

Variables

var	Variable de <i>scope</i> global.	var nombre = 'Martin'
let	Variable de <i>scope</i> local. Solo tiene efecto entre los {}.	let numero = 2
const	Contante su valor no cambia nunca.	const pi= 3.14

Funciones

Función Declarada	Se declara usando la <i>estructura básica</i> .	function sumar (a,b){ return a+b; }
Función Expresada	Se asigna como <i>valor</i> de una <i>variable</i> .	let calculadora = function (a,b){ return a+b }
Invocación de funcion	Se indica los argumentos para las variables de la función respetando el orden.	function saludar(ap, nom){ return 'Hola ' + nombre + ' + apellido; } function('Martin', 'Ordoñez'); (retorna 'Hola Martin Ordoñez')
Función Constr-uctora	Permite armar un <i>molde</i> para crear objetos. Por convención el nombre de la funcion constr-uctora se escribe la primer letra con mayúsculas.	function Auto(marca, modelo){ this.marca = marca; this.modelo=modelo; }
Instanciar un objeto	Para crea un objeto se usa la palabra New y se llama a la funcion.	let autoNuevo = new Auto('Ford','Falcon');

Funciones (cont)

Arrow Functions	Permite escribir las funciones con una <i>sintaxis</i> mas <i>compacta</i> .	let sumar =(a,b) =>a+b; (*let esMultiplo =(a,b) => { let resto = a % b; return resto==0; })
Callback	Es una <i>funcion</i> que pasa como <i>parámetro</i> de otra funcion. Puede estar definida o no.	setTimeout(function(){ console.log('Holamund-o')},1000) let miCallback= () => console.log('Hola Mundo') setTimeout(miCallback,1000);

(*)En las arrow functions no hace falta escribir {} ni return** salvo que haya as de una línea de código.-

JSON

JSON.parse()	Recibe una cadena de texto en formato JSON y la devuelve en formato JS	let datos = ('{"nombre":"Martin", "edad":46}'); console.log (JSON.parse(datos); //Devuelve {nombre: 'Martin', edad:46}
JSON.stringify()	Recibe un dato de JS y devuelve una cadena de texto JSON	let objeto={nombre: 'Martin', edad:46} console.log(JSON.stringify(objeto); //Devuelve '{"nombre":"Martin", "- edad":46}'

JavaScript Object Notation similar a un *Objeto Literal* se usa para intercambiar información entre sistemas.

```
{ "texto": "mi texto",  
  "numero": 16,  
  "array": ["uno", "dos"],  
  "booleano": true }
```

Strings

.length	Retorna la <i>cantidad total de caracteres</i> .	let saludo = 'Hola mundo!'; console.log(saludo.length); // Devuelve 11
indexOf()	Busca en el string, el string que recibe como parámetro.	console.log(saludo.indexOf(mundo)); //Devuelve 5 por la posición de la letra 'm'
.slice()	Corta el string y devuelve la parte donde se aplica.	console.log(saludo.slice(0,4)); //Devuelve 'Hola'
.trim()	Elimina espacios al principio y final del string.	let saludo = ' Hola mundo!'; console.log(saludo.trim()); //Devuelve 'Hola Mundo!'
split()	Recibe un <i>string</i> que usa como separador y devuelve un <i>array</i> con las partes del string.	console.log(saludo.split(' ')); //Devuelve ['Hola','mundo!']
replace()	Reemplaza una parte del string con otra°	console.log(saludo.replace('mundo','pais')); //Devuelve 'Hola Pais!'

Los *string* son un *array* de caracteres. El índice al igual que en el array comienza en 0.
P/Ej.: nombre=¡Hola!, soy Carli! //nombre[2] devuelve la letra 'o'

Ciclos

for	Permite <i>Repetir Instrucciones</i> de manera sencilla.	for (let i=0 ; i<=x ; i=i+2){ suma=suma+i; }
------------	--	---

Arrays (métodos)

.length	<i>Longitud</i> del array. Indica la cantidad de elementos que componen el array.	let colores=['Rojo','Azul','Amarillo']; console.log(colores.length); //Devuelve 3
----------------	---	---

Arrays (métodos) (cont)

.pop()	Elimina el ultimo elemento del array. Devuelve el elemento eliminado.	let ultimoColor = colores.pop(); console.log(colores); //Devuelve ['Rojo','Azul'] console.log(ultimoColor); //Devuelve 'Amarillo'
.push()	Agrega uno o mas elementos al final del array.	console.log(colores.push('Violeta','Gris')); //Devuelve ['Rojo','Azul','Amarillo','Violeta','Gris'];
.shift()	Elimina el primer elemento del array. Devuelve el elemento eliminado.	let primerColor = colores.shift(); console.log(colores); //Devuelve ['Azul','Amarillo'] console.log(primerColor); //Devuelve 'Rojo'
.unshift()	Agrega uno o mas elementos al principio del array.	console.log(colores.unshift('Violeta')); //Devuelve ['Violeta','Rojo','Azul','Amarillo']
.join()	Une los elementos del array usando un separador. Se debe indicar el separador.	console.log(colores.join()); //Devuelve 'Rojo, 'Azul, 'Amarillo' console.log(colores.join('-')); //Devuelve 'Rojo-Azul-Amarillo'
indexOf()	Busca en el array el parámetro indicado. Si no lo encuentra retorna -1. Siempre retorna el primer índice ocupado por el parámetro.	console.log(colores.indexOf('Azul')); //Devuelve 1

Arrays (métodos) (cont)

lastIndexOf Similar a `indexOf` pero busca de Derecha a Izquierda.

```
console.log(colores.lastIndexOf('Amarillo'));
//Devuelve 0
```

.includes Busca en el array el parámetro indicado y retorna un *Booleano*.

```
console.log(colores.includes('Azul'));
//Devuelve truef
```

.map() Recibe una función (callback) como parámetro y devuelve un array modificado.

```
let numeros =[2,4,6];
let doble = numeros.map(function(num){
  return 2*num;})
console.log(doble);// Devuelve [4,8,12]
```

filter() Recorre el array y filtra según la condición establecida.

```
let edades =[22,46,6,18,5,25,3];
let mayores = edades.filter(function(edad){
  return edad>=18;
}) console.log(mayores)// Devuelve [22,46,18,25]
```

reduce() Recorre el array y devuelve un único elemento. En este caso el callback recibe dos parámetros, un acumulador u el elemento actual que esta recorriendo.

```
let num = [5,7,16]
suma=num.reduce(function(acumulador,elemento){
  return acumulador+elemento;
}) console.log(suma);// Muestra 28
```

Arrays (métodos) (cont)

forEach() Tiene la finalidad de iterar sobre el array. Recibe un callback y un index (no es necesario escribir el index salvo que se necesite a futuro).

```
let paises=['Argentina' ,
'Colombia' , 'Chile' ,
'Ecuador'];
paises.forEach(function(pais)
{
  console.log(pais)
}) //Devuelve: Argentina,
Colombia, Chile, Ecuador
```

Las arrays se indican entre [] y son colecciones de datos ordenados.
 P/Ej.: `colores=['Rojo','Azul','Amarillo']`
El índice de cada elemento comienza en 0. P/Ej.: `colores[0]`
 //devuelve Rojo y `colores[2]` //devuelve Amarillo.

Destructuring

Desestructurar un *Array* u *Objeto Literal* consiste en extraer los elementos del del mismo. Esta acción no modifica el array u objeto literal, solo extrae los datos

Array

```
let destinosD elMundo = ['Marruecos', 'Baril -
oche', 'Barcelona', 'China', 'Grecia']
let[, , china ]=destinosD elMundo
console.log(china)// Devuelve china
```

Objetos Literales

```
let auto = {marca: 'Ferrari', kilometros: 31,
color: "Rojo"};
let { marca } = auto;
console.log(marca) //Devuelve Ferrari
```

Tipos de Datos

numéricos (number)	Números enteros o decimales (el separador decimal es el ".")	1, 2, 7, 4.5
cadenas de caracteres (string)	Cadena de texto se escriben entre " o '	"Hola Mundo!" 'Hola Mundo!'
lógicos o booleanos	Sus valores pueden ser <i>true</i> o <i>false</i>	6 < 8 = false

Tipos de Datos (cont)

objetos literales	Son <i>colecciones de datos</i> . Se reconocen por estar encerrados entre {}	let datos={Nombre:'Javier', edad: 55, soltero: true}
arrays	<i>Coleccion de datos</i> se declaran con []	let Comidas=["Milanesa",'- Ravioles con salsa','budin de pan']
NaN (<i>Not a number</i>)	Indica que un valor no puede ser evaluado como número	let division='35'/2 = NaN
Null (nulo)	Indica valor vacío o desconocido	let temperatura=null
Undefined (sin definir)	Indica ausencia de valor	let saludo;

Condicionales

if	Condicional Simple	let edad = 19; let acceso = " ; if (edad<16){ acceso = 'Prohibido'; }
else if	Agrega otra condición en el caso que la anterior sea falsa. Es opcional	let edad = 19; let acceso = " ; if (edad<16){ acceso = 'Prohibido'; }else if(edad>=16 && edad<=19){ acceso = 'Acompañado de un mayor'; }

Condicionales (cont)

else	Código a ejecutar si las condiciones anteriores son falsas. Es opcional	let edad = 19 let acceso = " if (edad<16){ acceso = 'Prohibido'; }else if(edad>=16 && edad<=19){ acceso = 'Acompañado de un mayor'; }else{ acceso = 'Permitido'; }
if ternario	Se escribe de forma <i>horizontal</i> .	4 > 10 ? 'EL 4 es mas grande' : 'El 10 es mas grande'
Switch	Esta compuesto por una expresión a evaluar, seguida de diferentes casos que terminan en <i>break</i> .	switch (fruta) { case 'manzana': case 'pera': console.log('Mas rica es la Naranja!'); break; default: console.log('Que fruta es?'); break; }

Operadores

De Asignación (=)	Asigna un valor a una variable	let color = 'rojo'
Aritméticos	Suma Resta Multiplicación División Incremento Decremento Módulo (devuelve el resto de una división)	15+6 = 21 15-6=9 2*6=12 21/3=7 15++=16 17--=16 15%2=1

Operadores (cont)

Comparación	Igualdad simple	15==15 -> True
	Desigualdad simple	10!=15->true
	Igualdad estricta	5==='15'->false
	Desigualdad estricta	10!=='10'->true
	Mayor	15>15->false
	Mayor igual	15>=15->true
	Menor	10<15->true
	Menor igual	15<=15->true

Lógicos	Y: and (&&)//Y	10>15 && 5<3-> false
	O: or ()	10>15 5<3->true
	No es: not (!)	!10<15->false

Concatenación	Une distintos tipos de datos.	let fila = 'M';
	Siempre devuelve un string	let asiento=7;
		let ubicacion=fila +
		asiento -> M7 como
		string

Objeto Date

.getDay()	Nos devuelve el día de la semana, donde 0 es Domingo y Sábado 7	let miFecha = fechaActual.getDay(); console.log(miFecha); Devuelve 5 (hoy es viernes)
getDate()	Nos devuelve el día del mes.	let miFecha = fechaActual.getDate(); console.log(miFecha); Devuelve 15 (hoy es Viernes 15 de Abril)
getMonth()	Nos devuelve el número de mes del año. Tener en cuenta que enero = 0 y diciembre = 11	let miFecha = fechaActual.getMonth(); console.log(miFecha); Devuelve 3 (abril)
getFullYear()	Nos devuelve el año actual.	let miFecha = fechaActual.getFullYear(); console.log(miFecha); Devuelve 2022

Objeto Date (cont)

new Date()	Permite <i>crear</i> una fecha. si no se indica nada devolverá la fecha actual	let miCumple = new Date(1975,08,06); console.log(miCumple); Devuelve 1975-09-06T03:00:00.000Z(**)
-------------------	--	---

Antes de usar el Objeto Date es necesario crear una instancia del mismo en una variable. P/Ej.:
let fechaActual = new Date;
console.log(fechaActual) // Devuelve 2022-04-15T12:25:00.538Z (**)
Notese que el mes en la fecha ingresada es 08 y la instancia devolvió 09, sumando automáticamente 1 al mes.-

Objetos Literales

objeto.propiedad	Accedemos al valor de una propiedad del objeto.	console.log(tenista.a.activo); //Devuelve true
Metodo	Cuando una <i>propiedad</i> almacena una <i>función</i> , a esa función la llamamos <i>método</i> .	let tenista = { nombre: 'Roger', edad: 38, activo: true, saludar: function(){ return '¡Hola! me llamo Roger' } }
objeto.metodo()	Ejecuta el método (<i>función</i>) almacenado en un objeto.	console.log(tenista.saludar()); //Devuelve '¡Hola! me llamo Roger'

Objetos Literales (cont)

this *Accedemos al valor de cada propiedad del objeto.*

```
let tenista = {
  nombre: 'Roger',
  edad: 38,
  activo: true,
  saludar: function() {
    return '¡Hola! me llamo ' + this.nombre;
  }
}
console.log(tenista.saludar());
//Devuelve '¡Hola! me llamo Roger'
```

Un Objeto es una estructura de datos.

```
let tenista = {
  nombre: 'Roger',
  edad: 38,
  activo: true,
}
```

Donde: **nombre**, **edad** y **activo** son *propiedades* y **'roger'**, **38** y **true** los *Valores* de esas propiedades.

Spread operator y Rest parameter

Spread operator *Permite expandir los elementos de un elemento iterable dentro de otro.*

```
let dias1 = ['Lunes', 'Martes', 'Miercoles'];
let dias2 = ['Jueves', 'Viernes', 'Sabado', 'Domingo'];
let semana = [...dias1, ...dias2];
console.log(semana); // Devuelve [
'Lunes', 'Martes', 'Miercoles', 'Jueves',
'Viernes', 'Sabado', 'Domingo' ]
```

Spread operator y Rest parameter (cont)

Spread operator *Idem en objetos*

```
let auto = {marca: 'Ferrari', anio: '2019'};
let piloto = {nombre: 'Vettel', edad: '31', ...auto};
console.log(piloto) //devuelve {
nombre: 'Vettel', edad: '31',
marca: 'Ferrari', anio: '2019' }
```

Spread operator *en funciones*

Rest Parameter *Usando el operador "..." como último parámetro de una función, permite capturar los parámetros adicionales de la función.*

```
function sumar(...numeros) {
  return numeros.reduce((acum, num) => acum += num);
}
console.log(sumar(6, 3)); // Devuelve 9
```