JavaScript Repaso

```
// Comment line
/*
Comment block
*/
alert("Hola soy un mensaje generado con alert!!!!");
// Variables
var nombre = "Uriel"; //String
var altura = 180;  //numero

//Concatenacion

var concatenacion = nombre + " " + altura;
//Escribir en el documento
document.writeln(concatenacion);
document.write("Texto mediante document.write");
```

Utilizando document.write no es posible controlar la ubicación donde aparecerá el texto dentro de la página web. La manera de controlar esto es agregando un espacio en el archivo HTML como se muestra a continuación

```
<!-- HTML -->
<div id="datos"></div>
        <script src="JS/main.js" type="text/javascript"></script>
```

y tomando en el archivo JS el elemento div para modificarlo desde ahí

```
// JS Code
var datos = document.getElementById("datos");
datos.innerHTML = concatenacion + ". Modificado mediante .innerHTML";
datos.innerHTML = `
```

```
    <h1>Soy la caja de texto modificada mediante comillas simples invertidas</h1>
    <h2>Mi nombre es: ${nombre}</h2>
    <h3>Mido: ${altura}</h3>
`;
if(altura>160)
    datos.innerHTML += '<h3>Soy una persona ALTA.</h3>';
```

Mediante las comillas simples invertidas se puede agregar un trozo de codigo HTML. De la última instrucción es instrucción es importante resaltar el operador +=, el cual permite conservar la información anterior en el div datos y agregar nueva información.

```
if(altura>160)
    datos.innerHTML += '<h3>Soy una persona ALTA.</h3>';
else
    datos.innerHTML += '<h3>Soy una persona BAJA.</h3>';

for(var i=2010; i<=2020;i++)
    datos.innerHTML += '<h2>Estamos en el anio '+ i +'</h2>';
```

Estructuras de control de flujo.

```
function muestraNombre(nombre, altura){
    var misDatos = `
            <hr />
            <h2>Datos agregados usando una función</h2>
            <h3>
                El nombre del usuario es: ${nombre}
            </h3>
            <h4>
                Mide: ${altura}
            </h4>
        `;
        if(altura>160)
```

```
            misDatos += '<h3>Soy una persona ALTA.</h3>';
        else
            misDatos += '<h3>Soy una persona BAJA.</h3>';

    return misDatos;
}
function imprimir(nombre, altura, areaDiv){
    areaDiv.innerHTML += muestraNombre(nombre, altura);
}

imprimir("Dylan", 67, document.getElementById('datos'));
imprimir("PAloma", 167, document.getElementById('datos'));
```

Funciones

```
var nombres = ["Victor", "Antonio", "Joaquin"];
var alturas = [198, 165, 87];

//       FORMA 1 DE RECORRER ARRAYS
document.write("<hr /><h3>Listado de nombre en un array recorrido con un ciclo for</h3>");
for(var i = 0; i < nombres.length; i++){
    document.write("Su nombre es: " + nombres[i] + "; ");
    document.write("Mide: " + alturas[i] + "; ");
    if(alturas[i] < 160)
        document.write(" Es una persona BAJA <br />");
    else
        document.write(" Es una persona ALTA <br />");
}

//       FORMA 2 DE RECORRER ARRAYS
document.write("<hr /><h3>Listado de nombre en un array usando funcion</h3>");
nombres.forEach(function(nombre){
    document.write("El nombre es: " + nombre + "<br />");
```

```
});


//       FORMA 3 DE RECORRER ARRAYS
document.write("<hr /><h3>Listado de nombre en un array usando funcion callback</h3>");
alturas.forEach((altura) =>{
    document.write("la altura es: " + altura + "; ");
    if(altura < 160)
        document.write(" Es una persona BAJA <br />");
    else
        document.write(" Es una persona ALTA <br />");
});
```

Objetos JSON

```
/ OBJETOS JSON
var coche = {
    modelo: 'Mercedes clase A',
    maxima: 500,
    antiguedad: 2020,
    mostrarDatos(){
        console.log(this.modelo, this.maxima, this.antiguedad, this.propiedad1);
    },
    propiedad1: "Cualquier valor"
};


document.write('<hr /><h1> Trabajando con objetos JSON</h1><hr />');
document.write(coche.modelo);
document.write("<p>A continuación se muestran datos en console. Abrir la consola para ver
los datos.</p>");
coche.mostrarDatos();
console.log(coche);
```

Tu puedes crear un objeto JSON segun tus necesidades, dentro de la funcion puedes hacer cualquier cosa.

PROMESAS: Son objetos que pueden devolver un resultado despues de un rato, o pueden no devolver nada; representan un valor que puede estar disponible ahora, en el futuro o nunca; se usan en:

- peticiones asincronas
- peticions AJAX
- Librerías

Con las promesas se puede:

- Capturar la respuesta positiva de un servicio o un trozo de codigo o de una peticion AJAX o cuando esta es rechazada.
- Controlar de mejor manera el codigo

La PROMESA no PROMETE que nos llegará un dato o un error, y podemos scar ese dato de manera asincrona cuando nos llege la respuesta.

```
// PROMESAS
var saludar = new Promise((resolve, reject) =>{


    setTimeout(() =>{
        let saludo = "Hola muy buenas a todos Chavales !!";
        saludo = false;
        if(saludo)
            resolve(saludo);
        else
            reject("No hay saludo disponible");
    }, 2000);
});


saludar.then(resultado => {
    alert(resultado);
}).catch(err =>{
    alert(err);
});
```

El metodo then no se ejecuta al cargar la pagina, sino que se espera para ejecturase de manera asincrona hasta que llega el resultado, ya sea que llegue el saludo o el reject o un error capturado en el catch.
La palabra let, permite que la variablesaludo este disponible de manera global.

CLASES

```
class Coche{
    constructor(modelo, velocidad, antiguedad){
        this.modelo = modelo;
        this.velocidad = velocidad;
        this.antiguedad = antiguedad;
    }
    aumentarVelocidad(){
        this.velocidad += 1;
    }
    reducirVelocidad(){
        this.velocidad -= 1;
    }

}

var coche1= new Coche("BMW", 200, 2017);
var coche2= new Coche("Audi", 100, 2018);
var coche3= new Coche("Mercedes", 200, 2013);
var coche4= new Coche("Ranault", 200, 2009);

coche1.aumentarVelocidad()
coche1.aumentarVelocidad()
coche1.aumentarVelocidad()
coche1.aumentarVelocidad()
coche1.aumentarVelocidad()
```

```
console.log(coche1)


var divClases = document.getElementById("divClases");
divClases.innerHTML = `
    <hr /><h1>Trabajando con clases</h1><hr />
    <p>
    Coche modelo: ` + coche2.modelo + `; velocidad: <strong>` + coche2.velocidad +
`</strong>
    ; y antigüedad: ` + coche2.antiguedad + `
    </p>
    `;
```

Herencia

```
class Autobus extends Coche{
    constructor(modelo, velocidad, antiguedad, altura){
        super(modelo, velocidad, antiguedad);
        this.altura = altura;
    }
    mostrarAltura(){
        return "La altura del bus es " + this.altura;
    }
}


var bus1= new Autobus("Pegasus", 200, 2017, 10);
```