INSTITUTO TECNOLÓGICO DE CIUDAD MADERO

TAREA #2:

"Aplicación del Clima (Weather App)"

Integrantes:

Anastasio Salas Juan Carlos

Cabrera Duran Itzel Amayramy

Materia:

Tecnologías de Aplicaciones Web

Horario:

14:00 - 15:00 hrs.

Cd. Madero, Tamaulipas

DEJANDO HUELLA

TAREA NO. 2: Aplicación del Clima (Weather App)

INSTRUCCIONES:

- **Funcionalidad:** Muestra el clima actual y el pronóstico de una ciudad.
- **Punto clave:** Necesitarás consumir una API de clima pública (como OpenWeatherMap o Weather API) para obtener los datos, lo que te ayudará a practicar la manipulación de datos asíncronos en JavaScript.
- **Lenguajes por utilizar:** HTML 5, CSS3 y Javascript.

DESARROLLO:

Descripción General

El proyecto es una Aplicación del Clima (Weather App) desarrollada con tecnologías web estándar (HTML, CSS, JavaScript). Su diseño sigue una estética moderna utilizando efectos de Glassmorphism (fondos translúcidos y desenfocados) y degradados vibrantes.

¿Qué hace?

La aplicación permite al usuario ingresar el nombre de una ciudad para consultar en tiempo real:

- La temperatura actual.
- Una descripción breve del clima (ej. "light rain").
- Un icono representativo de las condiciones actuales.
- Un pronóstico por horas para las próximas 24 horas (dividido en intervalos de 3 horas).

Tecnología Clave: Utiliza la API de OpenWeatherMap (/weather para clima actual y /forecast para pronóstico) para obtener los datos meteorológicos.

Ejecución Jerárquica

La ejecución del código sigue un flujo lineal desencadenado por la acción del usuario. El orden de llamadas es el siguiente:

1. getWeather(): Función principal iniciada por el usuario (click en botón).

- Llama a: API de OpenWeatherMap.
- Si es exitoso, llama a: displayWeather(data) (para clima actual).
- Si es exitoso, llama a: displayHourlyForecast(data.list) (para pronóstico).

2. displayWeather(data): Procesa y muestra la información actual.

- Llama a: showImage() (para hacer visible el icono).

3. displayHourlyForecast(hourlyData): Procesa y genera la lista visual del pronóstico.

Análisis Detallado de Funciones

A continuación, se explica cada función línea por línea según su lógica interna:

getWeather()

Función Principal (Trigger): Es el punto de entrada de la lógica. Se ejecuta cuando el usuario hace clic en el botón "Search".

Procesos:

1. **Captura de Datos:** Obtiene el valor ingresado en el input #city.
2. **Validación:** Si el campo está vacío, lanza una alerta y detiene la ejecución.
3. **Construcción de URLs:** Crea dos URLs dinámicas para la API usando la API Key y la ciudad:
    - **currentWeatherUrl:** Para el clima actual.
    - **forecastUrl:** Para el pronóstico de 5 días / 3 horas.
4. **Petición Clima Actual (fetch):**
    - Solicita datos a currentWeatherUrl.
    - Convierte la respuesta a JSON.
    - Llama a displayWeather(data) pasando los datos recibidos.
    - Incluye manejo de errores (catch) para notificar problemas de red.
5. **Petición Pronóstico (fetch):**
    - Solicita datos a forecastUrl.
    - Convierte la respuesta a JSON.
    - Llama a displayHourlyForecast(data.list) pasando la lista de pronósticos.
    - Incluye manejo de errores independiente.

displayWeather(data)

Renderizado del Clima: Actual Se encarga de pintar en el DOM la información principal del clima.

Props (Argumentos):

- data: Objeto JSON crudo devuelto por la API de OpenWeatherMap (/weather).

Procesos:

1. Selección de Elementos: Obtiene referencias a los contenedores .temp-div, .weather-info, #weather-icon, etc.
2. Limpieza: Borra el contenido HTML previo de estos contenedores para evitar duplicados.
3. Manejo de Errores de API: Verifica si data.cod es '404' (Ciudad no encontrada). Si es así, muestra el mensaje de error en pantalla.
4. Extracción de Datos: Si la ciudad existe, extrae:
    - cityName: Nombre de la ciudad.
    - temperature: Temperatura redondeada (Math.round).
    - description: Descripción textual del clima.
    - iconCode: Código del icono (ej. "10d").
5. Generación de URL de Icono: Construye la URL de la imagen del icono @4x.png (alta resolución).
6. Inyección al DOM: Crea strings de HTML con los datos y los asigna al innerHTML de los contenedores correspondientes.
    - Actualiza el src y alt de la etiqueta <img> del icono principal.
7. Visualización: Llama a showImage() para mostrar el icono (que estaba oculto por defecto).

showImage()

Utilidad Visual: Una función auxiliar simple para manejo de estilos.

Props: Ninguna.

Procesos:

1. Selecciona el elemento #weather-icon.
2. Cambia su estilo CSS display a 'block', haciéndolo visible (inicialmente está none en CSS).

displayHourlyForecast(hourlyData)

Renderizado del Pronóstico: Genera la lista horizontal de predicciones futuras.

Props (Argumentos):

- **hourlyData:** Un array de objetos, donde cada objeto representa un intervalo de 3 horas del pronóstico (viene de data.list).

Procesos:

1. **Selección:** Obtiene el contenedor #hourly-forecast.

2. **Filtrado de Datos:** Usa hourlyData.slice(0, 8) para tomar solo los próximos 8 elementos. Como cada intervalo es de 3 horas, esto cubre las próximas 24 horas (8 * 3 = 24).
3. **Iteración (forEach):** Recorre estos 8 elementos y por cada uno:
    - Calcula la hora legible usando new Date(item.dt * 1000).getHours().
    - Redondea la temperatura.
    - Construye la URL del icono correspondiente.
    - Crea una plantilla HTML (hourlyItemHtml) con la hora, el icono y la temperatura.
    - Añade (+=) este HTML al contenedor principal.