JavaScript

# Función Anónima

Es una función que no tiene nombre y se puede guardar dentro de una variable.

# Función de Callback

Es una función anonima que no tiene ningun nombre y que se le pasa como parametro a otra función, es decir, al final es le pasamos como parametro una función a otra función y esa propia función dentro de ella, ejecuta la función anonima pasada.

# Función de Flecha

Es una forma de definir funciones de callback de forma más limpia y clara, simplemente utilizando una flecha => en lugar de function. Si lleva un solo parametro no es necesario colocarlos entre parentesis.

# Fetch

Es un método que nos permite hacer peticiones asíncronas AJAX de una manera más simple, que como se hacían en JavaScript Clásico.

Una petición AJAX es simplemente una llamada a un servicio REST, API REST (backend) y nos devuelve un resultado y nosotros podemos hacer con ese resultado lo que queramos.

# JQUERY UI

Si solamente queremos utilizar los métodos más básicos, yo puedo incluir solamente el javascript (jqueri-ui.js), me valdría. Se tiene que agregar justo después de cargar el javascript de JQuery por que JQuery UI funciona sobre JQuery.

TypeScript

# Modificadores de Acceso

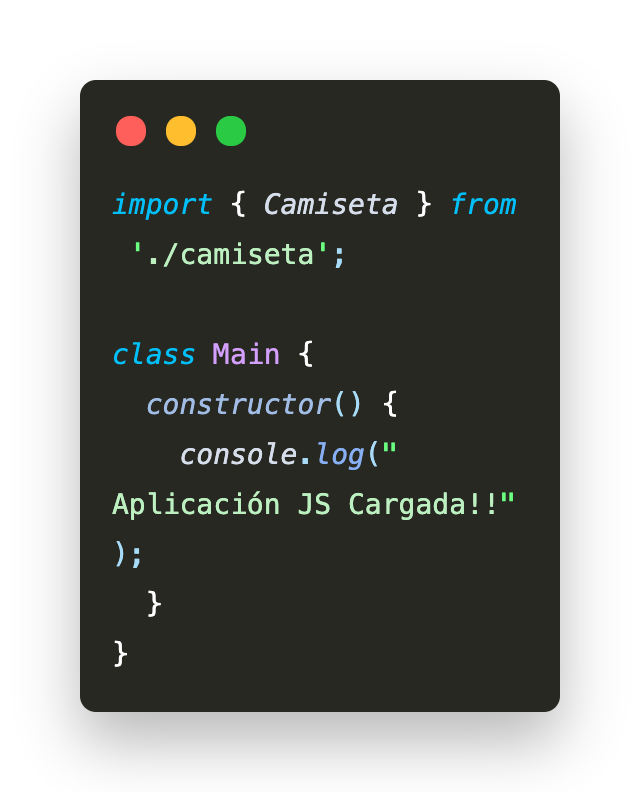
* *publicpodemos acceder a las propiedades y metodos desde cualquier lugar, desde clases que la heredan y desde otras clases*
* *protected se puede acceder a las propiedades y metodos desde la clase que lo define o desde cualquier otra clase que la herede*
* *private los atributos y metodos solamente son accesibles desde la clase que las define*

# Metodo Constructor

Metodo especial dentro de una clase y se suele usar para darle un valor inicial a los atributos o propiedades del objeto al crearlo. Es el primer metodo que se ejecuta al crear el objeto y se llama automaticamente nada mas crearlo. Este metodo puede recibir parametros como cualquier otro metodo. Y para paserselo tenemos que pasar los parametros al objeto en si cuando lo instanciamos. Nunca devuelve ningun dato.

# Exports e Imports





# Interfaces

Es un contrato en el que definimos que métodos y propiedades obligatorios va a tener una clase. Podemos definir interfaces y podemos aplicárselas a las clases que queramos y esas clases deben de cumplir ese contrato de la interfaz y tienen que existir los métodos y propiedades que existan en esa interfaz. Se suelen utilizar cuando necesitamos que un software sea muy robusto y que halla mucha rigurosidad a la hora de definir las clases.



# Herencia

Mecanismo muy básico en el cual una clase hija puede heredar las características de una clase padre.

# Decoradores

Un decorador es un patrón de diseño que nos permite mediante unos metadados que nosotros le definimos a una clase, hacer una copia de esa misma clase y modificarla para que haga una cosa u otra en función de los parámetros que se le pasen.