TEMA 4. PROGRAMACIÓN FUNCIONES, ARRAYS Y OBJETOS DEFINIDOS POR EL USUARIO.

Objetivos

- Comprender las principales funciones predefinidas del Lenguaje Javascript.
- Poder crear funciones personalizadas para realizar tareas especificas que las funciones predefinidas no logran hacer.
- Comprender el objeto Array de Javascript y familiarizarse con sus propiedades y métodos.
- Crear objetos personalizados diferentes a los predefinidos en el lenguaje.
- Definir propiedades y métodos de los objetos personalizados.

Contenidos

4.1.- Funciones predefinidas del lenguaje.

Javascript dispone de funciones predefinidas en el lenguaje que realizan tareas recibiendo una serie de datos llamados argumentos o parámetros, aunque también existen funciones que no precisan parámetros para ejecutar las tareas que realizan.

Funciones predefinidas en Javascript:

Función predefinida	Descripción
`isNaN()`	Comprueba si el valor que pasamos como argumento es un de tipo numérico.
`Number()`	Convierte el objeto pasado como argumento en un número que represente el valor de dicho objeto.
`String()`	Convierte el objeto pasado como argumento en una cadena que represente el valor de dicho objeto.
`parseInt()`	Convierte la cadena que pasamos como argumento en un valor numérico de tipo entero.
`parseFloat()`	Convierte la cadena que pasamos como argumento en un valor numérico de tipo flotante.

Ejemplos:

```
<script type="text/javascript">
	// isNaN()
	var numero=prompt("Introduce numero:");
	if (isNaN(numero)){
		document.write("No se ha introducido un numero.<br>");
	}else {
		resultado = Math.pow(numero, 2);
		document.write("El cuadrado es "+resultado+"<br>");
	}

	// String()
	var numeroOnce = 25785;
	var numString = String(numeroOnce);
	document.write("El numero de digitos es "+numeroOnce.length+"<br>");
	document.write("El numero de digitos es "+numString.length+"<br>");
```

```
        // Number()
        var dni = "24893481";
        var numDni = Number(dni);
        var letras = ['T','R','W','A','G','M','Y','F','P','D','X',
                      'B','N','J','Z','S','Q','V','H','L','C','K','E','T'];
        var letradni = letras[numDni%23]
        document.write("El dni es "+dni+"-"+letradni+"<br>");

        // parseInt()
        var entero=prompt("Introduce numero entero:");
        document.write("El numero entero introducido es: "
              +parseInt(entero)+"<br>");

        // parseFloat()
        var decimal=prompt("Introduce numero decimal:");
        document.write("El numero decimal introducido es: "
              +parseFloat(decimal)+"<br>");
</script>
```

4.2.- Funciones del usuario.

En Javascript podemos crear nuestras propias funciones personalizadas para realizar tareas definidas por un grupo de instrucciones.

Para ejecutar este grupo de instrucciones en cualquier otra parte de la página simplemente debemos utilizar el nombre que le hayamos dado a la función. Las instrucciones que se encuentren dentro de una función sólo se ejecutan cuando invoquemos dicha función. Las variables declaradas dentro de la función sólo son accesibles dentro de la misma.

La mejor zona para definir las funciones es dentro del bloque <head></head> ya que el navegador carga todo lo que se encuentre entre estas etiquetas antes que el cuerpo de la página.

Podemos definir una función siguiendo la siguiente sintaxis:

```
function nombre_funcion([parametro1, parametro2,..]){
      sentencias;
      [return valor;]
}
```

Ejemplo:

```
      function calculaIva(precio, porcIva){
            var iva = precio * porcIva;
            return iva;
      }
```

Si deseamos llamar a la función lo haremos utilizando su nombre y entre paréntesis los argumentos que precisa para su ejecución separados por comas.

```
nombre_fucnion([parametro1, parametro2,...]);
```

Ejemplo:

```
ivaProducto = calculaIva(150, 0.21);
document.write("El IVA del producto es: "+ivaProducto+" €");
```

4.3.- Arrays.

Un array es un conjunto de elementos o valores accesibles mediante un índice que indica la posición dentro del array. El array es identificado para su acceso por un nombre y las posiciones de los elementos siempre empiezan en 0.

Podemos declarar un array utilizaremos la siguiente sintaxis:

```
var nombre_array = new Array();
```

Podemos inicializar un array mediante:

```
nombre_array[indice] = valor;
```

Podemos declarar e inicializar el array en una sola sentencia:

```
var nombre_array = new Array(valor1, valor2,…, valorn);
```

Para acceder a los elementos del array utilizaremos el nombre del array y entre corchetes el índice del elemento al que queremos acceder:

```
nombre_array[indice]
```

Para acceder a todos los elementos del array podemos hacer uso de la sentencia de bucle for…in:

```
for (indice in nombre_array) {
      sentencias;
}
```

Ejemplo: Acceso a un elemento de un array.

```
var producto = new Array(1, "Pan", 75.3, true);
document.write("El nombre del producto es: "+producto[1]+"<br>");
```

Ejemplo: Acceso a todos los elementos de un array.

```
var producto = new Array(1, "Pan", 75.3, true);
document.write("Los datos del producto son: <br>");
for (indice in producto){
      document.write(producto[indice]+"<br>");
}
```

Propiedades y métodos de los arrays.

Los arrays son objetos con propiedades y métodos.

Propiedades del objeto Array :

Propiedad	Descripción
length	Devuelve el número de elementos del array

Ejemplo:

```
var producto = new Array(1, "Pan", 75.3, true);
document.write("Tenemos "+producto.length+" datos del producto.<br>");
```

Existen métodos que permiten manipular y gestionar arrays. Los principales métodos del objeto Array son:

Método	Descripción	Sintaxis
push()	Añade nuevos elementos al *array* y devuelve la nueva longitud del *array*.	nombre_array.push(valor1, valor2, …)
concat()	Selecciona un *array* y lo concatena con otros elementos en un nuevo *array*.	nombre_array.concat(valor1, valor2, …)
join()	Concatena los elementos de un *array* en una sola cadena separada por un carácter opcional.	nombre_array.join([separador])
reverse()	Invierte el orden de los elementos de un *array*.	nombre_array.reverse()
unshift()	Añade nuevos elementos al inicio de un *array* y devuelve el número de elementos del nuevo *array* modificado.	nombre_array.unShift(valor1, valor2, …)
shift()	Elimina el primer elemento de un *array*.	nombre_array.shift()
pop()	Elimina el último elemento de un *array*.	nombre_array.pop()
slice()	Devuelve un nuevo *array* con un subconjunto de los elementos del *array* que ha usado el método.	nombre_array.slice(indice_inicio, [indice_final])
sort()	Ordena alfabéticamente los elementos de un *array*. Podemos definir una nueva función para ordenarlos con otro criterio.	nombre_array.sort([función])
splice()	Elimina, sustituye o añade elementos del *array* dependiendo de los argumentos del método.	nombre_array.splice(inicio, [numero_elem_a_borrar], [valor1, valor2, …])

Ejemplos:

```
<script type="text/javascript">
    var caracter="*";

    var marcas = new Array("Ford","Cadillac","Jeep","Rover");
    // array.length
    document.write("Numero de elementos: "+marcas.length+"<br>");
    document.write("Marcas: "+marcas+"<br>");
    document.write(caracter.repeat(20)+"<br>");

    // push(valor1, valor2,...)
    totalElementos = marcas.push("Fiat","Maserati");
    document.write("Numero de elementos: "+totalElementos+"<br>");
    document.write("Añadir al final: "+marcas+"<br>");
    document.write(caracter.repeat(20)+"<br>");
```

```
// unshift(valor1, valor2,...)
totalElementos = marcas.unshift("Chevrolet","Porsche");
document.write("Numero de elementos: "+totalElementos+"<br>");
document.write("Añadir al principio: "+marcas+"<br>");
document.write(caracter.repeat(20)+"<br>");

// shift()
primeraMarca = marcas.shift();
document.write("La primera marca es: "+primeraMarca+"<br>");
document.write("Eliminar primero: "+marcas+"<br>");
document.write(caracter.repeat(20)+"<br>");

// pop()
ultimaMarca = marcas.pop();
document.write("La ultima marca es: "+ultimaMarca+"<br>");
document.write("Elminar final: "+marcas+"<br>");
document.write(caracter.repeat(20)+"<br>");

// slice(inicio,tamaño)
document.write("Marcas: "+marcas+"<br>");

var aPartirCuarta = marcas.slice(4);
document.write("Cuarta hasta final: "+aPartirCuarta+"<br>");
document.write(caracter.repeat(20)+"<br>");

var primerasTres = marcas.slice(0,3);
document.write("Tres primeras: "+primerasTres+"<br>");
document.write(caracter.repeat(20)+"<br>");

var ultimasDos = marcas.slice(-2);
document.write("Ultimos dos: "+ultimasDos+"<br>");
document.write(caracter.repeat(20)+"<br>");

// splice(posicion, num_eliminar, nuevos_elementos)
document.write("Marcas: "+marcas+"<br>");

marcas.splice(2,0,"BMW","Ferrari");
document.write("Insertado posicion dos: "+marcas+"<br>");
document.write(caracter.repeat(20)+"<br>");

marcas.splice(0,2,"Subaru","Pontiac","Chrysler");
document.write("Modifica los 2  primeras: "+marcas+"<br>");
document.write(caracter.repeat(20)+"<br>");

marcas.splice(4,2);
document.write("Elimina dos desde la cuarta: "+marcas+"<br>");
document.write(caracter.repeat(20)+"<br>");

// concat(array,array,...)
var marcas_inglaterra = new Array("Lotus","Jaguar");
var marcas_corea_sur = new Array("Kia","Hyundai");
var marcas_espana = new Array("Seat");
var marcas_francia = new Array("Renault","Peugeot");
var marcas_japon = new Array("Nissan","Toyota");
var marcas_alemania = new Array("Opel","Audi","Mercedes");

var  todasLasMarcas  =  marcas.concat(marcas_inglaterra,  marcas_corea_sur,
        marcas_espana, marcas_francia, marcas_japon, marcas_alemania);
document.write("Todas las marcas concatenadas: "+todasLasMarcas+"<br>");
```

```
    document.write(caracter.repeat(20)+"<br>");

    //join(separador)
    var lasMarcas = todasLasMarcas.join(" - ");
    document.write("Todas las Marcas como String: "+lasMarcas+"<br>");
    document.write(caracter.repeat(20)+"<br>");

    // reverse()
    todasLasMarcas.reverse();
    document.write("Todas las marcas orden inverso: "+todasLasMarcas+"<br>");
    document.write(caracter.repeat(20)+"<br>");

    // sort()
    todasLasMarcas.sort();
    document.write("Todas las marcas ordenadas: "+todasLasMarcas+"<br>");
    document.write(caracter.repeat(20)+"<br>");
</script>
```

Arrays Multidimensional.

Un array multidimensional es un Array cuyos elementos son también Arrays, es decir un array de arrays.

Ejemplo:

```
  var vehiculos = new Array( new Array(100, "Ford Focus", 3, 18500),
                             new Array(230, "Seat Ibiza", 6, 13000),
                             new Array(345, "Peugeot 307", 2, 23800),
                             new Array(457, "Audi A3", 1, 32600));
```

```
Ejemplo:
  var coches = [[100, "Ford Focus", 3, 18500],
                [230, "Seat Ibiza", 6, 13000],
                [345, "Peugeot 307", 2, 23800],
                [457, "Audi A3", 1, 32600]];
```

Para acceder a los elementos del array multidimensional necesitaremos dos índices:

nombre_array[i][j]

Ejemplo:

document.write(vehiculos[3][1]);

Para recorrer el array multidimensional necesitamos dos for..in anidados, el exterior recorrerá cada uno de los vehículos y el interno cada uno de los datos del vehículo:

Ejemplo:

```
<style>
      table {
            border:1px solid black;
            border-collapse:collapse;
      }

      td {
```

```
            border:1px solid black;
            width:100px;
      }
</style>

<script type="text/javascript">
      var vehiculos= [[100, "Ford Focus", 3, 18500],
                  [230, "Seat Ibiza", 6, 13000],
                  [345, "Peugeot 307", 2, 23800],
                  [457, "Audi A3", 1, 32600]];

      document.write("<table>");
      for(i in vehiculos) {
            document.write("<tr>");
            for(j in vehiculos[i]){
                  document.write("<td>"+vehiculos[i][j]+"</td>");
            }
            document.write("</tr>");
      }
      document.write("</table>");
</script>
```

Arrays Asociativo.

Los arrays asociativos son arrays cuyos índices son cadenas con valores personalizados.

Ejemplo:

```
var autos={"codigo":100, "marca":"Ford Focus", "unidades":3, "precio":18500};
```

Podemos acceder a los elementos del array asociativo utilizando el índice asociado entre comillas dentro del corchete que indica la posición:

```
autos["marca"]
```

Podemos recorrer el array con una estructura de bucle for .. in:

```
      var autos={"codigo":100, "marca":"Ford Focus", "unidades":3,
            "precio":18500};
      for(i in autos){
            document.write(i+": "+autos[i]+"</br>");
      }
```

4.4.- Objetos definidos por el usuario.

El usuario puede crear sus propios objetos con propiedades y métodos. Las propiedades son características del Objeto y los métodos son funciones que actúan sobre las propiedades.

La sintaxis para la declaración de objetos definidos por el usuario es:

```
      function mi_objeto(valor1, valor2, …, valorn){
            propiedades_objeto;
            metodos_objeto;
      }
```

Ejemplo:

```
function Coche(codigo_in, marca_in, unidades_in, precio_in){
	this.codigo = codigo_in;
	this.marca = marca_in;
	this.unidades = unidades_in;
	this.precio = precio_in;
}
```

Podemos crear instancias del objeto predefinido por el usuario con la sentencia new:

`var instancia = new Clase();`

Ejemplo:

`var miCoche = new Coche(104, "Ford KA", 7, 10450);`

Podemos añadir propiedades a la instancia del objeto definido por el usuario utilizando:

`instancia.propiedad = valor;`

Ejemplo:

`miCoche.color = "Gris";`

Podemos establecer un objeto como propiedad de otro objeto. Por ejemplo podemos crear un objeto llamado Concesionario que sea una propiedad del objeto Coche.

Ejemplo:

```
<script type="text/javascript">
	function Concesionario(cod_oficina_in, ciudad_in, telefono_in){
		this.cod_oficina = cod_oficina_in;
		this.ciudad = ciudad_in;
		this.telefono = telefono_in;
	}

	function Coche(codigo_in, marca_in, unidades_in, precio_in,
concesionario_in){
		this.codigo = codigo_in;
		this.marca = marca_in;
		this.unidades = unidades_in;
		this.precio = precio_in;
		this.concesionario = concesionario_in;
	}

	var concesionario_sur = new Concesionario(12, "Málaga", "952039123");
	var miCoche = new Coche(104, "Ford KA", 7, 10450, concesionario_sur);
	miCoche.color = "Gris";
</script>
```

Podemos crear métodos y asignarlos a la definición del objeto. Por lo que podríamos crear un método que imprima los datos del Coche y añadir dicha función a la definición del objeto.

Ejemplo:

```
function imprimeDatos(){
      document.write("Código: "+this.codigo+"<br>");
      document.write("Marca: "+this.marca+"<br>");
      document.write("Unidades: "+this.unidades+"<br>");
      document.write("Precio: "+this.precio+"<br>");
      document.write("Código Oficina Concesionario: "
            +this.concesionario.cod_oficina+"<br>");
      document.write("Ciudad Concesionario: "
            +this.concesionario.ciudad+"<br>");
      document.write("Teléfono Concesionario: "
            +this.concesionario.telefono+"<br>");
}

function Concesionario(cod_oficina_in, ciudad_in, telefono_in){
      this.cod_oficina = cod_oficina_in;
      this.ciudad = ciudad_in;
      this.telefono = telefono_in;
}

function Coche(codigo_in, marca_in, unidades_in, precio_in,
      concesionario_in){
      this.codigo = codigo_in;
      this.marca = marca_in;
      this.unidades = unidades_in;
      this.precio = precio_in;
      this.concesionario = concesionario_in;
      this.imprimeDatos = imprimeDatos;
}
```

Instanciamos y asignamos datos a la instancia concesionario_sur del objeto Concesionario y a la instancia miCoche del objeto Coche:

```
var concesionario_sur = new Concesionario(12, "Málaga", "952039123");
var miCoche = new Coche(104, "Ford KA", 7, 10450, concesionario_sur);
```

Para llamar al método imprimeDatos() de la instancia miCoche del objeto Coche utilizaremos la sentencia:

```
miCoche.imprimeDatos();
```

Podemos añadir propiedades **color** a la instancia miCoche del objeto Coche y mostrar su valor:

```
miCoche.color = "Gris";
document.write("Color: "+miCoche.color+"<br>");
```

Otra posibilidad es crear la clase mediante la sentencia **class**. Crearemos el constructor de la clase con la palabra reservada **constructor** y entre paréntesis los parámetros que pasamos al constructor. Por último, podemos definir los métodos mediante el **nombre del método** y entre paréntesis los parámetros que pasamos al método:

```
class nombreClase {
      constructor(dato1, dato2,..){
            this.dato1 = dato1;
            this.dato2 = dato2;
            :     :     :
      }

      metodo1 (dato1, dato2,…) {
            :     :     :
      }

      :     :     :     :
}
```

Ejemplo:

```
<script>

      class Concesionario {
            constructor(cod_oficina_in, ciudad_in, telefono_in){
                  this.cod_oficina = cod_oficina_in;
                  this.ciudad = ciudad_in;
                  this.telefono = telefono_in;
            }
      }

      class Coche {
            constructor(codigo_in, marca_in, unidades_in, precio_in,
concesionario_in){
                  this.codigo = codigo_in;
                  this.marca = marca_in;
                  this.unidades = unidades_in;
                  this.precio = precio_in;
                  this.concesionario = concesionario_in;
            }

            imprimeDatos(){
                  document.write("Código: "+this.codigo+"<br>");
                  document.write("Marca: "+this.marca+"<br>");
                  document.write("Unidades: "+this.unidades+"<br>");
                  document.write("Precio: "+this.precio+"<br>");
                  document.write("Código Oficina Concesionario: "
```

```
                        +this.concesionario.cod_oficina+"<br>");
                  document.write("Ciudad Concesionario: "
                        +this.concesionario.ciudad+"<br>");
                  document.write("Teléfono Concesionario: "
                        +this.concesionario.telefono+"<br>");
            }
      }

      var concesionario_sur = new Concesionario(12, "Málaga",
"952039123");
      var miCoche = new Coche(104, "Ford KA", 7, 10450,
concesionario_sur);

      miCoche.imprimeDatos();

      miCoche.color = "Gris";
      document.write("Color: "+miCoche.color+"<br>");

</script>
```

Javascript ofrece la posibilidad de realizar herencia de clases mediante la palabra reservada **extends**. Si necesitamos acceder a un método de la clase padre podemos utilizar la sentencia **super**.

Ejemplo:

```
<script>
      class Persona {
            constructor(nombre, apellidos, edad){
                  this.nombre = nombre;
                  this.apellidos = apellidos;
                  this.edad = edad;
            }
            toString(){
                  document.write("Nombre: "+ this.nombre+"<br>");
                  document.write("Apellidos:                          "+
this.apellidos+"<br>");
                  document.write("Edad: "+ this.edad+"<br>");
            }
      }

      class Empleado  extends Persona {
            constructor(nombre, apellidos, edad, sueldo) {
                  super(nombre, apellidos, edad);
                  this.sueldo = sueldo;
            }
```

```
            toString(){
                  super.toString();
                  document.write("Sueldo: "+this.sueldo+"<br>");
            }
      }

      empleado_01 = new Empleado("Marta", "Mas Moreno", 34, 2100);
      empleado_01.toString();
</script>
```