1. ¿Cuáles son los data types que soporta javascript ?
    a. String
    b. Number
    c. Bigint
    d. Boolean
    e. Undefined
    f. Null
    g. Symbol
    h. Object
        i. Arrays
        ii. Dates
        iii. Maps
        iv. Sets
        v. Intarrays
        vi. Floatarrays
        vii. Promises
2. ¿Cómo se puede crear un objeto en javascript? De un ejemplo

    Hay varias formas de crear objetos, en mi entendimiento casi todo en JS es un objeto, lo que permite hacer cosas bien raras comparadas a otros lenguajes.

- Literal

```
const persona = {
  nombre: "Erick",
  edad: 30,
  saludar: function() {
    return "Hola, soy " + this.nombre;
  }
};
```

- Constructor Object

```
const coche = new Object();
coche.marca = "Honda";
coche.modelo = "CR-V";
```

- Constructor

```
function Animal(especie, nombre) {
  this.especie = especie;
  this.nombre = nombre;
}
const perro = new Animal("perro", "Kira");
```

- Clases

```
class Producto {
```

```
  constructor(nombre, precio) {
    this.nombre = nombre;
    this.precio = precio;
  }
}
const control = new Producto("Control Xbox", 60);
```

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

- Block scope: dentro de {}.
- Function scope: locales a función.
- Global scope

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

Undefined indica que una variable no ha sido inicializada, mientras que null es que intencionalmente tiene un valor vacío.

5. ¿Qué es el DOM?

Document Object Model, permite el acceso al árbol HTML desde JS, con esto se puede acceder y manipular el contenido, estilo y estructura de la página.

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

getElementById: Devuelve un elemento único por su ID, retorna un único elemento o null
```
const titulo = document.getElementById("titulo");
```

getElementsByClassName: Devuelve una colección de elementos con la misma clase, retorna HTMLCollection.
```
const parrafos = document.getElementsByClassName("parrafo");
```

getElementsByTagName: Devuelve elementos por nombre de etiqueta, retorna HTMLCollection.
```
const botones = document.getElementsByTagName("button");
```

querySelector: Selecciona el primer elemento que coincida con el selector CSS, retorna un único elemento o null.
```
const primerBoton = document.querySelector(".boton");
```

querySelectorAll: Selecciona todos los elementos que coincidan, retorna NodeList.
```
const todosLosParrafos = document.querySelectorAll("p");
```

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

Se utiliza la instrucción .createElement(“<name>”), este nuevo elemento puede agregarse al documento o agregarse a otro elemento ya existente.

```
const btn = document.createElement("button");

btn.innerHTML = "Hello Button";

document.body.appendChild(btn);
```

8. ¿Cuál es el propósito del operador this?

El operador this. hace referencia a un objeto, este tiende ser el objeto en uso, pero puede ser el objeto global. Si este se usa en un método de un objeto this. apunta a ese objeto, si se utiliza en una función o solo hace referencia al objeto global.

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

Un promise es código que contiene código productor y consumidor, cuando el código de producción tiene un resultado debe llamar myResolve o myReject si es exitoso o no. Una promesa puede ser pending: resultado undefined, fulfilled: resultado, rejected: error. Se realizan como funciones asíncronas.

```
const miPromesa = new Promise((resolve, reject) => {
  const exito = true;

  setTimeout(() => {
    if (exito) {
      resolve("Operación completada");
    } else {
      reject("Operación fallida");
    }
  }, 2000);
});

miPromesa
  .then(resultado => {
    console.log(resultado);
  })
  .catch(error => {
    console.error(error);
  });
```

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

Fetch es un API que realiza una función de async y await, fetch() inicia un proceso de buscar un recurso de un servidor y también retorna la Promesa de la respuesta.

```
fetch(file)
.then(x => x.text())
.then(y => myDisplay(y));
```

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

Async hace a una función retornar una Promesa y Awai hace la función esperar una promesa.

```
async function myFunction() {
  return "Hello";
}
myFunction().then(
  function(value) {myDisplayer(value);},
  function(error) {myDisplayer(error);}
);
```

12. ¿Qué es un Callback? De un ejemplo

Es una función a la cual se le pasa otra función como argumento. Estas se utilizan cuando se requiere un fino control de la secuencia de ejecución.

```
function myDisplayer(some) {
  document.getElementById("demo").innerHTML = some;
}
function myCalculator(num1, num2, myCallback) {
  let sum = num1 + num2;
  myCallback(sum);
}
myCalculator(5, 5, myDisplayer);
```

13. ¿Qué es Closure?

Un closure es una manera de hacer una variable privada dentro de una función. También es una función que tiene acceso al ámbito de su función externa aun cuando esta ya haya terminado de ejecutarse.

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

Utilizando la propiedad document.cookie.

```
function createCookie(nombre, valor, dias) {
  let expiracion = "";

  if (dias) {
    const fecha = new Date();
    fecha.setTime(fecha.getTime() + (dias * 24 * 60 * 60 * 1000));
    expiracion = "; expires=" + fecha.toUTCString();
  }

  document.cookie = nombre + "=" + encodeURIComponent(valor) + expiracion + "; path=/";
}

crearCookie("usuario", "Erick", 7);
```

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

- var: declara variables con scope global o de función, permite la re-declaración y redefinirse dentro del mismo scope. Tiene hoisting a diferencia de las otras.
- let: declara variables dentro del scope de bloque {...}, permite redefinirlas, pero no re-declararlas.
- const: declara “variables” que no se pueden redefinir luego de su primera definición, estas están en el scope de bloque. Necesitan ser definidas para poderse declarar.

Referencias

JavaScript Data Types
JavaScript Objects
JavaScript Scope
Undefined Vs Null in JavaScript | GeeksforGeeks
null and undefined | web.dev
JavaScript HTML DOM
JavaScript DOM Document
JavaScript this
JavaScript Promises
JavaScript Fetch API
Using the Fetch API - Web APIs | MDN
JavaScript Async
JavaScript Callbacks

JavaScript Function Closures
JavaScript Cookies
var, let y const: ¿Cuál es la diferencia?
Difference between var, let and const keywords in JavaScript | GeeksforGeeks