Ejercicio en clase

1}

1- el código de js se ejecuta directamente sin necesidad de antes ser compilado, en lugar de eso el navegador interpreta y ejecuta el código línea por linea y por otra parte el tipado dinámico significa que no es necesario declarar el tipo de la variable cuando la defines sabiendo que esto puede cambiar durante la ejecución del programa

* Los tipos de variable se determinan en tiempo de ejecución, no en tiempo de compilación
* Una misma variable piede contener un numero, luego cadena, luego un objeto etc

2- Las diferencias entre var, let y const radican en su ámbito (de función vs. de bloque), reasignación y re-declaración, y el concepto de elevación.

var tiene alcance de función, permite re-declarar y reasignar, y se eleva con un valor inicial de undefined.

let tiene alcance de bloque, no se puede re-declarar pero sí reasignar, y se eleva sin valor inicial.

const también tiene alcance de bloque, no se puede re-declarar ni reasignar, y se eleva sin valor inicial, requiriendo inicialización al momento de la declaración

3- operadores aritméticos

+ suma % modulo (resto

- resta \*\* exponente

\* multiplicación ++ incremento

/ división -- decremento

* Operadores de comparación

== igualdad ( sin tipo estricto) > mayor que

¡= desigualdad < menor que

=== igualdad estricta (valor y tipo) >=mayor o igual que

¡== desigualdad estrica <= menor o igual que

4) tipos de datos primitivos: son datos simples e inmutables que se almacenan directamente en la memoria

string – texto

number – números enteros o decimales

boolean – valor lógico

undefined – valor de una variable no inicializada

null – ausencia intencional de valor

symbol – identificadores únicos

Objetos

Object – colección de pares clave – valor

Array - lista ordenada de elementos

Function – bloques de código ejecutable

Date – fecha y hora

RegExp – expresiones regulares

5)

PARAMETROS Y ARGUMENTOS : Los **parámetros** son variables que se declaran en la definición de la función.  
Los **argumentos** son los valores que se pasan cuando se llama la función.

function saludar(nombre) {

console.log("Hola, " + nombre);

}

saludar("Ana"); // Argumento: "Ana"

VALOR DE RETORNO : Una función puede devolver un valor con la palabra clave return

function restar(a, b) {

return a - b;

}

let resultado = restar(10, 4); // 6

console.log(resultado);

SCOPE : define donde una variable esta disponible

Tipos de scope

Global – fuera de cualquier función, disponible en todo el código

Funcional – dentro de una función, solo es accesible ahí

De bloque – dentro de [] como en if , for ,let , y const}

let x = 10; // Global

function prueba() {

let y = 20; // Local (funcional)

console.log(x); // Accede a x

console.log(y); // Accede a y

}

prueba();

console.log(y); // ❌ Error: y no está definido

6) ARRAY: un array es una estructura que permite almacenar varios valores en una sola variable

Let frutas = [ “manzana”, “banana”, “pera”]; aquí el array frutas tiene 3 elementos (indicees 0,1,2)

PROPIEDAD .LENGTH: sirve para saber cuantos elementos tiene el array

Console.log (frutas.length); //3

PUSH: agrega uno o mas elementos al final del array

Frutas.push(“kiwi”);

Consolo.log (frutas) ; //{ “manzana”, “banana”, “pera”, “kiwi”}

POP: elimina el ultimo elemento del array y lo devuelve

Let ultima = frutas.pop();

Console.log (ultima); //”kiwi”

Console.log (frutas); // manzana, banana , pera

UNSHIFT: agrega elementos al inicio del array

Frutas. Unshift (“fresa”;

Console.log (frutas); // fresa, manzana, banana, pera

7) OBJETOS LITERALES Y LA PALABRA CLAVE THIS

Objetos literales: es una estructura que permite agrupar datos (propiedades) y funciones (métodos) dentro de una misma entidad

Let persona = {

Nombre: ana,

Edad: 25,

Saludar: function(){

Console.log(“hola,soy” + this.nombre); } };