente al DOM. Estos componentes podrían ser, por ejemplo: un vídeo, un botón, una imagen barras de navegación, etc.
* **Respaldado por Facebook**: el tener una gran compañía como Facebook detrás de esta tecnología siempre es y será un beneficio.
* **Entró en el mercado en el momento perfecto**: Angular cambió de versión dejando de dar retrocompatibilidad a las versiones anteriores. Provocando un malestar en todos los desarrolladores que en muchas ocasiones se traducía en miedo a que se volviera a repetir esta situación y que decidieran buscar alternativas a Angular debido a la desafortunada decisión que tomó Angular y todo este malestar provocó que muchos programadores se cambiarán hacía React debido a que como hemos dicho en el punto anterior está respaldado por Facebook.
* **Single Page Application (SPA)**: el poder que nos otorga el poder trabajar sobre una única sola página. Nos da el añadido de no tener que estar recargando la página constantemente.
* **Similitudes con otros Frameworks como**: Angular, Vue.JS, Meteor, etc. Por tanto, si aprendemos React, nos será “bastante sencillo” cambiar de un framework a otro ya que tendremos experiencia y tienen un comportamiento “similar”.

**1. MATERIALES**

Para la experiencia a realizar se requiere lo siguiente:

1. EQUIPO

Se requiere tener un equipo de cómputo (PC o laptop).

1. HERRAMIENTAS

* Visual Stude Code
* Node.js on WSL 2
* Npm
* Windows 10
* JavaScript

**2. PROCEDIMIENTO**

Crear aplicación React

Para aprender y probar React, debe configurar un entorno React en su computadora. Esta guía usa el **create-react-app**.

El **create-react-app** es una manera con apoyo oficial para crear aplicaciones React. Si tiene NPM y Node.js instalados, puede crear una aplicación React instalando primero la aplicación create-react-app.

Instale create-react-app ejecutando este comando en su terminal:



¡Ahora está listo para crear su primera aplicación React! Ejecute este comando para crear una aplicación React llamada **myfirstreact**:



El **create-react-app** configurará todo lo que necesita para ejecutar una aplicación React.

**Ejecute la aplicación React**

Si siguió los dos comandos anteriores, ¡está listo para ejecutar su primera aplicación React real! Ejecute este comando para moverse al **myfirstreact** directorio:

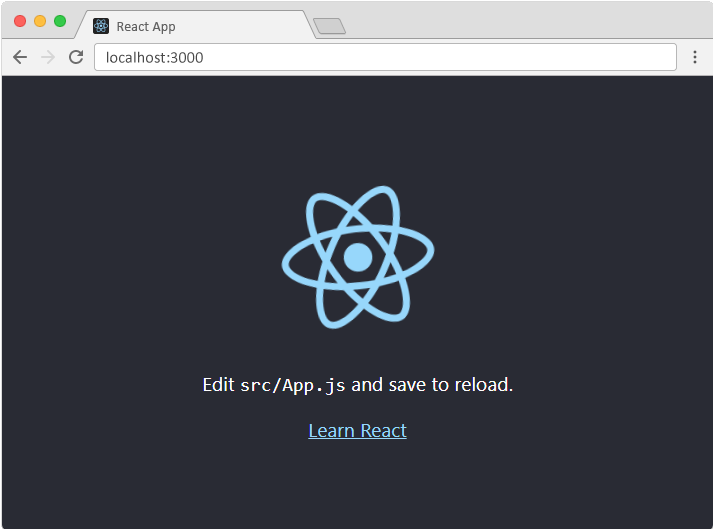


Ejecute este comando para ejecutar la aplicación React **myfirstreact**:



¡Aparecerá una nueva ventana del navegador con su aplicación React recién creada! Si no es así, abra su navegador y escriba **localhost:3000** en la barra de direcciones.

El resultado:

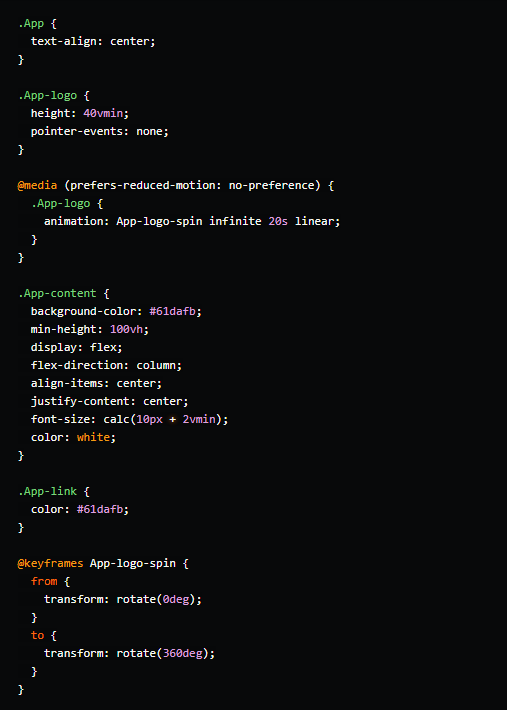


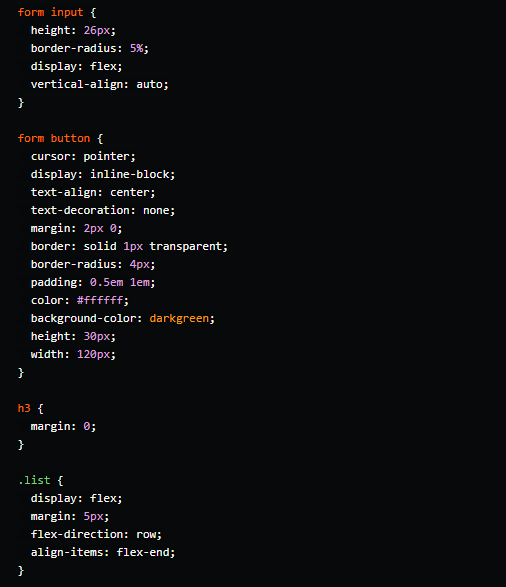
**Proyecto a realizar**

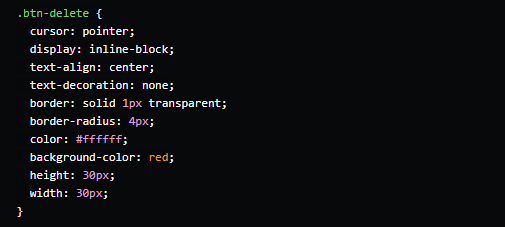
Este proyecto será simple y básico de aplicación de tareas, quizá puede que sea muy sencilla, pero se aprenderá lo que se necesita saber de react-create-app.

**Primeros pasos**

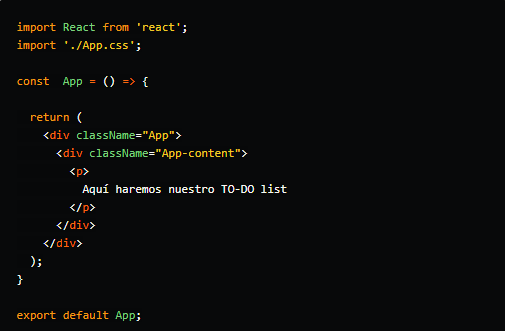
Vamos de lleno al código, vamos a modificar algunos archivos para poder comenzar a trabajar, abrimos el archivo src/App.css y eliminamos el código y copiamos lo siguiente:



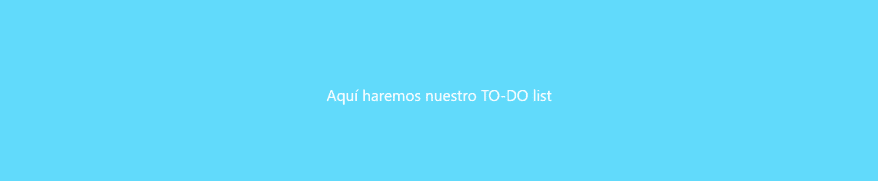




Y ahora en **src/App.js** eliminamos el contenido y copiamos lo siguiente:



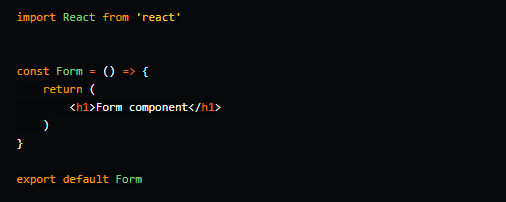
Se hace algunas pequeñas modificaciones a la hoja de estilos que usaremos y al componente **App** el cual solo lo convertimos a una **arrow function**, así que continuemos si vamos a nuestro navegador podemos ver los cambios realizados:



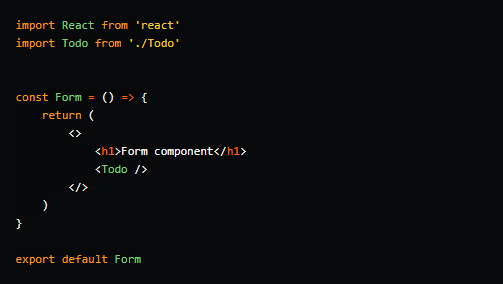
Es todo, no se necesita nada más con lo que a estilos corresponde, se empezará con la parte ruda, lo primero que tenemos que hacer es crear una carpeta en la raíz de **src** a la cual llamaremos **components**, obviamente y como lo imaginas ahí estarán nuestros componentes, para este ejemplo solo crearemos 2, vamos al primero, dentro de **components** creamos un archivo llamado **Todo.jsx**, antes de ir al código te explico, con React podemos utilizar la extensión **.js** o **.jsx** para nuestros componentes sin ningún problema, el motivo por el cual yo uso la extensión **.jsx** es únicamente por mi editor de código, ya que con esta extensión tengo un mejor autocompletado, continuemos agregamos el siguiente código dentro de nuestro componente **Todo**:



Y ahora también dentro de la carpeta **components** creamos un archivo llamado **Form.jsx** con el siguiente código:



Ahora que tenemos ambos componentes hagamos lo siguiente dentro del componente **Form**, agregamos lo siguiente:



Y ahora en nuestro componente **App** hacemos lo siguiente:

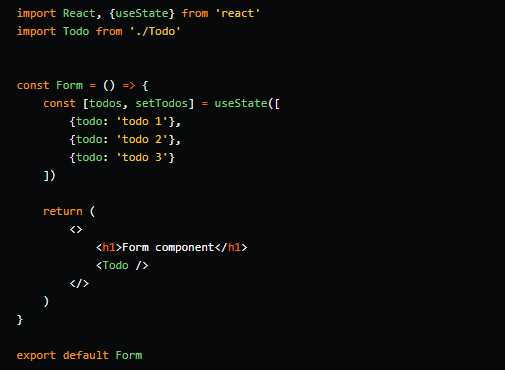


En nuestro navegador ya podemos ver que nuestros dos componentes se han incluido correctamente:

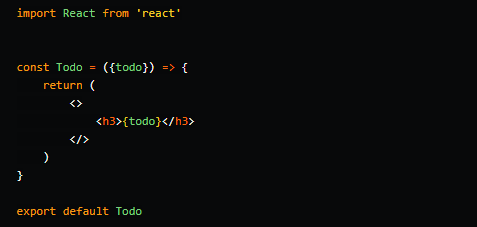


**Construyendo la lógica de nuestra aplicación**

Hasta este punto tenemos todo lo que necesitamos para trabajar, la mayor parte del trabajo lo vamos a realizar en nuestro componente **Form** así que vamos a el, estamos construyendo una aplicación con tareas, entonces necesitamos las tareas (si ya se repetí la palabra tareas, pero es para que no lo olvides), entonces lo primero que haremos es usar uno de nuestros queridos **Hooks** para esto, vamos al código:



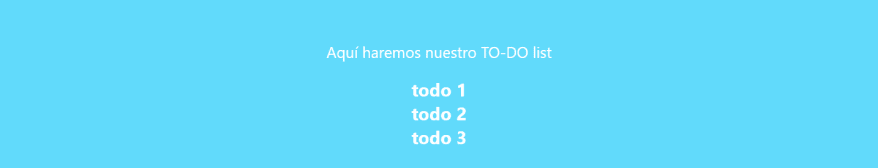
Lo que hicimos aquí fue importar **useState** y declarar un estado dentro de nuestro componente **Form**, el cual es un **array** de **objetos** que serán nuestras tareas, bien, pero esto aún no hace absolutamente nada, así que arreglemos eso, en nuestro componente **Todo** realizamos los siguientes cambios:



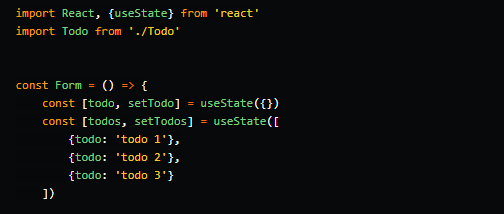
Eliminamos la etiqueta **h1** y en su lugar agregamos un **h3** el cual se encarga de imprimir un **prop** que es únicamente el nombre de nuestra tarea, vamos de nuevo al componente **Form** y agregamos lo siguiente:

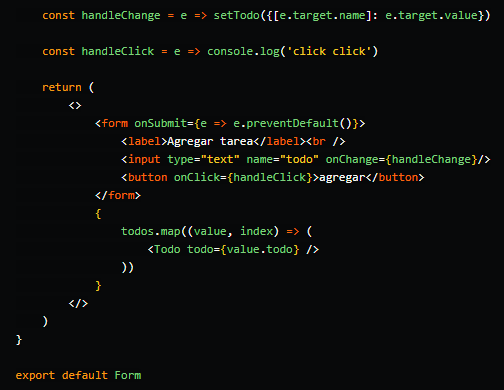


Muy bien, lo que hacemos es recorrer nuestro array de tareas (todos) el cual habíamos inicializado con tres objetos, dentro de la función map incluímos nuestro componente Todo y le pasamos el prop que necesita, si vamos al navegador tenemos el siguiente resultado:

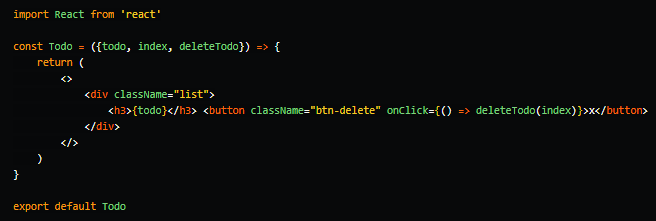


¡Perfecto! ya tenemos nuestras tareas renderizadas, pero ahora necesitamos empezar a agregar más tareas, vamos al código, vamos a realizar unas modificaciones algo amplias en nuestro componente **Form** para que se vea así:



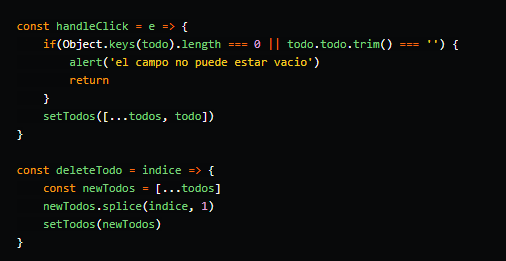


Voy a tratar de explicarte a detalle cada cambio, vamos a empezar con la nueva constante que definimos arriba de nuestro estado de **TAREAS**, agregamos este fragmento de código **const [todo, setTodo] = useState({})**, el cual inicializa un nuevo estado que nos va a servir para agregar UNA tarea, ya que el estado anterior, nos ayuda a crear UNA LISTA de TAREAS, una vez aclarado esto, vamos a la siguiente, la función **const handleChange = e => setTodo({[e.target.name]: e.target.value})**, captura el evento **change** de nuestro **input** el cual se mira ahora así **<input type="text" name="todo" onChange={handleChange}/>**, si te fijas el **input** tiene un atributo **name** en el cual el nombre es el mismo que la **key** de nuestros objetos (tareas), esto es porque en la función **handleChange** recibimos como parámetro el evento como tal y está representado con la variable **e**, **e** nos permite acceder a algunas propiedades entre ellas al **name** del input y al **value** del mismo y estos se encuentran dentro de **target**, es por eso que hacemos **setTodo({[e.target.name]: e.target.value})** dentro de **handleChange**, modificamos el estado de nuestra aplicación para capturar una nueva tarea, la siguiente función se llama **handleClick**, su única función en este momento es imprimir en consola un mensaje, pero mas adelante eso cambiará, para terminar con los detalles nuestro formulario también ejecuta un evento, solo que esta vez no creamos una nueva función sino que ejecutamos una **arrow function** directamente, de nuevo capturamos el evento y hacemos un **e.preventDefault()**, si vienes de **jQuery** creo que sabes perfectamente para que sirve esto, y si no sabes pues es para que al hacer **submit** de nuestro formulario no se refresque nuestro navegador, bien por último nuestro botón ejecuta en el evento **onClick** la función **handleClick**, si en efecto esa que solo imprime un mensajito, perfecto, teniendo ya todo esto vamos a nuestro componente **Todo** para realizar unas modificaciones y se vea de la siguiente manera:

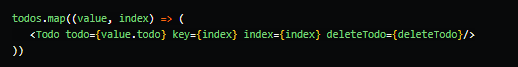


Aquí no hay mucho que explicar, agregamos un boton con una clase **btn-delete**, recordar que en react no podemos usar la palabra **class** en nuestro código **jsx**, recuerda que **jsx** NO es HTML sino una extensión del lenguaje **JavaScript** y **class** es una palabra reservada del lenguaje, en su lugar usamos **className**, aclarado este punto, continuemos, agregamos dos **props** más **index** y **deleteTodo**, estas dos propiedades la cual una es un entero y la otra es una función nos ayudarán más adelante para eliminar tareas, la función **deleteTodo** se ejecuta en el evento **onClick** del botón que acabamos de agregar (comunicación hijos a padres, ¿recuerdas?), hemos terminado nuestro componente **Todo** aquí ya no haremos más.

Volvamos a nuestro componente **Form** modificamos **handleclick** y de igual manera agregamos un nuevo método, así es **deleteTodo** que será el prop que le pasaremos a nuestro componente Todo al igual que **index**, entonces el código se vería de la siguiente manera.



La función **handleClick** ahora ya tiene una funcionalidad, lo primero que hacemos es validar que nuestro input no este vacío y después solo lo agregamos al estado, hacemos uso del **spread operator** dentro de **setTodos** para agregar nuestra nueva tarea y conservar todas las que tenemos, y la función **deleteTodo** obtiene todas las tareas, y elimina la tarea con el indice que le pasamos como parámetro y por último actualizamos el estado con la nueva lista de tareas, para terminar con nuestro ejemplo vamos a modificar la función **map** dentro del **return** para pasarle a **Todo** los props que necesita:



Y nuestra aplicación se ve así:



**ACTIVIDAD VIRTUAL:**

1. **CUESTIONARIO TÉCNICO**

Revisa y analiza el tema desarrollado en la presente sesión, luego responde las siguientes preguntas propuestas:

* ¿Qué es React y para qué sirve?
* ¿Cuáles son las ramas de React? Dé una breve explicación
* ¿Cuáles son las ventajas de React? Dé una breve explicación
* Realice una App aplicando todo lo aprendido de esta guía

1. **ENLACES DE REFERENCIA**

* https://www.w3schools.com/react/react\_getstarted.asp
* https://javadesde0.com/category/front-end/react/level-0-react-fundamentos-beginner/page/4/
* https://dev.to/izakntun/todo-app-con-react-ejemplo-basico-16n0

1. **CONCLUSIONES DE LA EXPERIENCIA**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_