1. ¿Cuáles son los data types que soporta JavaScript?

JavaScript soporta 2 tipos de de data types: Primitivos y Objetos

Primitivos: Number, String, Boolean, Null, Undefined, BigInt, Symbol

Obejetos: Arrays, Object, Function

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

```
const Carro = {
    nombre: Rachit,
    edad: 21,
    pasarCurso: () =>{
        console.log("Pasé con 9")
    }
}
```

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

En javascript existen 3 tipos de scope: Global, Function, Block.

Global: Son variables o constantes disponibles en todo el programa.

Function: Son variables o constantes que solo están disponibles en dentro de las llaves de una función en al que fueron declarados.

Block: Son variables o constantes que solo son accesibles dentro del bloque {} donde fueron declaradas (como en if, for, while, etc...).

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

Las variables undefined son las variables que el programador no ha definido y son establecidas por default

let a;

console.log(a) -> retorna undefined

null son las variables que el prograamdor asigna intencionalmente, o sea es la ausencia intencional de valor.

Además

let a;

console.log(typeof(a)) -> object

5. ¿Qué es el DOM?

(Document Object Model) Es como un árbol jerárquico de nodos, donde cada etiqueta HTML se convierte en un objeto que puede ser accedido y manipulado con JavaScript.

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: Devuelve el elemento según sea seleccionado puede ser por Id, clase, nombre, etc... Retorna un único elemento o null en caso de no encontralo.

querySelector: Devuelve el primer elemento que coincida con un selector CSS. Retorna un único elemento o null si no lo encuentra.

```
<p id="element ">Hola con ElementId</p>

<p class="query ">Hola con QuerySelector</p>
<script>
 const parrafoQuery = document.querySelector("query");
 console.log(parrafoQuery.textContent);  -> Hola con QuerySelector

 const parrafoElement = document.getElementById("element");

 console.log(parrafoElement.textContent); -> Hola con ElementId


</script>
```

7. Investigue cómo se pueden crear nuevos elementos en el DOM usando Javascript. De un ejemplo
   Se puede usar document.createElement para crear un nuevo nodo HTML, luego añadirlo al DOM con appendChild o append

```
const nuevoElemento = document.createElement("p”
 nuevoParrafo.textContent = "nmeuvo elemento";

 const padre = document.getElementById(“padre");
 contenedor.appendChild(nuevoElemento);
```

8. ¿Cuál es el propósito del operador this?
   Sirve para hacer referencia al objeto en el que está la función, Depende de cada objeto el valor que tenga
9. ¿Qué es un promise en Javascript? De un ejemplo
   Sirve para ejecutar operaciones asincrónicas y tiene 3 estados: pending, fulfilled, rejected. Se pueden manecar con .then, .catch, .finally para manejar el resultado de acuerdo a la situación}

```
const promesa = new Promise((resolve, reject) => {
 let exito = true;
 setTimeout(() => {
  if (exito) {
   resolve("Éxito!");
  } else {
   reject("Falló.");
  }
 }, 1000);
});

promesa
 .then(resultado => console.log(resultado))
 .catch(error => console.error(error));
```

10. ¿Qué es Fetch en Javascript? De un ejemplo
    fetch() es una función nativa de JavaScript para realizar peticiones HTTP (GET, POST, etc.) también funciona con GraphQL y trabajar con datos de servidores o APIs. Devuelve una Promise.
    Fetch("https://pokeapi.co/api/v2/pokemon/ditto").then(respuesta => respuesta.json())
     .then(datos => console.log(datos))
11. ¿Qué es Asynch/Await en Javascript? De un ejemplo
    Es una forma más reciente y más legible de trabajar con promesas. O sea que también permiten ejecutar operaciones asincrónicas

```
async function obtenerDatos() {
try {
  const respuesta = await fetch("https://pokeapi.co/api/v2/pokemon/ditto");
    const datos = await respuesta.json();
    console.log(datos);
  } catch (error) {
      console.error('Hubo un error:', error);
  }
}
```

12. ¿Qué es un Callback? De un ejemplo
    Es una función que se pasa como argumento a otra función y que se ejecuta después de que la otra termina su tarea. U

```
function saludar(nombre, callback) {
 console.log(“Hola, “ + nombre);
 callback();
}

function despedida() {
 console.log(“Adiós”);
}
```

13. ¿Qué es Clousure?Usando el video que brindó la profe “Closures are nothing but FUNCTIONS WITH PRESERVED DATA”

14. ¿Cómo se puede crear un cookie usando Javascript?
    Se pueden crear utilziando el document.cookie

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

Var tiene ámbito de función

```
function ejemploVar() {
    if (true) {
        var x = 10;
    }
    console.log(x);
}
```

Mientras que let y const tienen ámbito de bloque. Por otro lado let y var se pueden reasignar mientras que const por ser una constante no se puede asignar otro valor. Pero si se pueden modificar propiedades de objetos declarados con const