INTRO A JAVASCRIPT 2

* Array functions:
  + forma mas compacta de definir una función en JS
  + No es 100% análoga a definir una función con la palabra clave function()
* Falsey statemens : undefined, null, NaN, ) son considerados falsey y se evaluand a false
* Closures
  + Excecution context:
    - el ambiente en donde se ejecuta el codigo de JS
    - Ambiente: varibales, funciones, this, objetos
    - Dos tipos
      * Global excecution context: se crea cuando el motor comienza a ejecutr el codigo
      * Funcrion excecution conte: se crea cuando se unvoca una funcion
    - Existen dos fases al ejecutar un excecution context: fase de creacion y fase de ejecution
  + Scope:
    - es el execution context actual donde el codigo se esta ejecutando.
    - Existen dos tipos de scopes en JS, el global y el local
  + CLOSURES: al definir una funcion se crear un scope local que existe solo dentro del cuerpo de esa funcion
* Se pueden crear clases, se puede tener programacion orientada a objetos
* Las funciones en JS se deonminan firtst class
* This binding
  + Existen 4 o 5 modos de como se produce el binding del objeto this en JS
    - Default: cuando una funcion se usa sin tener un “.” Abtes de ka invocacion
    - Implicit: cuando la funcion se usa con un “.”antes de la invocacion 🡪 var p = Rectantulo(12,10) , p.calcArea()
    - Explicit: cuando la funcion se usa con un “bind()” “apply()”o “call()” 🡪 no muy usado
    - New: cuando la funcion se invoca con un new
    - Arrow funcions: TD como funciona???
* OPERACIONES ASINCRONICAS:
  + Callbacks: cuando una funcion acepta como parametro a otra funcion. La funcion que acepta a otra como uno de sus argumentos se llama “high order function”
  + Se pueden enganchar callbacks
  + los callbacks son bastante engorrosos de manejar – NO ESCALA, SE VUELVE INMANEJABLE, es dificil de debaguear
  + PROMISES: son objetos que retornan otro objeto en el futuro. Son mas claras para expresar el flujo del codigo que los callbacks. Tiene 3 estados Pending, Rejected, Fulfilled. Ejemplo: lo q te dan los restaurantes cuando pedis comida y te dan el cosito q hace ruido cuando esta tu comida.
  + Cuando la promise falla te devuelve Rejected
  + Cando no falla y funciona toda bien queda enl estado de fulfilled
  + Para definirla es bastante sencilla : const promise = new Promise(function(resolve,reject){
  + //promise description
  + })
  + Then es lo q se ejecuta cuando la promesa se resuelve en caso positivo
  + Catch es lo q se ejecuta cuando la promesa se resuelve en caso negativo
  + Nos permite de manejar un porquito mas feliz de manejar los callbacks, no es lo mejor pero es muchisimo mejor q los calbacks
  + Then te retorna otra promise, se pueden enganchar varios thens
* ASYNC&AWAIT
  + Son unas palabras claves para manejar codigo asincronico de forma sincronica
  + Las funciones async siempre retornarn una promise
  + Asi como en las promises esperamos un resultado con el then, cuando tenemos una funcion asyn lo hacemos con el await
  + Es la forma mas nueva de manejar la asincronia
  + El async /await Maquillar un poco el manejo de las promises
  + Mejora bastante tu codigo
  + Se lee sincronicamente, pero es asincronica
  + Es la forma mas clara de manejar la asincronia en javascript