Módulo 2 – Estilos y Event Handlers

# Repaso

* Explicar el uso correcto de falsy y truly. Explicar que && rey
* Mostrar la implementación del pro “verified”

# Librerías para estilos

Slide 2

Muchas veces cuando uno debe empezar un nuevo proyecto desde cero para un cliente por encargo, uno no desea estar reinventando la rueda. Uno desea enfocarse en desarrollar e implementar la lógica del negoció o los comportamientos que la aplicación web tiene que cumplir, la definición de los datos y no estar definiendo desde cero los estilos de la UI o creando varias veces un kit de reglas CSS transversales.

Para ello existen librerías que nos pueden ayudar simplemente usar su conjunto de reglas precargadas o facilitar la definición de estilos, como es el caso de styled-componentes y Tailwind. Este último es el que usaremos en el curso para aplicar estilos de manera más rápida a nuestros componentes.

Cabe recalcar que esto no significa que no es necesario aprender y conocer cómo funciona CSS y estar al tanto de la evolución de este. Debido a que, en proyectos más grandes, con un grupo de devs donde la empresa necesita de una identidad clara de su aplicación es más sencillo estar lo más cercano a la plataforma para definir el estilo de la empresa en la aplicación web. Entiéndase “plataforma” como el entorno donde nuestra aplicación se ejecuta, en el caso de una web sería el navegador donde lo más básico todo se resume a CSS, HTML y JavaScript.

# Flexbox

Slide 3

Realizar una pequeña practica y demostración de Flex box con la página web <https://flexboxfroggy.com> para que puedan hacer el correcto uso de Flex con Tailwind para la siguiente actividad. Solo realizar hasta el **nivel 6** que es donde se puede demostrar como justify-content funciona en base al primary-axis y el align-items en el cross-axis. Explicar que por defecto flex-direcction es row pero se puede cambiar a columna.

# Configurar Tailwind en ReactJS (ViteJS)

Slide 4 – 5

Para poder hacer uso de las clases de Tailwind primero debemos configurarlo en el proyecto y usar la extensión de Tailwind en VS Code, para esto debemos revisar la documentación y seguir los pasos añadirlo a ViteJS que es el empaquetador que estamos usando para convertir toda nuestra aplicación de ReactJS a el simple HTML, CSS y Javascript que el navegador si puede entender.

# Práctica con Tailwind

Slide 6 - 8

El entorno del desarrollo web es muy frenético. Suele salir nuevas librerías, incluso, nuevas versiones de las que ya conocemos y pueden cambiar mucho, u ofrecer otro enfoque para realizar “lo mismo”. Es imposible quizás poder implementarlo y probarlo al 100% todo, sin embargo, lo que sí es importante es estar consiente de estas opciones emergentes y leer su documentación como mínimo. Entender su funcionamiento, detectar sus fortalezas y debilidades para saber cuándo optar por una o no. Generalmente todo depende de la combinación de la complejidad y tamaño del equipo de trabajo.

Así como existen muchísimas propiedades CSS y algunas específicas para cada elemento HTML, existen muchísimas clases de Tailwind que se corresponden a cada propiedad directa en CSS. No es necesario aprenderlas de memoria cada una de ellas, pero si es recomendable tener la noción de cuál es la que demos usar o saber cómo buscar las propiedades que pueda cumplir con nuestras necesidades.

Para saber cuál es la clase que debemos usar para aplicar un estilo podemos buscarlo en la documentación de Tailwind <https://tailwindcss.com/docs>

Por ser la primera vez que se usa Tailwind, se puede hacer una guía de al menos como modificar el componente App.tsx para que el componente UserProfile se centre vertical y horizontalmente.

Para la práctica debemos estilizar nuestro proyecto actual para que luzca lo más similar a lo que tenemos como objetivo. Algunos requisitos son:

* Agrupar a los componentes de manera horizontal.
* Centrarlos de manera tanto vertical como horizontal.
* Añadir un borde y sombra del componente UserProfile.
* En caso de que el usuario esté verificado el texto que lo indica debe ser de color Verde, caso contrario Rojo.
* El texto de cada UserProfile debe estar centrado.

# Event Handlers

Slides 9 - 10

Los manejadores de eventos en JSX son propiedades que reciben una función. Algunos de estas propiedades pueden enviar parámetros a la función que declaramos, como es el caso de los eventos “onChange”, el cual recibe un evento que contiene el nuevo valor que ha cambiado en el elemento HTML.

Recordemos que los props pueden además de valores primitivos, JSON y Clases también pueden recibir funciones, por lo que un componente puede declarar que puede recibir un prop de tipo función que luego hará uso como evento handler en alguno del elemento HTML que contiene.

Hay que tener en consideración que lo que se debe enviar en el prop de evento handler es una función anónima o la referencia de una función, pero no la invocación de la función, de ser ese el caso la lógica se ejecutará cuando el componente se renderice y no cuando el evento se ha disparado, ya sea un “onClick”, “onChange”, etc.

# Práctica Event Handlers

Slide 11 - 12

Agregar un botón debajo del componente que tenga como texto “Copiar datos”, al dar clic este debe copiar en el portapapeles los datos del usuario en texto plano y mostrar una alerta en el navegador indicando que los datos han sido copiados exitosamente.

No teman en buscar en internet como acceder o manipular el portapapeles desde el navegador, siempre se debe recurrir a la documentación cuando no se conoce de alguna tecnología o se está aprendiendo de ella.

# Extras

Leet sobre "Routing Design Patterns" para aplicaciones web.

Un ejemplo puede ser el diseño de rutas que existe en el portafolio, el ejemplo entre las sección y en detalle de los proyectos. Se aconseja a siempre empezar con el nombre de la entidad y luego con la acción y por último el ID de la entidad en caso de tenerla.

Preparar el ambiente para react-router, instalar las dependencias y el Root del router.