Capitulo 2: Programacón Imperativa

Apéndice

* [Referencia rápida del lenguaje JavaScript](https://mumuki.io/argentina-programa/chapters/531-programacion-imperativa/appendix#referencia-rapida-del-lenguaje-javascript)
  + [Declaración de Funciones](https://mumuki.io/argentina-programa/chapters/531-programacion-imperativa/appendix#declaracion-de-funciones)
  + [Operadores matemáticos](https://mumuki.io/argentina-programa/chapters/531-programacion-imperativa/appendix#operadores-matematicos)
  + [Operadores lógicos](https://mumuki.io/argentina-programa/chapters/531-programacion-imperativa/appendix#operadores-logicos)
  + [Comparaciones](https://mumuki.io/argentina-programa/chapters/531-programacion-imperativa/appendix#comparaciones)
  + [Alternativa Condicional](https://mumuki.io/argentina-programa/chapters/531-programacion-imperativa/appendix#alternativa-condicional)
  + [Variables](https://mumuki.io/argentina-programa/chapters/531-programacion-imperativa/appendix#variables)
  + [Repetición indexada](https://mumuki.io/argentina-programa/chapters/531-programacion-imperativa/appendix#repeticion-indexada)
* [Biblioteca simplificada](https://mumuki.io/argentina-programa/chapters/531-programacion-imperativa/appendix#biblioteca-simplificada)
  + [longitud(unString)](https://mumuki.io/argentina-programa/chapters/531-programacion-imperativa/appendix#longitudunstring)
  + [convertirEnMayuscula(unString)](https://mumuki.io/argentina-programa/chapters/531-programacion-imperativa/appendix#convertirenmayusculaunstring)
  + [comienzaCon(unString, otroString)](https://mumuki.io/argentina-programa/chapters/531-programacion-imperativa/appendix#comienzaconunstring-otrostring)
  + [imprimir(unString)](https://mumuki.io/argentina-programa/chapters/531-programacion-imperativa/appendix#imprimirunstring)
  + [tirarDado()](https://mumuki.io/argentina-programa/chapters/531-programacion-imperativa/appendix#tirardado)
  + [listasIguales](https://mumuki.io/argentina-programa/chapters/531-programacion-imperativa/appendix#listasiguales)
  + [longitud(unaLista)](https://mumuki.io/argentina-programa/chapters/531-programacion-imperativa/appendix#longitudunalista)
  + [agregar(unaLista, unElemento)](https://mumuki.io/argentina-programa/chapters/531-programacion-imperativa/appendix#agregarunalista-unelemento)
  + [remover(unaLista)](https://mumuki.io/argentina-programa/chapters/531-programacion-imperativa/appendix#removerunalista)
  + [posicion(unaLista, unElemento)](https://mumuki.io/argentina-programa/chapters/531-programacion-imperativa/appendix#posicionunalista-unelemento)
* [Bibliografía complementaria](https://mumuki.io/argentina-programa/chapters/531-programacion-imperativa/appendix#bibliografia-complementaria)

Referencia rápida del lenguaje JavaScript

El lenguaje JavaScript es utilizado ampliamente para construir software en todo el mundo, siendo una de las principales tecnologías de la Web. En Mumuki sólo usamos una muy pequeña parte del mismo, que listamos a continuación:

Declaración de Funciones

A partir de la [Lección 1: Funciones y tipos de datos](https://mumuki.io/argentina-programa/guides/flbulgarelli/fundamentos-javascript-funciones-tipos-de-datos)

Las funciones en JavaScript se declaran mediante la *palabra clave* function, y su cuerpo va entre llaves { y }:

function nombreDeLaFuncion(parametro1, parametro2, parametro3) {

return ...;

}

Toda función debe tener al menos un retorno, que se expresa mediante return.

Operadores matemáticos

A partir de la [Lección 1: Funciones y tipos de datos](https://mumuki.io/argentina-programa/guides/flbulgarelli/fundamentos-javascript-funciones-tipos-de-datos)

4 + 5

10 - 5

8 \* 9

10 / 5

Operadores lógicos

A partir de la [Lección 1: Funciones y tipos de datos](https://mumuki.io/argentina-programa/guides/flbulgarelli/fundamentos-javascript-funciones-tipos-de-datos)

true && false

true || false

! false

Comparaciones

A partir de la [Lección 1: Funciones y tipos de datos](https://mumuki.io/argentina-programa/guides/flbulgarelli/fundamentos-javascript-funciones-tipos-de-datos)

// para cualquier tipo de dato

"hola" === "hola"

"hola" !== "chau"

// para números

4 >= 5

4 > 5

4 <= 5

4 < 5

Alternativa Condicional

A partir de la [Lección 1: Funciones y tipos de datos](https://mumuki.io/argentina-programa/guides/flbulgarelli/fundamentos-javascript-funciones-tipos-de-datos)

Los ifs en JavaScript encierran la condición entre paréntesis y su cuerpo entre llaves:

if (hayPersonasEnEspera()) {

llamarSiguientePersona();

}

Además, los ifs pueden opcionalmente tener un else:

if (hayPersonasEnEspera()) {

llamarSiguientePersona();

} else {

esperarSiguientePersona();

}

Por último, podemos combinar varios ifs para tomar decisiones ante múltiples condiciones:

if (hayPersonasEnEspera()) {

llamarSiguientePersona();

} else if (elPuestoDebeSeguirAbierto()) {

esperarSiguientePersona();

} else {

cerrarPuesto();

}

Variables

A partir de la [Lección 3: Variables y procedimientos](https://mumuki.io/argentina-programa/guides/flbulgarelli/mumuki-guia-fundamentos-javascript-variables-y-procedimientos)

Las variables nos permiten *recordar* valores y se declaran mediante la palabra reservada let y se les da un valor inicial usando =:

let pesosEnMiBilletera = 100;

let diasQueFaltanParaElVerano = 10;

La mismas se asignan mediante =:

pesosEnMiBilletera = 65;

diasQueFaltanParaElVerano = 7;

En ocasiones las asignaremos usando el valor anterior:

pesosEnMiBilletera = pesosEnMiBilletera \* 2;

diasQueFaltanParaElVerano = diasQueFaltanParaElVerano - 1;

La asignación anterior se puede compactar combinando el signo = y la operación:

pesosEnMiBilletera \*= 2;

diasQueFaltanParaElVerano -= 1;

Repetición indexada

A partir de la [Lección 7: Recorridos](https://mumuki.io/argentina-programa/guides/mumukiproject/mumuki-guia-javascript-practica-de-listas-y-registros)

Las listas pueden ser *recorridas*, visitando y haciendo algo con cada uno de sus elementos. Para ello contamos con la estructura de control for..of, que encierra su generador entre paréntesis (( y )) y su cuerpo entre llaves ({ y }):

let patrimoniosDeLaHumanidad = [

{declarado: 1979, nombre: "Parque nacional Tikal", pais: "Guatemala"},

{declarado: 1983, nombre: "Santuario histórico de Machu Picchu", pais: "Perú"}

{declarado: 1986, nombre: "Parque nacional do Iguaçu", pais: "Brasil"},

{declarado: 1995, nombre: "Parque nacional de Rapa Nui", pais: "Chile"},

{declarado: 2003, nombre: "Quebrada de Humahuaca", pais: "Argentina"}

]

let cantidadPatrimoniosDeclaradosEnEsteSiglo = 0;

for (let patrimonio of patrimoniosDeLaHumanidad) {

if (patrimonio.declarado >= 2000) {

cantidadPatrimoniosDeclaradosEnEsteSiglo += 1;

}

}

Biblioteca simplificada

Dentro de Mumuki usamos una biblioteca de funciones inspirada en la que ya viene con JavaScript, pero simplifiacada para que sea más sencilla y segura de usar. A continuación listamos las principales funciones que se pueden usar, indicando el equivalente *real* en JavaScript cuando corresponda.

longitud(unString)

A partir de la [Lección 1: Funciones y tipos de datos](https://mumuki.io/argentina-programa/guides/flbulgarelli/fundamentos-javascript-funciones-tipos-de-datos)

Versión simplificada de [length](https://developer.mozilla.org/es/docs/Web/JavaScript/Referencia/Objetos_globales/String/length" \o "" \t "_blank)

Uso:

longitud("hola")

4

convertirEnMayuscula(unString)

A partir de la [Lección 1: Funciones y tipos de datos](https://mumuki.io/argentina-programa/guides/flbulgarelli/fundamentos-javascript-funciones-tipos-de-datos)

Versión simplificada de [toUpperCase](https://developer.mozilla.org/es/docs/Web/JavaScript/Referencia/Objetos_globales/String/toUpperCase" \o "" \t "_blank)

Convierte un unString en mayúsculas:

convertirEnMayuscula("hola")

"HOLA"

comienzaCon(unString, otroString)

A partir de la [Lección 1: Funciones y tipos de datos](https://mumuki.io/argentina-programa/guides/flbulgarelli/fundamentos-javascript-funciones-tipos-de-datos)

Versión simplificada de [startsWith](https://developer.mozilla.org/es/docs/Web/JavaScript/Referencia/Objetos_globales/String/startsWith" \o "" \t "_blank)

Dice si unString empieza con otroString:

comienzaCon("aprendiendo a programar", "aprendiendo")

true

comienzaCon("aprendiendo a programar", "aprend")

true

comienzaCon("aprendiendo a programar", "programar")

false

comienzaCon("aprendiendo a programar", "tomar el té")

false

imprimir(unString)

A partir de la [Lección 3: Variables y procedimientos](https://mumuki.io/argentina-programa/guides/flbulgarelli/mumuki-guia-fundamentos-javascript-variables-y-procedimientos)

Versión simplificada de [console.log](https://developer.mozilla.org/es/docs/Web/API/Console/log)

Imprime por pantalla unString:

imprimir("¡estoy imprimiendo!")

¡estoy imprimiendo!

tirarDado()

A partir de la [Lección 3: Variables y procedimientos](https://mumuki.io/argentina-programa/guides/flbulgarelli/mumuki-guia-fundamentos-javascript-variables-y-procedimientos)

Devuelve al azar un número entre 1 y 6:

tirarDado()

5

tirarDado()

1

tirarDado()

2

listasIguales(unaLista, otraLista)

A partir de la [Lección 5: Listas](https://mumuki.io/argentina-programa/guides/flbulgarelli/mumuki-guia-fundamentos-javascript-vectores)

listasIguales([1,4,7], [1,4,7])

true

listasIguales([1,4,7], [1,4,8])

false

longitud(unaLista)

A partir de la [Lección 5: Listas](https://mumuki.io/argentina-programa/guides/flbulgarelli/mumuki-guia-fundamentos-javascript-vectores)

* [length de listas](https://developer.mozilla.org/es/docs/Web/JavaScript/Referencia/Objetos_globales/Array/length)

Nos dice cuan largo es unaLista:

longitud([true, false, false, true])

4

longitud([5, 6, 3])

3

agregar(unaLista, unElemento)

A partir de la [Lección 5: Listas](https://mumuki.io/argentina-programa/guides/flbulgarelli/mumuki-guia-fundamentos-javascript-vectores)

Versión simplificada de [push](https://developer.mozilla.org/es/docs/Web/JavaScript/Referencia/Objetos_globales/Array/push" \o "" \t "_blank)

Inserta unElemento al final de unaLista. Este es un procedimiento que no devuelve nada pero modifica a unaLista:

let cancionesFavoritas = ["La colina de la vida", "Zamba por vos"]

// agrega el elemento "Seminare" a la lista cancionesFavoritas

agregar(cancionesFavoritas, "Seminare")

// ahora la lista tiene un elemento más:

cancionesFavoritas

["La colina de la vida", "Zamba por vos", "Seminare"]

remover(unaLista)

A partir de la [Lección 5: Listas](https://mumuki.io/argentina-programa/guides/flbulgarelli/mumuki-guia-fundamentos-javascript-vectores)

Versión simplificada de [pop](https://developer.mozilla.org/es/docs/Web/JavaScript/Referencia/Objetos_globales/Array/pop)

Quita el último elemento de unaLista. Este es un procedimiento que no devuelve nada pero modifica a unaLista:

let listaDeCompras = ["leche", "pan", "arroz", "aceite", "yerba"]

// removemos el último elemento

remove(listaDeCompras)

// la "yerba" ya no está en lista de compras

listaDeCompras

["leche", "pan", "arroz", "aceite"]

posicion(unaLista, unElemento)

A partir de la [Lección 5: Listas](https://mumuki.io/argentina-programa/guides/flbulgarelli/mumuki-guia-fundamentos-javascript-vectores)

Versión simplificada de [indexOf](https://developer.mozilla.org/es/docs/Web/JavaScript/Referencia/Objetos_globales/Array/indexOf" \o "" \t "_blank)

Nos dice en qué posición se encuentra unElemento dentro de unaLista. Si el elemento no está en la lista, devuelve -1

let premios = ["dani", "agus", "juli", "fran"]

posicion(premios, "dani")

0

posicion(premios, "juli")

2

posicion(premios, "feli")

-1

Bibliografía complementaria

* <https://developer.mozilla.org/es/docs/Web/JavaScript>
* <https://es.javascript.info/>