JavaScript

* Es un lenguaje de programación dinámico
* Las limitantes de JavaScript es que no permite la lectura ni escritura de archivos del servidor, porque JavaScript trabaja del lado del cliente.
* No puede ser utilizado para aplicaciones de red por si solo.
* JavaScript no tiene capacidad multihilos o múltiples procesos simultáneos
* JavaScript es un lenguaje interpretado pero en contexto, esto quiere decir que el navegador lee el archivo pero no lo ejecuta, cuando ve una definición de variable se reserva un espacio en memoria y los inicializa con undefined los tipos de datos primitivos.
* Undefined es diferente a null.
* JavaScript no es asíncrono, por eso no puede ejecutar varias funciones en simultaneo. Primero termina de ejecutar una función y luego inicia con la siguiente.
* JavaScript maneja una escritura dinámica, esto quiere decir que las variables no estan amarradas a un tipo de dato en particular. Las variables pueden tomar cualquier valor. Recordar que hay cinco tipo de datos primitivos: numerico, string, booleanos, undefined, null.
* Los objetos se declaran de la siguiente manera: Var nom\_objeto = { }; Se puede identificar un objeto porque solo utilizan llaves.

Un objeto puede tener tipos primitivos como objetos, un objeto puede tener muchos objetos y tipos primitivos, su capacidad será la memoria RAM. Dentro de los objetos todo esta dentro de una nomenclatura de dos puntos.

Ejemplo.

Var obj = {

Numero: 10,

texto: “nuevo texto”

};

Un objeto este compuesto de propiedades, las propiedades terminan en coma (,) excepto la ultima propiedad que no lleva la coma.

Los nombres de variables inician en minúscula y utilizan la notación camelCase.

Un objeto puede tener varias variables, pero denotro de este no es necesario colocar var, no es necesario reservar un espacio de memoria porque este ya se reservo con el objeto.

# Por valor y por referencia

Las variables primitivas siempre se pasan por valor, cada variable va a tener un espacio de memoria diferente

Html y enter completa el código inicial de html

Control + shift + arriba o abajo mueve una línea de código

# Functions

En Javascript solo hay un tipo de instrucción y es la función pero no hay procedimientos.

Todas las funciones regresan un valor, desde un undefined, la definición de la función o el valor establecido.

Se deben escribir las variables a utilizar por las funciones antes de que estas sean definidas.

El llamado a la función puede estar antes de definirse la función o después de esta y no va a generar error o undefined.

Una buena practica es colocar todas las funciones al inicio del codigo

Una variable anónima es cuando no esta explicita en el código, sino que la declaro en el momento de llamar una función.

Una variable primitiva cuando se reserva en memoria tiene el valor undefined.

Defino una función con parámetros puedo hacer una estructura de decisión para que imprima un valor cuando no se envie argumento al momento de llamar a la función.

function persona(nombre, apellido) {

if ( apellido === undefined ){

apellido = "xxx";

}

console.log(nombre + " " + apellido);

}

persona("Fernando");

Otra manera de hacer esto con menos líneas de código y que devuelve el mismo resultado es el siguiente:

primeraFuncion();

function persona(nombre, apellido) {

apellido = apellido || "xxx"; //si apellido tiene un valor lo coloca y sino lo reemplaza el indefined por xxx

console.log(nombre + " " + apellido);

}

persona("Fernando");

Las funciones son objetos

Las funciones pueden retornar un tipo primitivo, un undefined, un null, un objeto o una función. La palabra reservada para hacer un returno es “return”.

A las funciones le podemos poner cualquier cosa que se le pueda poner a un objeto, como propiedades y métodos

# Métodos

Los métodos son funciones dentro del objeto

# Desestructuración de un objeto

({tarea, id, completado, creado})

Las referencias estáticas se hace con el nombre de la clase y la notación de punto

# Desplegar proyecto en GIT

Para desplegar el proyecto en git, se utilizan los siguientes comandos

**Git init** Comando para inicializar el repositorio

**git branch refs/remotes/origin/main** crea una ubicacion para guardar el proyecto

**git checkout -b main** Revisa que la ubicación se encuentre creada

**git add .** Este comando le dice a Git que se prepare para tomar una fotografía de como esta el proyecto en ese momento.

**git commit -m “nombre con que se guardara”** toma la fotografía del proyecto, una vez guardado se puede recuperar el proyecto tal cual como se guardo.

**git checkout --.** Reconstruye el proyecto

git log muestra si tenemos proyectos guardados

# desplegar a github

Luego de haber realizado las instrucciones de Git, se procede a desplegar al proyecto en github.

Pegar la instrucción de nuestro proyecto

**git remote add origin https://github.com/chinofer24/proyecto-Creador-Actividades.git**

**git push -u origin main** Sube los archivos al origin

en el proyecto, se crea un nuevo archivo con el nombre **README.md** que este contenido en la carpeta raíz del proyecto, dentro del archivo se coloca

# el nombre que queramos del proyecto

Buscamos en la opción view – Command palette – marckdown: open preview lo seleccionamos y lo colocamos a un lado de la pantalla

En el archivo README.md podemos escribir un par de notas e instrucciones que necesitemos para iniciar nuestro proyecto

Npm install

Npm run build

Para subir este archivo al repositorio hacemos nuevamente el git add. Git commit y el git push

Una vez cargado el archivo a github, vamos a la carpeta settings, buscamos la carpeta y seleccionamos guardar. Esperamos hasta que se guarde el archivo y aparezca que ya fue guardado

Para poder desplegar el proyecto desde github solo debemos hacer click en el enlace que nos creo para nuestro repositorio,

# Event.target

La propiedad target de la interfaz del event.currentTarget es una referencia al objeto en el cual se lanzo el evento.

# LocalStorage y SesionStorage

No se requiere instalar nada y sirve cuando la aplicación esta destinada a la web. Si se va a trabajar en node es decir en el backend, no va a funcionar el storage.

El SesionStorage se va a borra cuando se cierre por completo el navegador web.

El localStorage almacena la información y esta no tiene tiempo de expiración y esta información es visible por parte del usuario final.

El método localStorage.setItem(‘key’, ‘value’), recibe como parámetros dos strings, si tenemos un arreglo debemos usar la función JSON.stringify para pasar el arreglo a formato JSON que utiliza un formato string

Antes de trabara con el localStorage, se debe verificar si el objeto existe.

Cuando se utiliza la función JSON se pierden los métodos de la clase, para resolver este inconveniente es crear un elemento statico que cree un objeto temporal y pase la información al momento de instanciar la clase. El “map” va a barrer cada elemento que esta en el arreglo y retornar un nuevo arreglo con sus objetos mutados. Consultar mas información del map.

# Callback

Los callback son funciones que se envían como argumento

Un estándar en el funcionamiento de callbacks es que siempre el primer argumento que se va a enviar es un error. Ejemplo buscarHeroes( heroeId, ( err, héroe )=>{ }