Documentacion de practicas del dossier de React

Introduccion

En el siguiente documento iremos viendo las diferentes practicas realizadas del dossier de React. A partir de la practica 5, se realiza con typescript.

Se pueden ver y cargar las diferentes practicas de typescript en el index.tsx del typescriptapp

Para restaurar:

```
$ npm install
```

Practica 1

Crear el hola mundo descrito y agrega tu nombre completo al `<h3>` (usando npx para crear la app y npm start para arrancarla como se indica en el tema).

Realizado en el proyecto pr01-pr04, carpeta src/prácticas/pr01.js .

Practica 2

Realizar lo descrito y tomar captura de pantalla del mensaje en el navegador (recordar que por defecto la web está en el puerto 3000).

Incompleta en el proyecto pr02, carpeta src/prácticas/pr02.js .

Practica 3

Reproducir el ejemplo anterior, pero en lugar de mostrar números primos en el `<h1>` dirá: "mis datos:" y en el `<h4>` le habremos pasado un objeto literal JSON con tu nombre, apellidos y estudios que estás realizando.

Realizado en el proyecto pr01-pr04, carpeta src/prácticas/pr03.js .

Practica 4

Reproducir el ejemplo anterior, pero cambiando que los atributos que reciba sean: `num1` y `num2`, y lo que muestre es: *La suma de num1 y num2 es: num1 + num2* (donde `num1` y `num2` serían los datos que recibiera el componente).

Realizado en el proyecto pr01-pr04, carpeta src/prácticas/pr04.js .

Practica 5

Reproducir el ejemplo anterior de componente con TypeScript, pero cambiando que los atributos que reciba sean de tipo numérico: `num1` y `num2`, y lo que muestre es: *La suma de num1 y num2 es: num1 + num2* (donde `num1` y `num2` serían los datos que reciba el componente).

Realizado en la carpeta typescript ruta src/practicas/pr05

Practica 6

Generar el renderizado anterior, creando el componente `Reloj.ts` apropiado para obtener el resultado mostrado.

Realizado en la carpeta typescript ruta src/practicas/pr06

Practica 7

Probar el código de un botón que muestra la hora al pulsarlo. Tomar captura de pantalla del navegador al pulsar el botón.

Incompleto

Practica 8

Crear el código de componente `Contador` (clásico) en un fichero nuevo y cargarlo en `index.tsx` en la parte de renderizado con `ReactDOM.render()`. Probarlo en el navegador y comprobar que efectivamente cambia el contador con los clics.

Realizado en la carpeta typescript ruta src/practicas/pr08

Practica 9

Realizar con el Hook useState dentro de un functional component un componente que sirva a un usuario para prácticar la tabla del 2. Cada vez que pulse en el botón se le mostrará la solución correcta de la tabla. Así: la primera vez que haga clic se le mostrará:

2x1 = 2 La segunda vez: 2x2=4

y así sucesivamente.

En definitiva: que vaya mostrando la tabla del 2 a cada click Observar que después de 2x10 mostrará 2x1

Realizado en la carpeta typescript ruta src/practicas/pr09

Practica 10

Crear un functional component react (usa el snippet: tsrafc) que tenga un botón. Este botón al pulsarlo va agregando un nuevo número aleatorio de 0 a 100 de tal forma que podemos ver gracias al state toda la lista de aleatorios generados (Nota: podemos usar: JSON.stringfy(nombredelarray) para ver el array u otro objeto) Nota: hay una forma sencilla de crear un nuevo array con un nuevo elemento conservando los datos del anterior. Imaginemos que queremos agregar el número 5: const arrayanterior: Array = [4, 2, 7]; [...arrayanterior, 5]

Realizado en la carpeta typescript ruta src/practicas/pr10

Practica 11

Crear el anterior functional component, ejecútalo y abre la consola ¿ se está actualizando la información del atributo estático ? ¿ y de la variable: dato ? Ahora quita el comentario de la línea: sethoraactual(“” + new Date()); Sabemos que de esa manera al actualizar el state se fuerza un nuevo renderizado ¿ se está actualizando la info del atributo estático ? ¿ y de la variable: dato ?

Se están actualizando, pero no se renderiza de nuevo así que no se ve más que en el console.log();

Ahora si se ve en la página .

Realizado en la carpeta typescript ruta src/practicas/pr11

Practica 12

Crear un componente que muestre un mensaje según el color elegido (verde o azul), utilizando un `useState` y un método que reciba parámetros en el `onClick`.

Incompleto?

Practica 13

Reproducir el ejemplo de lista de monedas usando `map`. Mostrar cada moneda con su país en el formato: *moneda de país* (por ejemplo: *libra de UK*).

No realizado

Practica 14

Crear un componente `TodasLasTablas` que utilice un componente `PracticarTabla` para mostrar las tablas del 2 al 10, usando un `map` para recorrer un array de números del 2 al 10.

Realizado en la carpeta typescript ruta src/practicas/pr14

A partir de aquí me quedé, y empecé a realizar solo las practicas. Están todas siguiendo el mismo formato hasta la 51, hay varias incompletas, pero no me daba tiempo a arreglarlas todas ya...

Practica 15

Crear un componente donde, al pulsar botones con diferentes colores (rojo, verde, azul), se cambien los estilos CSS de un área para reflejar el color seleccionado.

Practica 16

En la práctica de relojes de zonas horarias, crear un array con 5 zonas horarias (incluyendo Londres y Madrid). Usar `map` para generar componentes `Reloj` con sus respectivas propiedades `timezone`, y darles estilos CSS.

Practica 17

Crear un componente con dos botones. Al pulsar el primer botón, se cargará un componente que muestra 10 números aleatorios de 0 a 100. Al pulsar el segundo botón, se reemplazará con un componente que muestra un saludo y la fecha actual (pasada como props).

Practica 18

Implementar ejemplos con `useEffect` y componentes tradicionales (`componentDidMount` y `componentDidUpdate`). Adaptarlos a TypeScript.

Practica 19

Realizar un ejemplo con `useEffect` para entender su funcionamiento en los ciclos de vida. Responder preguntas sobre su ejecución al modificar estados y renderizados.

Practica 20

Crear un componente para un juego de adivinar el número secreto (de 0 a 9). Tendrá 10 botones; al pulsar en uno, el programa evaluará si el número es correcto o indicará si es mayor o menor.

Practica 21

Crear un cronómetro usando `setInterval` y `useEffect`. Responder preguntas sobre su funcionamiento y explicar el significado del intervalo de tiempo configurado.

Practica 22

Modificar la práctica de los relojes mundiales para que se muestren con la hora actualizada cada segundo usando `setInterval`.

Practica 23

Usar `useRef` para crear un componente con dos inputs (nombre y apellidos) y un párrafo. Al pulsar un botón, mostrar el nombre completo en el párrafo y la cantidad de letras que tiene.

Practica 24

Modificar el ejercicio de adivinar el número secreto para usar un único input y botón. Tomar la información del input mediante `useRef`.

Practica 25

Crear un componente con dos botones. Uno agrega números aleatorios a un array usando `useRef`, y el otro muestra los números almacenados en el estado.

Practica 26

Crear un componente que tenga un input y un botón. Dependiendo del valor ingresado (número o palabra), cargar un componente que muestra una tabla de multiplicar o analiza el texto ingresado.

Practica 27

Crear un cronómetro donde el usuario introduzca una cantidad de segundos y al pulsar un botón comience la cuenta atrás.

Practica 28

Crear un componente `MostrarInput` que muestre en un `h5` el texto ingresado en un input en tiempo real.

Practica 29

Crear un componente con dos botones. Uno divide el valor actual entre 2 y el otro lo multiplica. Ambos deben compartir un único método para manejar el evento `onClick`.

Practica 30

Implementar el juego de adivinar el número secreto con un input y botón, utilizando manejo de eventos en lugar de referencias.

Practica 31

Crear un juego de memoria donde el usuario deba recordar 8 números que se muestran durante 3 segundos y luego seleccionarlos en orden mediante botones.

Practica 32

Crear un componente con un formulario que permita filtrar productos por nombre y muestre el resultado en una lista con estilo CSS.

Practica 33

Crear un componente con un formulario que tome dos números y muestre los números primos entre esos valores.

Practica 34

Crear un componente que calcule la edad humana de un perro en función de su tamaño y edad, usando un formulario con un input y radio buttons.

Practica 35

Crear tres componentes donde un componente padre contenga dos hijos: uno con un input que actualiza el estado del padre y otro con un botón que lo modifica.

Practica 36

Crear un componente que gestione un array de personas, donde cada persona tenga su propio componente para editar datos.

Practica 37

Crear un componente padre que pase un número aleatorio al hijo. El hijo debe mostrarlo en un input editable y descomponerlo en un textarea.

Practica 38

Crear un componente padre con un array de usuarios que permita modificar los nombres desde componentes hijos mediante botones.

Practica 39

Crear un componente `InputToUpper` que transforme el texto ingresado en un input a mayúsculas en tiempo real.

Practica 40

Crear un reproductor multimedia con una lista de reproducción, donde el usuario pueda seleccionar una canción para reproducir.

Practica 41

Implementar enrutado SPA básico en React, con rutas para al menos dos componentes como `Cronometro` y `RelojesMundiales`.

Practica 42

Crear un sistema de navegación con múltiples rutas y un menú que permita cambiar entre ellas.

Practica 43

Implementar un ejemplo de `useParams` para capturar parámetros de la URL y mostrarlos en un componente.

Practica 44

Crear un formulario que envíe datos a una API con Axios y muestre el resultado en un componente.

Practica 45

Implementar un sistema de login que utilice `useContext` para manejar el estado de autenticación.

Practica 46

Crear un componente que use `localStorage` para guardar datos persistentes y mostrarlos al recargar la página.

Practica 47

Crear un sistema que controle el acceso a rutas protegidas en React utilizando tokens.

Practica 48

Crear un componente que utilice `useReducer` para manejar el estado de un formulario.

Practica 49

Implementar un ejemplo de `useNavigate` para redirigir entre componentes al completar una acción.

Practica 50

Crear un sistema de registro y autenticación con React, usando almacenamiento de tokens y rutas protegidas.

Practica 51

Crear un sistema de enrutado avanzado que combine `useContext` y `useReducer` para manejar la navegación y estado global.