Exercicio 60: Aplicación do tempo con API

Descrición:

Neste exercicio, imos **crear unha aplicación do tempo que utiliza unha API p**ara obter datos meteorolóxicos en tempo real.

Obxectivo:

O obxectivo principal é entender como funciona o método *fetch()* en JavaScript para consumir unha API e como se poden utilizar estes datos para crear unha interface de usuario interactiva.

Que é unha API?

Unha **API** (Interface de Programación de Aplicacións) **é un conxunto de definicións e protocolos que permiten que diferentes aplicacións se comuniquen entre si.** No contexto deste exercicio, utilizaremos a API de *OpenWeatherMap*, que nos proporciona datos meteorolóxicos actualizados para diferentes localizacións.

Obxectivos do exercicio:

1. Aprender a utilizar *fetch*() para realizar chamadas a unha API externa.
2. Entender como procesar e mostrar os datos recibidos dunha API.
3. Practicar o uso de *BulmaCSS* para crear unha interface de usuario atractiva e responsiva.

Compoñentes principais do exercicio:

1. **Chamada á API**: Utilizaremos *fetch*() para solicitar datos á API de *OpenWeatherMap*.
2. **Procesamento de datos**: Aprenderemos a manexar a resposta da API e extraer a información relevante.
3. **Visualización**: Mostraremos os datos do tempo (temperatura, icona, humidade...) de forma atractiva.
4. **Deseño responsivo**: Implementaremos un deseño adaptable utilizando as clases de *BulmaCSS*.

Pasos a seguir:

Para realizar esta páxina web necesitarás as seguintes estruturas e elementos:

1. **HTML** (index.html):

- **Assets**: Na carpeta *assets* atoparás as **imaxes.**
- **Scripts**: Vincula o ficheiro JavaScript.

2.**CSS** (style.css):

- **Crear a folla de estilos**: Crea o ficheiro `*style.css*` e non esquezas vinculalo ao `*index.html*`
- **Descarga e vincula a folla de estilos de BulmaCSS (https://bulma.io)**
- Na carpeta *assets* atoparás os recursos restantes para completar o deseño como as **fontes tipográficas e as imaxes**.

3.**JavaScript** (script.js):

- Para poder facer uso da API de *OperWeatherMaps* é preciso facerse unha conta, xa que **necesitamos unha API Key ou clave de acceso a API**. Podes crear unha conta gratuíta no seguinte enlace: https://openweathermap.org/price
- A **función principal que imos usar para obter os datos é a dun *fetch*** , que acostuma a ser sempre unha estrutura moi semellante a que mostramos a continuación:

```
async function obterDatosTempo(lat, lon) {
    const apiURL = `URL da API xunto cos parámetros`
    try {
        const response = await fetch(apiURL);
        return await response.json();
    } catch (error) {
        console.error("Error al obtener datos del clima:",
error);
    }
}
```

Explicación desta función paso a paso:

1. Declaración da función asíncrona

```
async function obterDatosTempo(lat, lon) {
```

- **async**: Indica que é unha función asíncrona, o que permite usar await dentro dela.
- **Parámetros**: Recibe latitude (lat) e lonxitude (lon) para localizar xeograficamente a consulta meteorolóxica.

2. Construción do URL da API

```
const apiURL = `URL da API xunto cos parámetros`;
```

- **Dinamismo**: Aquí se construiría o URL real da API de OpenWeatherMap cos parámetros necesarios, como:
  - **lat**: Latitude da localización
  - **lon**: Lonxitude da localización
  - **appid**: Clave de API do usuario
  - **units=metric**: Para obter temperaturas en Celsius
  - **lang=gl:** Para recibir descricións en galego

3. Bloque try (Intento de conexión)

```
try {
    const response = await fetch(apiURL);
```

- **await fetch()**: Pausa a execución ata recibir resposta da API.
- **fetch()**: Método nativo de JavaScript para facer peticións HTTP.

4. Procesamento da resposta

```
return await response.json();
```

- **response.json()**: Converte a resposta da API (en formato JSON) a un obxecto JavaScript usable.
- **Segundo await**: Necesario porque a conversión a JSON tamén é unha operación asíncrona.

5. Bloque catch (Xestión de erros)

```
} catch (error) {
    console.error("Error ao obter datos do tempo:", error);
}
```

- Captura calquera erro durante o proceso (rede, URL incorrecto, datos inválidos...).
- **console.error**: Amosaría o erro na consola para depuración.

Entrega: Unha vez completado todo **subiremos a carpeta *Exercicio60*** ao espazo para entrega de exercicios que temos no **OneDrive** de **Microsoft Teams**.

Ejercicio 60: Aplicación del tiempo con API

Descripción:
En este ejercicio, vamos a **crear una aplicación del tiempo que utiliza una API** para obtener datos meteorológicos en tiempo real.

Objetivo:
El objetivo principal es entender cómo funciona el método ***fetch***() en JavaScript para consumir una API y cómo se pueden utilizar estos datos para crear una interfaz de usuario interactiva.

¿Qué es una API?
Una **API** (Interfaz de Programación de Aplicaciones) **es un conjunto de definiciones y protocolos que permiten que diferentes aplicaciones se comuniquen entre sí.** En el contexto de este ejercicio, utilizaremos la API de ***OpenWeatherMap***, que nos proporciona datos meteorológicos actualizados para diferentes localizaciones.

Objetivos del ejercicio:

1. **Aprender a utilizar fetch()** para realizar llamadas a una API externa.
2. **Entender cómo procesar y mostrar los datos** recibidos de una API.
3. Practicar el uso de **BulmaCSS** para crear una interfaz de usuario atractiva y responsiva.

Componentes principales del ejercicio:

1. **Llamada a la API**: Utilizaremos fetch() para solicitar datos a la API de OpenWeatherMap.
2. **Procesamiento de datos**: Aprenderemos a manejar la respuesta de la API y extraer la información relevante.
3. **Visualización**: Mostraremos los datos del tiempo (temperatura, icono, humedad...) de forma atractiva.
4. **Diseño responsivo**: Implementaremos un diseño adaptable utilizando las clases de BulmaCSS.

Pasos a seguir:
Para realizar esta página web necesitarás las siguientes estructuras y elementos:

1. **HTML** (index.html):

- Assets: En la carpeta assets encontrarás las imágenes.
- Scripts: Vincula el archivo JavaScript.

2. **CSS** (style.css):

- Crear la hoja de estilos: Crea el archivo style.css y no olvides vincularlo al index.html
- Descarga y vincula la hoja de estilos de BulmaCSS (https://bulma.io)
- En la carpeta assets encontrarás los recursos restantes para completar el diseño como las fuentes tipográficas y las imágenes.

3. **JavaScript** (script.js):

- Para poder hacer uso de la API de OpenWeatherMaps es necesario hacerse una cuenta, ya que **necesitamos una API Key o clave de acceso a API**. Puedes crear una cuenta gratuita en el siguiente enlace: https://openweathermap.org/price
- **La función principal que vamos a usar para obtener los datos es la de un *fetch***, que suele ser siempre una estructura muy similar a la que mostramos a continuación:

```
async function obtenerDatosTiempo(lat, lon) {
    const apiURL = `URL de la API junto con los parámetros`
    try {
        const response = await fetch(apiURL);
        return await response.json();
    } catch (error) {
        console.error("Error al obtener datos del clima:",
error);
    }
}
```

Explicación de esta función paso a paso:

1. **Declaración de la función asíncrona**

```
async function obtenerDatosTiempo(lat, lon) {
```

- **async**: Indica que es una función asíncrona, lo que permite usar await dentro de ella.
- **Parámetros:** Recibe latitud (lat) y longitud (lon) para localizar geográficamente la consulta meteorológica.

2. **Construcción de la URL de la API**

```
const apiURL = `URL de la API junto con los parámetros`;
```

- **Dinamismo:** Aquí se construiría la URL real de la API de OpenWeatherMap con los parámetros necesarios, como:
    - **lat:** Latitud de la localización
    - **lon:** Longitud de la localización
    - **appid:** Clave de API del usuario
    - **units=metric:** Para obtener temperaturas en Celsius
    - **lang=es:** Para recibir descripciones en español

3. **Bloque try (Intento de conexión)**

```
try {
    const response = await fetch(apiURL);
```

- **await fetch():** Pausa la ejecución hasta recibir respuesta de la API.
- **fetch():** Método nativo de JavaScript para hacer peticiones HTTP.

4. **Procesamiento de la respuesta**

```
return await response.json();
```

- **response.json():** Convierte la respuesta de la API (en formato JSON) a un objeto JavaScript utilizable.
- **Segundo await:** Necesario porque la conversión a JSON también es una operación asíncrona.

5. **Bloque catch (Gestión de errores)**

```
} catch (error) {
    console.error("Error al obtener datos del tiempo:", error);
}
```

- Captura cualquier error durante el proceso (red, URL incorrecta, datos inválidos...).
- **console.error:** Mostraría el error en la consola para depuración.

Entrega: Una vez completado todo subiremos la carpeta **Ejercicio60** al espacio para entrega de ejercicios que tenemos en el **OneDrive** de **Microsoft Teams**.