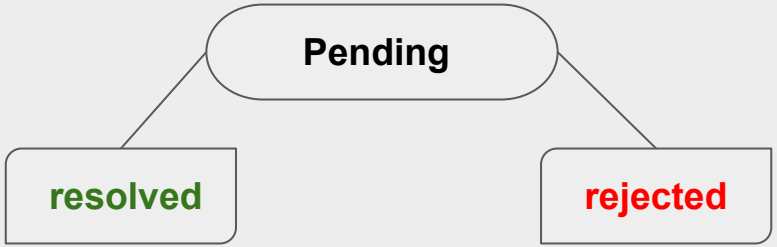

ICARO
Nuestro compromiso es con el futuro.

Introducción al Front End

Clase 4

⌛Asincronía en JavaScript

La programación **asíncrona** en **JavaScript** será moneda corriente mientras estés desarrollando tus programas. La mayoría de las bibliotecas y librerías de **JavaScript** son asíncronas.
El enfoque tradicional de la programación **asíncrona** era utilizar callbacks.
Pero ya sabemos que hay muchos problemas con los **callbacks**.
En el **JavaScript** moderno, las promesas sustituyen a los **callbacks** para la programación **asíncrona**.
Un código que devuelve una **promesa** es no bloqueante y eventualmente devolverá un resultado o un error a la persona que llama a través de la **promesa**.



⌛Asincronía en JavaScript

Si una **función asíncrona** no ha completado su tarea, la **promesa** que ha devuelto estará en estado **pendiente**. Cuando la función asíncrona complete con éxito, pondrá la promesa en el estado resuelto. En este momento, la promesa genera o emite el resultado pasado a través de ella - nos referimos a esto como la promesa resuelta. Si una **función asíncrona** termina con un error, entonces pone la **promesa** en el estado rechazado. En este momento, la **promesa** genera o emite el error pasado a través de ella - nos referimos a esto como la promesa rechaza.
Una **promesa** puede ser creada en uno de estos tres estados; sin embargo, **resolved** y **rejected** son estados finales.

Veamos el ejemplo en el código

⌛Asincronía en JavaScript

Como ya sabemos se pueden extender las promesas para usar los nuevos **async** y **await**.
La función **async** y **await** se introdujo para mantener la estructura del código idéntica entre el código **síncrono** y el **asíncrono**. Esto no afecta a la forma en que escribimos las funciones asíncronas, pero cambia en gran medida la forma en que las usamos.
Hay dos reglas para utilizar esta función:

- Para poder utilizar una función **asíncrona** como si fuera una función **síncrona** colocar la palabra clave **async** delante de la función **asíncrona** que devuelve la promesa.
- Para llamar a la función asíncrona como si fuera una función **síncrona** coloque la palabra clave **await** justo delante de la llamada. La palabra clave **await** puede ser utilizarse sólo dentro de las funciones marcadas como asíncronas.

Veamos el ejemplo en el código

Fetch

¿Qué es?

*La API **Fetch** proporciona una interfaz **JavaScript** para acceder y manipular partes del canal **HTTP**, tales como **peticiones** y **respuestas**. También provee un método global **fetch()** (en-US) que proporciona una forma fácil y lógica de obtener recursos de forma **asíncrona** por la red.*

(https://developer.mozilla.org)

Una petición básica de fetch es muy simple de realizar:

```
fetch('http://example.com/movies.json')
  .then(response => response.json())
  .then(data => console.log(data));
```

Pero mejor veamos un ejemplo práctico en el código👽

Vamos a usar la api de Pokemon para poder traernos data y mostrarla en el html

https://pokeapi.co/ 👈

Object Location

Este **objeto**, el cual puede utilizarse sin el prefijo **window**, nos sirve tanto para obtener la **url** o parte de ella, de la web donde nos encontramos como para **redireccionarnos** hacia otra página.
¿Cómo usarlo?

Obtener la url de la página
Una de sus funciones más importantes puede ser para recoger la url de una web, esto ya lo vimos con php, y puede resultar muy interesante, de hecho en javascript es mucho más sencillo si cabe, ya que solo con la siguiente instrucción obtenemos la url de la página:
window.location.href
La cual podemos imprimir, guardar en una variable, introducir en un array, en una condición if o cualquier cosa que podamos necesitar

Obtener el nombre del hosting
Puede ser que no necesitemos toda la url de la página, ya que puede que estemos por ejemplo ejecutando el código en un resultado de búsqueda o por cualquier otra razón:
window.location.hostname
De esta forma solo recibiriamos un resultado del hosting, que sería para esta página: mialtoweb.es

Object Location

Obtener el protocolo
Puede resultar interesante también saber si la web se encuentra bajo un protocolo http o https, y location continúa haciendo su magia con otra propiedad…

window.location.protocol

Redireccionando con location
Utilizando location en javascript es tan sencillo como:
window.location="dirección donde queremos ir"

Naturalmente debemos de poner una url válida o recibiremos un error.

Veamos un ejemplo práctico en el código👽

Local Storage

El **local storage** es un lugar donde podemos guardar información, en lugar de guardarla en las cookies de **JavaScript**, pero de esta manera podemos grabar más información y es más simple grabar y acceder a dichos datos almacenados.

Guardar en el Local Storage - Sintaxis:

Hay que tener en cuenta que solo almacena *strings*. Por lo tanto si queremos guardar un objeto debemos pasarlo antes por JSON.stringify(elObjeto) y almacenarlo como **localStorage.setItem("nombre",propiedad) o localStorage.setItem("nombre",JSON.stringify(elObjeto))**

Obtener del Local Storage:

Para obtenerlo podemos acceder con **localStorage.getItem("nombre",propiedad),** o si es un objeto **JSON.parse(localStorage.getItem("nombre"))**

Eliminar elemento:

localStorage.removeItem("nombre");

Session Storage

La propiedad **sessionStorage** es similar a **localStorage**, la única diferencia es que la información almacenada en **localStorage** no posee tiempo de expiración, por el contrario la información almacenada en **sessionStorage** es *eliminada al finalizar la sesion de la página.*
La sesión de la página perdura mientras el navegador se encuentra abierto, y se mantiene por sobre las recargas y reaperturas de la página. Abrir una página en una nueva pestaña o ventana iniciará una nueva sesión, lo que difiere en la forma en que trabajan las cookies de sesión.

Guardar en el Session Storage tiene la misma sintaxis:

```
// Almacena la información en sessionStorage
sessionStorage.setItem('key', 'value');

// Obtiene la información almacenada desde sessionStorage
let data = sessionStorage.getItem('key');
```

Veamos un ejemplo práctico en el código👽

Bibliografía consultada y links para seguir investigando

- Gascón Gonzalez U. (2017) ***JavaScript, ¡Inspírate!-*** Leanpub
- Venkat Subramaniam, ***Rediscovering JavaScript -****Jacquelyn Carter*
- mialtoweb.es/el-objeto-location-en-javascript/
- https://developer.mozilla.org/es/docs/Web/API/Fetch_API/Using_Fetch
- https://developer.mozilla.org/es/docs/Web/API/Window/localStorage

Muchas gracias!

ICARO Asociación Civil
CUIT 30716564815
info@icaro.org.ar
www.icaro.org.ar