Trabajo de Aplicaciones Móviles


```
Nombre : Juan Carlos Jara


Docente: Ing.Jhonathan Vallejo


Carrera: Desarrollo de Software


Ciclo: 3 Nivel


Tema: Resumen de el Video
```

`**RESUMEN DEL VIDEO.**`

Ejercicio 2

`- La palabra clave` export `se utiliza para hacer que la función` twoFer` esté disponible para otros archivos o módulos que importen este archivo.

- `function twoFer(name: string = 'you')`: Esto define una función llamada `twoFer` que acepta un parámetro llamado `name`. El tipo de `name` se especifica como `string`, lo que significa que debe ser una cadena de texto. Además, se le asigna un valor predeterminado de `'you'`, lo que significa que si no se proporciona ningún valor para `name`, se utilizará `'you'`.
- `: string {`: Esto indica que la función `twoFer` devuelve un valor de tipo `string`.
- 1. `return One for ${name}, one for me.;`:

- `return`: Esta palabra clave indica que la función está devolviendo un valor.
- `One for ${name}, one for me.`: Aquí se utiliza una **plantilla literal** (template literal) para construir una cadena de texto. La parte `${name}` se reemplaza por el valor del parámetro `name`. Por ejemplo, si llamas a la función con `twoFer('Alice')`, esta línea se convertirá en `"One for Alice, one for me."`. Si no se proporciona un valor para `name`, se utilizará el valor predeterminado `'you'`, y la cadena resultante será `"One for you, one for me."`.

- 1. `import React from 'react';`: Esta línea importa la biblioteca `React` para que podamos usar sus componentes y funciones en nuestro archivo.

2. `const TwoFerComponent = ({ name = 'you' }) => {`: Aquí estamos definiendo un componente funcional llamado `TwoFerComponent`. Toma un objeto de propiedades como argumento, y si no se proporciona un valor para `name`, se establece por defecto como `'you'`.
3. `return ( ... );`: El componente devuelve un fragmento de JSX (JavaScript XML) dentro de los paréntesis. Este fragmento representa el contenido que se renderizará en la interfaz de usuario.
4. `<div> ... </div>`: Esto crea un elemento `<div>` que contiene el siguiente texto: “One for {name}, one for me.”. El valor de `{name}` se reemplazará por el valor proporcionado en las propiedades del componente.
5. `export default TwoFerComponent;`: Finalmente, exportamos el componente `TwoFerComponent` para que pueda ser utilizado en otros archivos de la aplicación.
   ``import React from 'react';

```
const TwoFerComponent = ({ name = 'you' }) => {

 return (

  <div>

   One for {name}, one for me.

  </div>

 );

};``

export default TwoFerComponent;
```

Ejercicio 3

1. `export const colorCode = (color: string) => { return COLORS.indexOf(color) }`:

- `export const colorCode`: Esto define una **función flecha** llamada `colorCode`. La función acepta un parámetro llamado `color`, que debe ser de tipo `string`.
- `(color: string) => { return COLORS.indexOf(color) }`: Esta es la implementación de la función. Aquí, se utiliza una función flecha para definir la lógica. La función toma el color proporcionado como argumento y busca su índice dentro del arreglo `COLORS`. Luego, devuelve ese índice.

2. `export const COLORS = [`:
   - Esto define una **constante** llamada `COLORS`.
   - El valor de `COLORS` es un **arreglo** que contiene los siguientes colores
3. `import React, { useState } from 'react';`: Importamos la biblioteca `React` y la función `useState` desde el módulo `'react'`. `useState` es un hook que nos permite gestionar el estado en componentes funcionales.
4. `const COLORS = [ ... ];`: Creamos una constante llamada `COLORS` que contiene una lista de colores. Cada color es una cadena de texto.
5. `const ColorCodeComponent = () => {`: Definimos un componente funcional llamado `ColorCodeComponent`.
6. `const [inputColor, setInputColor] = useState('');`: Utilizamos el hook `useState` para crear una variable de estado llamada `inputColor` con un valor inicial vacío. También creamos una función `setInputColor` para actualizar este estado.
7. `const [colorIndex, setColorIndex] = useState(null);`: Similar al paso anterior, creamos otra variable de estado llamada `colorIndex` con un valor inicial de `null`.
8. `const handleColorSearch = () => { ... };`: Definimos una función llamada `handleColorSearch`. Esta función se ejecutará cuando el usuario haga clic en el botón de búsqueda.
9. `const colorCode = (color) => { ... };`: Creamos otra función llamada `colorCode` que toma un color como argumento y devuelve su índice en la lista `COLORS`.
10. El resto del código dentro del componente renderiza una lista de colores con sus índices, un campo de entrada de texto para escribir un color, un botón de búsqueda y muestra el índice del color ingresado si se encuentra en la lista.

```
import React, { useState } from 'react';

const COLORS = [

 'black',

 'brown',

 'red',
```

```
  'orange',
  'yellow',
  'green',
  'blue',
  'violet',
  'grey',
  'white',
];
const ColorCodeComponent = () => {
  const [inputColor, setInputColor] = useState('');
  const [colorIndex, setColorIndex] = useState(null);
  const handleColorSearch = () => {
    const index = colorCode(inputColor);
    setColorIndex(index);
  };
  const colorCode = (color) => {
    return COLORS.indexOf(color);
  };
  return (
    <div>
      <p>Índices de colores:</p>
      <ul>
        {COLORS.map((color, index) => (
          <li key={color}>
```

```
        {color}: {index}
      </li>
    ))}
  </ul>
  <input
    type="text"
    value={inputColor}
    onChange={(e) => setInputColor(e.target.value)}
    placeholder="Escribe un color..."
  />
  <button onClick={handleColorSearch}>Buscar</button>
  {colorIndex !== null && (
    <p>
      El índice del color {inputColor} es: {colorIndex}
    </p>
  )}
 </div>
 );
};

export default ColorCodeComponent;
```

ENLACE DE YOUTUBE

https://youtu.be/MwXSSMZc_gk