



# ¿Y el tablero?

- 

Hasta ahora en esta película hay un gran personaje que está faltando: *el tablero*. Seguro está por aparecer, de forma triunfal y rimbombante..., ¿no?

No. En JavaScript, lamentamos informarte, no hay tablero .

Pero tampoco es tan grave: en JavaScript no hay tablero, ¡porque no lo necesitás! Suceden dos cosas:

1. El tablero nos servía para *ver* lo que nuestro programa hacía y qué resultados generaba. Nos permitía también observar los cambios de estado a causa del programa. Pero ahora ya tenemos experiencia suficiente como para lanzarnos a programar sin tener que "ver" lo que sucede.
2. Ahora contamos con la **consola**: una herramienta poderosa que nos permite hacer pruebas más detalladas y flexibles.

¿No nos creés? Te presentamos un desafío: usando la consola, decí con tus propias palabras qué hace la función `funcionMisteriosa` , que recibe dos números enteros como argumentos.  
¡Vas a ver que podés averiguarlo sin tener un tablero!

>



Esta guía fue desarrollada por Franco Bulgarelli bajo los términos de la [Licencia Creative Commons Compartir-Igual, 4.0](#).

© 2015-2022 Ikumi SRL

[Información importante](#)

[Términos y Condiciones](#)

[Reglas del Espacio de Consultas](#)





# Impresión por pantalla

•

Ahora que ya te convencimos de que no necesitamos al tablero, vamos a mostrarte que sí hay algo parecido en JavaScript : la impresión por pantalla. Veamos un ejemplo:

```
function funcionEgocentrica() {  
  imprimir("soy una función que imprime por pantalla");  
  imprimir("y estoy por devolver el valor 5");  
  return 5;  
}
```

Probá `funcionEgocentrica` en la consola.

>



Esta guía fue desarrollada por Franco Bulgarelli bajo los términos de la [Licencia Creative Commons Compartir-Igual, 4.0](#).

© 2015-2022 Ikumi SRL

[Información importante](#)

[Términos y Condiciones](#)

[Reglas del Espacio de Consultas](#)







# Martín Fierro



¿Qué acabamos de hacer con esto? Al igual que `Per(bolita)`, `imprimir` es una funcionalidad que siempre está disponible. Si llamamos a la función anterior, veremos que, además de devolver el valor 5, imprime dos líneas:

```
soy una función que imprime por pantalla
y estoy por devolver el valor 5
```



Sin embargo, sólo podemos escribir strings y, una vez que escribimos en la pantalla, no hay vuelta atrás: no hay forma de retroceder o deshacer.

Veamos si va quedando claro, definí la `function` `versosMartinFierro` que imprima por pantalla los primeros versos del Martín Fierro:

```
Aquí me pongo a cantar
Al compás de la vigüela;
Que el hombre que lo desvela
Una pena extraordinaria
```



Esta `function` debe retornar 0

💡 ¡Dame una pista!

Solución Consola

```
1 function versosMartinFierro () {
2   imprimir("Aquí me pongo a cantar");
3   imprimir("Al compás de la vigüela;");
4   imprimir("Que el hombre que lo desvela");
5   imprimir("Una pena extraordinaria");
6   return 0;
7 }
```



 Enviar

✓ ¡Muy bien! Tu solución pasó todas las pruebas

¡Bien hecho!

Sin embargo, ¿tiene sentido que `versosMartinFierro` devuelva `0`? ¿Usamos para algo este resultado?

Acá parecería que llamamos a esta `function` porque nos interesa su efecto de imprimir líneas; nos da igual lo que retorna. Quizás más que una función, necesitamos definir un procedimiento. ¿Se podrá hacer esto en JavaScript?

La respuesta, ¡en el siguiente ejercicio!

---

Esta guía fue desarrollada por Franco Bulgarelli bajo los términos de la [Licencia Creative Commons Compartir-Igual, 4.0](#).

© 2015-2022 Ikumi SRL

[Información importante](#)

[Términos y Condiciones](#)

[Reglas del Espacio de Consultas](#)





# ¿Y los procedimientos?

•

En el ejercicio anterior, construiste una `function` que se ejecutaba con el ~~solo~~ fin de imprimir por pantalla. Y por ello, tuvimos que devolver un valor cualquiera. 4. ¿Y los procedimientos? ¿No te huele mal?

Además, hagamos memoria: cuando queremos reutilizar código, podíamos declarar:

- *funciones*, que siempre retornan algo y no producen ningún efecto
- *procedimientos*, que no retornan nada, y producen efectos

Entonces `versosMartinFierro`, no es una función... ¡sino un procedimiento! ¿Cómo se declaran procedimientos en JavaScript?

¡De la misma forma que las funciones!: usando la palabra clave `function`.

```
function versosMartinFierro() {  
  imprimir("Aquí me pongo a cantar");  
  imprimir("Al compás de la vigüela");  
  imprimir("Que el hombre que lo desvela");  
  imprimir("Una pena extraordinaria");  
}
```

Envía esta nueva versión de `versosMartinFierro`

 Solución

 Consola

```
1 function versosMartinFierro() {  
2   imprimir("Aquí me pongo a cantar");  
3   imprimir("Al compás de la vigüela");  
4   imprimir("Que el hombre que lo desvela");  
5   imprimir("Una pena extraordinaria");  
6 }
```

 Enviar

✓ ¡Muy bien! Tu solución pasó todas las pruebas

Esto puede ser un poco perturbador : JavaScript no diferencia funciones de procedimientos: todos pueden tener efectos y todos pueden o no tener retorno.

Vos sos responsable de escribir una `function` que tenga sentido y se comporte o bien como un procedimiento (sin retorno y con efecto) o bien como una función (con retorno y sin efecto).

Si empezás a mezclar funciones con retornos y efecto, funcionará, pero tu código se volverá de a poco más difícil de entender. Esto nos va a pasar mucho en JavaScript: que puedas hacer algo no significa que debas hacerlo .

Esta guía fue desarrollada por Franco Bulgarelli bajo los términos de la [Licencia Creative Commons Compartir-Igual, 4.0](#).

© 2015-2022 Ikumi SRL

[Información importante](#)

[Términos y Condiciones](#)

[Reglas del Espacio de Consultas](#)







# ¿Y el program?

Ahora bien, más allá de que podamos consultar el resultado de una función a través de la **consola**, también aprendimos anteriormente que los programas tienen un punto de entrada: el **program**.  
¿Dónde quedó?

La respuesta es tan simple como sorprendente: en JavaScript todo lo que escribamos fuera de una **function** será, implícitamente, dicho punto de entrada. Por ejemplo, si queremos un programa que imprime por pantalla el clásico **"Hola, mundo!"**, lo podremos escribir así:

```
imprimir("Hola, mundo!");
```

O si queremos un programa que tire tres veces los dados e imprima sus resultados, podemos escribirlo así:

```
imprimir("Tirando dados");  
imprimir("La primera tirada dio " + tirarDado());  
imprimir("La segunda tirada dio " + tirarDado());  
imprimir("La tercera tirada dio " + tirarDado());
```

Copí y enviá este programa

 **Solución**  **Consola**

```
1 imprimir("Tirando dados");  
2 imprimir("La primera tirada dio " + tirarDado());  
3 imprimir("La segunda tirada dio " + tirarDado());  
4 imprimir("La tercera tirada dio " + tirarDado());
```

 Enviar

✓ ¡Muy bien! Tu solución pasó todas las pruebas

¿Ooooups, y el resultado? ¿Dónde está lo que imprimimos por pantalla? ¿Es que nuestro programa no anduvo?

No, para nada, es que simplemente no te estamos mostrando lo que sale por pantalla .

¿Por qué? ¿Porque somos malvados? Bueno, quizás en parte , pero tenemos además una buena razón: cuando escribís programas reales, es muy, **muy** frecuente que no sea fácil ver lo que el `imprimir` imprime, por decenas de motivos. Entonces, como rara vez vas poder ver *a tiempo* lo que se imprime en la pantalla, terminan siendo una técnica poco útil.

Moraleja: en los ejercicios que quedan, **no uses** `imprimir` salvo que te lo pidamos explícitamente.

¡Nos vemos en el próximo ejercicio!

Esta guía fue desarrollada por Franco Bulgarelli bajo los términos de la [Licencia Creative Commons Compartir-Igual, 4.0](#).

© 2015-2022 Ikumi SRL

[Información importante](#)

[Términos y Condiciones](#)

[Reglas del Espacio de Consultas](#)





# Coerciones

Volvamos un momento al código anterior. ¿Notás algo extraño en esta expresión?

6. Coerciones

```
"La primera tirada dio " + primeraTirada
```

Utilizamos el operador `+` de una forma diferente, operando un string y un número, y lo que hizo fue concatenar al string con la representación textual del número. Es decir que:

- si operamos dos números con `+`, se suman
- si operamos dos strings con `+`, se concatenan
- si operamos un string y un número `+`, se *convierte implícitamente* el número a string, y luego se concatenan, al igual que antes

En JavaScript, estas conversiones implícitas, también llamadas *coerciones*, ocurren mucho.

¡Quizás incluso *más de lo que nos gustaría!*

Veamos si queda claro, definí una función `elefantesEquilibristas`, que tome un número de elefantes y **devuelva** una rima de una conocida canción:

```
> elefantesEquilibristas(3)
"3 elefantes se balanceaban"
> elefantesEquilibristas(462)
"462 elefantes se balanceaban"
```

Solución Consola

```
1 function elefantesEquilibristas(num) {
2   return num+" elefantes se balanceaban";
3 }
```



✓ ¡Muy bien! Tu solución pasó todas las pruebas

Esta guía fue desarrollada por Franco Bulgarelli bajo los términos de la [Licencia Creative Commons Compartir-Igual, 4.0](#).

© 2015-2022 Ikumi SRL

[Información importante](#)

[Términos y Condiciones](#)

[Reglas del Espacio de Consultas](#)





# El círculo de la vida

En programación buscamos que resolver nuestros problemas usando... programas. Y entre los problemas que casi nadie quiere resolver están los matemáticos. Sobre todo aquellos que aparecen números como pi con infinitos decimales imposibles de recordar.

Considerando al número pi igual a `3.14159265358979` (no es infinito pero lo suficientemente preciso para nuestros cálculos):

Definí las funciones `perimetroCirculo` y `areaCirculo` que reciben el radio de un círculo y a partir del mismo nos retornan su perímetro y su área.

💡 ¡Dame una pista!

🔍 Solución

>\_ Consola

```
1 function areaCirculo(rad) {  
2     return 3.14159265358979*rad*rad;  
3 }  
4  
5 function perimetroCirculo(rad) {  
6     return 3.14159265358979*rad*2;  
7 }
```

▶ Enviar

✅ ¡Muy bien! Tu solución pasó todas las pruebas

Excelente, la precisión de nuestros cálculos es innegable , pero tuvimos que escribir un número larguísimo. Pensemos que pi aparece en un montón de fórmulas matemáticas. ¿Es necesario escribir este número cada vez?¿No podemos hacer algo más cómodo?

---

Esta guía fue desarrollada por Franco Bulgarelli bajo los términos de la [Licencia Creative Commons Compartir-Igual, 4.0](#).

© 2015-2022 Ikumi SRL

[Información importante](#)

[Términos y Condiciones](#)

[Reglas del Espacio de Consultas](#)





# Plenso que así es más fácil

Por suerte existe una herramienta que va a simplificar nuestra tarea de ahora en adelante: las *variables*.

8. Plenso que así es más f.

Las variables nos permiten nombrar y reutilizar *valores*. Similar a cómo los procedimientos y funciones nos permiten dar nombres y reutilizar soluciones a problemas más pequeños. Por ejemplo, si hacemos...

```
let primerMes = "enero"
```

...estamos *asignándole* el valor "enero" a la variable `primerMes`. En criollo, estamos dándole ese valor a la variable.

Cambiá los lugares donde aparece `3.14159265358979` por la variable `pi` en las funciones que tenemos definidas.

Solución Consola

```
1 let pi = 3.14159265358979;
2
3 function areaCirculo(rad) {
4   return pi*rad*rad;
5 }
6
7 function perimetroCirculo(rad) {
8   return pi*rad*2;
9 }
```

Enviar

¡Muy bien! Tu solución pasó todas las pruebas

¡Excelente! Gracias a la variable `pi` no tuvimos que escribir el número cada vez que teníamos que usarlo y ¡nuestro programa quedó mucho más entendible!

Esta guía fue desarrollada por Franco Bulgarelli bajo los términos de la [Licencia Creative Commons Compartir-Igual, 4.0](#).

© 2015-2022 Ikumi SRL

[Información importante](#)

[Términos y Condiciones](#)

[Reglas del Espacio de Consultas](#)





# Esto no tiene valor

Ya que vas entendiendo cómo se **asignan** las variables, te traemos algo para pensar: ¿qué pasa si intento **usar** una variable a la que nunca le asigné un valor?

Tenemos esta función definida:

```
function sumaSinSentido() {  
  return numero + 8;  
}
```

Probala en la consola y fijate qué sucede.

>



Esta guía fue desarrollada por Franco Bulgarelli bajo los términos de la [Licencia Creative Commons Compartir-Igual, 4.0](#).

© 2015-2022 Ikumi SRL

[Información importante](#)

[Términos y Condiciones](#)

[Reglas del Espacio de Consultas](#)







# Variables globales

Entonces, ¿es necesario darle valor a nuestras variables antes de usarlas?

¡Sí! Cuando declarás una variable tenés que darle un valor inicial, lo cual se conoce como *inicializar* la variable.

¡Y sorpresa! Podemos declarar variables tanto directamente en el programa, como dentro de una `function`:

```
function cuentaExtravagante(unNumero) {  
  let elDoble = unNumero * 2;  
  if (elDoble > 10) {  
    return elDoble;  
  } else {  
    return 0;  
  }  
}
```

Las variables declaradas dentro de una `function`, conocidas como *variables locales*, no presentan mayor misterio. Sin embargo, hay que tener un particular cuidado: sólo se pueden utilizar desde dentro de la `function` en cuestión. Si quiero referenciarla desde un programa:

```
let elCuadruple = elDoble * 4;
```

Kaboom, ¡se romperá!

Sin embargo, las variables declaradas directamente en el programa, conocidas como *variables globales*, pueden ser utilizadas desde cualquier `function`. Por ejemplo:

```
let pesoMaximoEquipajeEnGramos = 5000;  
  
function puedeLlevar(pesoEquipaje) {  
  return pesoEquipaje <= pesoMaximoEquipajeEnGramos;  
}
```

Veamos si queda claro: definí una función `ascensorSobrecargado`, que toma una cantidad de personas y retorna si entre todas superan la carga máxima de 300 kg.

Tené en cuenta que nuestra función va a utilizar dos variables globales:

- `pesoPromedioPersonaEnKilogramos`, la cual ya está declarada,
- `cargaMaximaEnKilogramos` que vas a tener que declarar.

Solución Consola

```
1 let cargaMaximaEnKilogramos = 300  
2 function ascensorSobrecargado(personas) {  
3   return pesoPromedioPersonaEnKilogramos * personas > cargaMaximaEnKilogramos;  
4 }
```

Enviar

✓ ¡Muy bien! Tu solución pasó todas las pruebas

Esta guía fue desarrollada por Franco Bulgarelli bajo los términos de la [Licencia Creative Commons Compartir-Igual, 4.0](#).

© 2015-2022 Ikumi SRL

[Información importante](#)

[Términos y Condiciones](#)

[Reglas del Espacio de Consultas](#)





# La buena fortuna

Las variables no serían tan interesantes si no se pudieran modificar. Afortunadamente, JavaScript nos da nuevamente el gusto y nos lo permite:

```
function pasarUnDiaNormal() {  
  diasSinAccidentesConVelociraptores = diasSinAccidentesConVelociraptores + 1  
}  
  
function tenerAccidenteConVelociraptores() {  
  diasSinAccidentesConVelociraptores = 0;  
}
```

¡Ahora vamos a hacer algo de dinero !

Definí el procedimiento `aumentarFortuna` que duplique el valor de la variable global `pesosEnMiBilletera` . No declares la variable, ya lo hicimos por vos (con una cantidad secreta de dinero) .

💡 ¡Dame una pista!

☒ Solución [>\\_ Consola](#)

```
1 function aumentarFortuna() {  
2   pesosEnMiBilletera = pesosEnMiBilletera * 2  
3 }
```

▶ Enviar

✅ ¡Muy bien! Tu solución pasó todas las pruebas

Actualizaciones como duplicar, triplicar, incrementar en uno o en una cierta cantidad son tan comunes que JavaScript presenta algunos atajos:

```
x += y; //equivalente a x = x + y;  
x *= y; //equivalente a x = x * y;  
x -= y; //equivalente a x = x - y;  
x++;   //equivalente a x = x + 1;
```

¡Usalos cuando quieras!

información importante

Términos y Condiciones

Reglas del Espacio de Consultas





# ¿Y esto cuánto vale?

Vimos que una variable solo puede tener un valor, entonces cada vez que le asignamos uno nuevo, perdemos el anterior. Entonces, dada la función:

```
function cuentaExtravagante() {  
  let numero = 8;  
  numero *= 2;  
  numero += 4;  
  return numero;  
}
```

Si la invocamos ¿qué crees que retornaría?

Veamos el paso a paso:

- inicialmente la variable `numero` vale `8` ;
- al hacer `numero *= 2` la variable pasa a tener su valor multiplicado por 2, es decir, `16` ;
- al hacer `numero += 4` le sumamos 4 a 16 y lo guardamos en `numero`, por ende la función `cuentaExtravagante` retorna `20` .

Esta guía fue desarrollada por Franco Bulgarelli bajo los términos de la [Licencia Creative Commons Compartir-Igual, 4.0](#).

© 2015-2022 Ikumi SRL

[Información importante](#)

[Términos y Condiciones](#)

[Reglas del Espacio de Consultas](#)

