Sebastián Hernández C23770

Preguntas

1¿Cuáles son los data types que soporta javascript ?

string (cadenas de texto)
number (números, enteros y decimales)
bigint (enteros muy grandes)
boolean (true/false)
undefined (valor no asignado)
null (ausencia intencional de valor)
symbol (valores únicos e inmutables)
No primitivos (objetos):
Object, Array, Function, etc.

2. ¿Cómo se puede crear un objeto en javascript? De un ejemplo

Entre llaves {} usando lit o const

```
lit persona = {
        nombre = “sebas”
        edad = 21
        id = “C23770”
}
```

3. ¿Cuáles son los alcances (scope) de las variables en javascript?

Global: accesible en todo el programa.
De función: solo dentro de una función.
De bloque: solo dentro de {} (usando let o const).

4. ¿Cuál es la diferencia entre undefined y null?

undefined: una variable declarada pero sin valor asignado.
null: representa la ausencia intencional de valor.

5. ¿Qué es el DOM?

El DOM (Document Object Model) es una representación estructurada del documento HTML como un árbol de nodos que JavaScript puede manipular (crear, modificar o eliminar elementos).

6. Usando Javascript se puede acceder a diferentes elementos del DOM. ¿Qué hacen, que retorna y para qué funcionan las funciones getElement y querySelector? Cree un ejemplo

getElement y querySelector
getElementById("id"): selecciona un elemento por su id.
querySelector("selector"): selecciona el primer elemento que coincida con un selector CSS.
Ejemplo:

```
let titulo = document.getElementById("titulo");
let parrafo = document.querySelector(".texto");
```

7. Investigue cómo se pueden crear nuevos elementos en el DOM usando Javascript. De un ejemplo

```
let nuevo = document.createElement("p");
nuevo.textContent = "Hola desde JS";
document.body.appendChild(nuevo);
```

8. ¿Cuál es el propósito del operador this?

Referenciar al objeto o estructura actual donde se ejecuta el codigo

9. ¿Qué es un promise en Javascript?De un ejemplo

Las promesas en JavaScript no solo representan el resultado de una operación asincrónica, sino que también proporcionan métodos que facilitan el manejo y la manipulación de los datos una vez que la promesa se resuelve.

Una promesa es un objeto que representa un valor que puede que esté disponible «ahora», en un «futuro» o que «nunca» lo esté. Como no se sabe cuándo va a estar disponible, todas las operaciones dependientes de ese valor, tendrán que posponerse en el tiempo.

10.¿Qué es Fetch en Javascript? De un ejemplo

fetch() permite hacer peticiones HTTP de forma sencilla.

```
fetch("https://hola_esto_es_un_ejemplo_aleatorio.com")
  .then(res => res.json())
  .then(data => console.log(data));
```

11. ¿Qué es Asynch/Await en Javascript ? De un ejemplo

Se usan para manejar las promesas con mayor claridad

```
async function cargar() {
  let res = await fetch("https://jsonplaceholder.typicode.com/posts/1");
  let data = await res.json();
  console.log(data);
}
cargar();
```

12.¿Qué es un Callback? De un ejemplo

En JavaScript, un callback es una función que se pasa como argumento a otra función, y que luego es invocada dentro de la función receptora.

```
function saludar(nombre, callback) {
  callback(`Hola, ${nombre}`);
}
saludar("Sebastián", msg => console.log(msg));
```

13. ¿Qué es Clousure?

Es una función que recuerda el entorno léxico en el que fue creada, incluyendo las variables a las que tenía acceso en ese momento. Esto significa que una función interna puede acceder a las variables de su función contenedora incluso después de que la función contenedora haya terminado de ejecutarse.

14.¿Cómo se puede crear un cookie usando Javascript?

`document.cookie = "usuario=Sebastian; expires=Fri, 31 Dec 2025 23:59:59 UTC; path=/";`

15.¿Cuál es la diferencia entre var, let y const?

var:

Tiene alcance de función. Si se declara dentro de una función, solo es accesible dentro de esa función. Si se declara fuera de cualquier función, es global.
Puede ser redeclarada y reasignada dentro de su ámbito.
Sufre de "hoisting", lo que significa que la declaración se mueve al principio del ámbito, pero no su inicialización.

let:

Tiene alcance de bloque (entre llaves {}). Solo es accesible dentro del bloque donde se declara.
No puede ser redeclarada dentro del mismo ámbito.
Puede ser reasignada.

const:

También tiene alcance de bloque.
No puede ser redeclarada ni reasignada después de la inicialización.
Es ideal para declarar valores constantes que no deben cambiar durante la ejecución del programa.

Var ya no se suele usar debido a que puede dar problemas de scoping