CLASE 4 - Práctica -

Aclaración para todas las prácticas posteriores a la clase: Cada componente generado que reciba props, deberá tener sus propTypes y defaultProps declaradas.

1 - Hacer una lista de pokémons

- Crear un componente llamado “MyPokemons” que muestre en el método render un <input>, un <button> que diga "Agregar" y otro <button> que diga "Vaciar".

- Inicializar el estado de nuestro componente con 2 propiedades: "inputValue" que como default será un string vacío, y otra llamada "items" que inicialmente será un array vacío.

- Hacer que el <input> muestre como valor lo que esté en el estado de nuestro componente en la posición "inputValue", y configurar el evento "onChange" para que actualice el input con lo que vayamos escribiendo.

- Configurar el evento "onClick" de nuestro botón "Agregar" para que modifique el estado y agregue al array "items" una nueva posición con el texto que esté escrito en el input.

- Agregar en nuestro método render el código necesario para generar una lista mostrando los ítems que tengamos en la posición "items" de nuestro estado.

- Configurar el evento "onClick" de nuestro botón "Vaciar" para que modifique el estado vaciando el array de items.

Teniendo nuestra lista:

- Agregar un título que muestre la cantidad de ítems utilizando template strings con el siguiente texto "Total de pokémons: [cantidad]" que solo se muestre cuando haya ítems.

- Crear una condición que no permita agregar ítems duplicados.

- Vaciar el ínput una vez que agregamos un ítem nuevo.

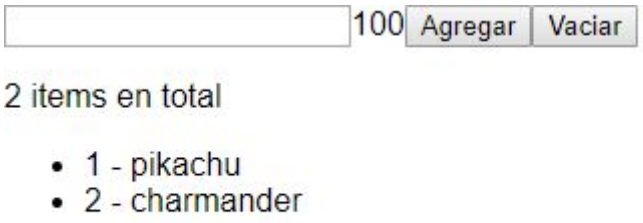
- Preguntarle al usuario utilizando "confirm" si está seguro de vaciar la lista.

- Limitar nuestro ínput a 100 caracteres y agregar entre el ínput y los botones un contador que muestre cuántos nos quedan disponibles para escribir.

Más detalles...

- Hacer que si presionamos la tecla Enter al estar escribiendo en el input se agregue el ítem a la lista tal cual lo hace con el botón “Agregar”.

Se debería lograr algo parecido a la siguiente imagen



2 - Utilizando componentes stateless

- Crear un componente "MyInput" que renderee un <input> y que reciba por props un string "value" y una función "onChange".
  Pasarle ambas props al <input> que renderea. Modificarle al <input> su estilo utilizando clases css con el atributo "className" o css inline con el atributo "style". Reemplazar en nuestro componente “MyPokemons” el <input> por este nuevo componente.

- Crear un componente "MyButton" que renderee un <button> y que reciba por props un string "title" y una función "onClick". La prop "onClick" será pasada a <button> en el atributo "onClick" y la prop "title" la usaremos para mostrar el texto del botón. Modificarle los estilos al botón como en el componente anterior. Reemplazar en nuestro componente “MyPokemons” los <button> por este nuevo componente.

- Crear un componente “MyItems” para el listado de ítems siguiendo la misma lógica que en los puntos anteriores y utilizarlo en “MyPokemons”.

- Crear un componente “MyTotal” para el mostrado del total de ítems y reemplazarlo en “MyPokemons”.

Al completar todos estos puntos, el método render de nuestro componente “MyPokemons” debería quedar parecido a la siguiente imagen

```
render() {
  return (
    <div>
      <MyInput {...propiedadesNecesarias} />
      <MyButton {...propiedadesNecesarias}  />
      <MyButton {...propiedadesNecesarias}  />
      <MyTotal {...propiedadesNecesarias}  />
      <MyItems {...propiedadesNecesarias}  />
    </div>
  )
}
```

donde en lugar de “{...propiedadesNecesarias}” en realidad tendremos las props que cada componente necesita.

3 - Components LifeCycle

- Agregarle a “MyPokemons” un console.log antes de ser montado que diga “MyPokemons será montado” y observar la consola.

- Agregarle a “MyPokemons” un console.log luego de ser montado que diga “[Nombre del componente] fue montado”.

- Agregarle a “MyPokemons” un console.log cuando se actualice el texto del ínput que diga “El texto pasó [textoAnterior] a [textoNuevo]”.

- Agregar en el estado de nuestro componente “MyPokemons” un ítem llamado “loading” que se inicialice en false y otro llamado “imgPokemon” que se inicialice en null.

- Agregar en nuestro método “render” el código necesario para que si “this.state.loading” es true muestre un “ Buscando pokemon... ”, y en caso de que sea false muestre una <img> que en el atributo “src” imprima lo que haya en “this.state.imgPokemon”. Ojo, validar que sólo muestre la imagen si “imgPokemon” es distinto a null.

- Agregar a “MyItems” una nueva prop llamada “onClickItem” y en cada ítem de nuestra lista agregar un evento “onClick” que ejecute onClickItem pasando como parámetro el ítem clickeado.

- En nuestro componente “MyPokemons” configurar un handler para el evento “onClickItem” que:

    - cambie el state para que “loading” sea true, y “imgPokemon” sea null.

    - realice una llamada por ajax (utilizando axios ;)) a la API de pokemon con el pokemon seleccionado y en la respuesta cambie es state volviendo el “loading” a false y en “imgPokemon” poner la imagen frontal del pokemon que viene en la respuesta.

    - En caso de error en la consulta ajax, debemos setear “loading” en false y “imgPokemon” en null.

Ejercicio Integrador:

1. Convertir todos los componentes de nuestra App que no manejen estados en componentes funcionales (stateless). (<GridItem>, <ListItem>, <Button>, <ItemsSection>, etc).

2. Agregar a nuestro componente <ItemsSection> una propiedad “loading” (boolean). En el método render, configurar de modo que si el valor de esta propiedad es TRUE, mostramos el HTML del loading (index-loading.html).

3. Agregar en el método render de nuestro componente <ItemsSection> el código necesario para mostrar un mensaje en caso de que el array de ítems esté vacío. (El html de esto se puede obtener de index-loading.html).

4. Crear un componente “PopularMoviesItemsSection” que:

    a. Inicialice con un state que tenga: loading (bool default true), items (array), type (string).

    b. En el método render que renderee un componente <ItemsSection> pasándole como parámetros:

        i. loading -> this.state.loading
        ii. items -> this.state.items
        iii. type -> “grid” o “list”

    c. Agregarle a nuestro componente <PopularMoviesItemsSection> el método “componentDidMount” y que dentro del mismo ejecutemos un timeout de 5 segundos, que en el callback cambie el estado de loading a true y llene el array de ítems con películas.