< / > CLUB PROGRAMADOR

COLECCIÓN DE DESARROLLO WEB

150 DESAFÍOS

DE PROGRAMACIÓN WEB

JavaScript

De los primeros programas a las aplicaciones interactivas

---

clubprogramador.com / desarrollo web

JAVASCRIPT / COMPORTAMIENTO

```
const nombre = "Ana";
console.log(`¡Hola, ${nombre}!`);
```

SALIDA DEL PROGRAMA

¡Hola, Ana!

DATOS + LÓGICA + INTERACCIÓN

PRIMERA EDICIÓN / 2026

150 ejercicios resueltos / Código fuente y capturas reales

MUESTRA GRATUITA / 10 PRIMEROS DESAFIOS

Créditos editoriales

150 desafíos de programación web con JavaScript

Autor y editor: Club Programador

Sitio web: https://clubprogramador.com

Contacto: info@clubprogramador.com

Primera edición digital / noviembre de 2026

Formato: PDF + archivos HTML, CSS y JavaScript

Idioma: español

Contenido de esta edición

Esta edición reúne 150 desafíos en ocho módulos, desde los fundamentos de JavaScript hasta veinte proyectos integradores. Cada ejercicio incluye enunciado, solución, explicación, captura real, comprobaciones, un error frecuente y una propuesta de ampliación. El paquete de práctica acompaña al PDF e incluye los recursos y el servidor local.

Uso del material

© 2026 Club Programador. Material original para estudio personal. No se autoriza su redistribución, reventa ni publicación, sin perjuicio de las excepciones aplicables. Para comunicar una errata, indicá el número de ejercicio, navegador y resultado observado.

Referencias y edición independiente

La explicación se apoya en la documentación de MDN Web Docs. Esta publicación educativa es independiente de los fabricantes de navegadores. Los ejemplos usan datos ficticios.

Tipografías: DejaVu Sans, DejaVu Serif y DejaVu Sans Mono. Las capturas muestran los ejercicios ejecutados en el navegador. Los gráficos SVG incluidos son recursos de práctica.

Antes de empezar

Este libro continúa el recorrido del libro de HTML y CSS de Club Programador. JavaScript se enseña desde cero: empezamos con datos, operaciones y programas breves, y después los conectamos con una página web.

> **Antes de empezar: HTML y CSS básicos**
>
> Se dan por conocidos los conceptos del libro anterior: etiquetas y atributos HTML, id y class, enlaces, imágenes, listas y formularios; y, en CSS, selectores, clases, propiedades y valores. También necesitás crear archivos, relacionar un HTML con su hoja de estilos y abrirlo en el navegador. Si alguno de estos temas te resulta nuevo, estudiá o repasá primero esas secciones del libro de HTML y CSS. No hace falta memorizar todas las etiquetas ni propiedades.

Cómo trabajar con cada desafío

01. Leé el problema e intentá resolverlo antes de consultar el código.

02. Abrí script.js en tu editor y conservá una copia de la solución.

03. Ejecutá index.html y compará el resultado con la captura.

04. Modificá los datos, guardá y recargá con Ctrl+R.

05. Probá los casos indicados y resolvé el desafío adicional.

> **Tres tecnologías, tres responsabilidades**
>
> HTML estructura el contenido, CSS define su presentación y JavaScript agrega comportamiento. En el primer módulo nos concentramos en el lenguaje. El acceso al DOM se explica desde el módulo 05.

> **Acerca de las capturas**
>
> Los resultados se capturan ejecutando los archivos del ejercicio en Microsoft Edge. El panel de salida pertenece al material de apoyo: reproduce console.log sin exigir conocimientos de DOM antes de tiempo. La consola del navegador sigue disponible.

Cómo ejecutar los ejercicios

01. Abrí la carpeta ejercicios y elegí Ejercicio_01.

02. Hacé doble clic en index.html para ejecutar el programa.

03. Editá script.js, guardá y recargá la página.

04. Abrí F12 y la pestaña Consola para revisar mensajes y errores.

Qué contiene cada carpeta

script.js: solución JavaScript que se reproduce en el libro.

index.html: HTML base y carga de los scripts con defer.

apoyo.js: copia la salida de la consola al panel visible.

estilos.css: presentación del material de apoyo.

LEEME.txt: guía de práctica, explicación y comprobaciones.

> **Archivos locales y servidor**
>
> Los ejercicios 01 a 110 funcionan al abrir index.html directamente. Para 111 a 150, iniciá el servidor local incluido siguiendo la preparación del módulo 07. Los datos de práctica y los módulos ES se sirven por HTTP, sin servicios pagos ni APIs externas.

Documentación consultada

MDN: gramática y tipos

MDN: expresiones y operadores

MDN: control de flujo

MDN: bucles e iteración

MDN: funciones

MDN: números y textos

MDN: colecciones indexadas

MDN: objetos

Ruta de aprendizaje

MÓDULO 01 / Fundamentos de JavaScript

Ejercicios 01-15 / Desarrollado

MÓDULO 02 / Decisiones y repeticiones

Ejercicios 16-30 / Desarrollado

MÓDULO 03 / Funciones y resolución de problemas

Ejercicios 31-45 / Desarrollado

MÓDULO 04 / Textos, arrays y objetos

Ejercicios 46-60 / Desarrollado

MÓDULO 05 / Manipulación del DOM

Ejercicios 61-85 / Desarrollado

MÓDULO 06 / Eventos y formularios

Ejercicios 86-110 / Desarrollado

MÓDULO 07 / Persistencia y programación asíncrona

Ejercicios 111-130 / Desarrollado

MÓDULO 08 / Proyectos integradores

Ejercicios 131-150 / Desarrollado

Cada módulo enlaza a su índice. Los ocho módulos incluyen los ejercicios completos.

Fundamentos de JavaScript

01. Mi primer programa

02. Guardar datos con const y let

03. Reconocer tipos de datos

04. Crear mensajes con plantillas

05. Operaciones y prioridades

06. Calcular un descuento

07. Convertir texto en números

08. Comparar valores y tipos

09. Combinar condiciones lógicas

10. Repartir unidades y calcular el resto

11. Convertir temperaturas

12. Calcular un promedio

13. Actualizar un saldo

14. Presentar importes con decimales

15. Construir un resumen de compra

> **Navegación del módulo**
>
> Seleccioná un título para ir al ejercicio.

Decisiones y repeticiones

16. Decidir con if y else

17. Clasificar una calificación

18. Elegir con switch

19. Usar el operador condicional

20. Detectar entradas inválidas

21. Contar con for

22. Repetir con while

23. Validar con do while

24. Sumar una secuencia

25. Crear una tabla de multiplicar

26. Contar pares e impares

27. Buscar divisores

28. Detectar un número primo

29. Encontrar un valor con break

30. Construir un juego de intentos

> **Navegación del módulo**
>
> Seleccioná un título para ir al ejercicio.

Funciones y resolución de problemas

31. Declarar y llamar una función

32. Devolver un resultado

33. Definir parámetros predeterminados

34. Escribir funciones flecha

35. Comprender el alcance de las variables

36. Separar cálculo y presentación

37. Validar parámetros

38. Calcular con funciones pequeñas

39. Pasar una función como argumento

40. Crear un contador con cierre

41. Trabajar con parámetros rest

42. Diseñar funciones sin efectos externos

43. Descomponer un problema

44. Comprobar casos límite

45. Construir una calculadora reutilizable

> **Navegación del módulo**
>
> Seleccioná un título para ir al ejercicio.

Textos, arrays y objetos

46. Normalizar un texto

47. Buscar y reemplazar palabras

48. Separar y unir cadenas

49. Crear y recorrer un array

50. Agregar y quitar elementos

51. Transformar datos con map

52. Filtrar resultados con filter

53. Buscar con find y some

54. Acumular valores con reduce

55. Ordenar sin perder el original

56. Modelar un objeto

57. Desestructurar datos

58. Copiar con spread y conocer sus límites

59. Agrupar información de productos

60. Construir un informe de ventas

> **Navegación del módulo**
>
> Seleccioná un título para ir al ejercicio.

Manipulación del DOM

61. Seleccionar por identificador

62. Seleccionar con querySelector

63. Recorrer varias coincidencias

64. Actualizar texto con textContent

65. Insertar HTML de confianza

66. Agregar y quitar clases

67. Alternar clases según un estado

68. Modificar estilos y variables CSS

69. Actualizar atributos de una imagen

70. Leer y escribir datos con dataset

71. Mostrar y ocultar elementos

72. Leer y modificar campos de texto

73. Cambiar casillas y opciones

74. Recorrer padres e hijos

75. Recorrer elementos hermanos

> **Navegación del módulo**
>
> Seleccioná un título para ir al ejercicio.

Manipulación del DOM / continuación

76. Localizar un contenedor con closest

77. Crear y anexar elementos

78. Insertar antes y después

79. Construir con DocumentFragment

80. Eliminar elementos seleccionados

81. Reemplazar un elemento

82. Clonar una plantilla HTML

83. Renderizar listas desde datos

84. Construir una tabla de datos

85. Crear una tarjeta de producto

> **Navegación del módulo**
>
> Seleccioná un título para ir al ejercicio.

Eventos y formularios

86. Responder a un clic

87. Escuchar cambios con input

88. Enviar un formulario con submit

89. Validar campos y mostrar mensajes

90. Manejar eventos de teclado

91. Delegar eventos en una lista

92. Crear un contador interactivo

93. Filtrar una lista en tiempo real

94. Calcular un presupuesto

95. Construir un formulario de inscripción

96. Abrir y cerrar un menú accesible

97. Crear pestañas con teclado

98. Mostrar una ventana modal

99. Validar campos relacionados

100. Editar elementos de una lista

> **Navegación del módulo**
>
> Seleccioná un título para ir al ejercicio.

Eventos y formularios / continuación

101. Agregar y quitar filas de una tabla

102. Seleccionar elementos en grupo

103. Ordenar una tabla desde sus encabezados

104. Crear un acordeón

105. Mostrar un contador de caracteres

106. Previsualizar una imagen seleccionada

107. Validar y mostrar errores por campo

108. Controlar envío y reinicio

109. Sincronizar filtros y resultados

110. Construir un gestor de elementos HTML

> **Navegación del módulo**
>
> Seleccioná un título para ir al ejercicio.

Persistencia y programación asíncrona

111. Convertir datos con JSON

112. Guardar preferencias en localStorage

113. Recuperar datos y manejar fallos

114. Programar una tarea con setTimeout

115. Crear y detener un intervalo

116. Comprender una promesa

117. Esperar tareas con async y await

118. Leer un JSON local con fetch

119. Mostrar estados de carga y error

120. Separar responsabilidades con módulos ES

121. Versionar datos guardados

122. Persistir una colección de objetos

123. Sincronizar cambios entre pestañas

124. Encadenar tareas asíncronas

125. Resolver tareas en paralelo

> **Navegación del módulo**
>
> Seleccioná un título para ir al ejercicio.

Persistencia y programación asíncrona / continuación

126. Cancelar una petición con AbortController

127. Distinguir errores HTTP y de red

128. Evitar respuestas desactualizadas

129. Cargar y filtrar datos remotos

130. Construir un explorador de datos con caché local

Navegación del módulo

Seleccioná un título para ir al ejercicio.

Proyectos integradores

131. Lista de tareas con persistencia

132. Cuestionario con puntaje

133. Conversor de unidades accesible

134. Control de gastos personales

135. Buscador de productos

136. Carrito de compras

137. Agenda de contactos

138. Juego de memoria

139. Panel de datos desde un JSON local

140. Aplicación de hábitos con resumen semanal

141. Tablero de tareas por estado

142. Planificador de estudio

143. Gestor de inventario

144. Editor de fichas de productos

145. Comparador de presupuestos

> **Navegación del módulo**
>
> Seleccioná un título para ir al ejercicio.

Proyectos integradores / continuación

146. Tabla de datos con paginación

147. Galería filtrable con vista ampliada

148. Formulario de varias etapas

149. Panel personal configurable

150. Aplicación integradora de gestión de cursos

> **Navegación del módulo**
>
> Seleccioná un título para ir al ejercicio.

Fundamentos de JavaScript

Ejercicios 01 a 15. Aprendé a representar datos y a transformar valores antes de conectar la lógica con botones y formularios. Los programas usan datos definidos en el código para que puedas reproducir cada resultado.

- Declarar variables y distinguir valores de diferentes tipos.
- Crear mensajes con plantillas y observarlos con console.log.
- Aplicar operaciones, conversiones y comparaciones explícitas.
- Combinar resultados en cálculos con significado.
- Verificar salidas y modificar un dato por vez.

> **La solución y el soporte**
>
> El libro imprime script.js. index.html, estilos.css y apoyo.js ya están incluidos: no hace falta copiarlos para cada ejercicio. apoyo.js usa funciones, arrays y DOM, que se explicarán en módulos posteriores. Podés ejecutar script.js en la consola sin ese soporte.

> **Criterio de comprobación**
>
> Cada salida inicial se contrasta automáticamente con valores esperados, y se ejecutan variantes de datos para comprobar fórmulas y límites. La captura muestra el ejemplo inicial. Usá los casos de la sección Comprobación para practicar los cambios.

01. Mi primer programa

El problema

Mostrá un mensaje de bienvenida y el nombre del curso. Ejecutá el programa, cambiá el mensaje y comprobá el resultado al recargar.

Solución JavaScript / script.js

```
01  // Cada llamada escribe una línea en la consola.
02  console.log("¡Hola, mundo!");
03  console.log("Estoy aprendiendo JavaScript.");
```

Explicación

console.log permite observar valores durante la ejecución. Los textos se escriben entre comillas; los paréntesis contienen el valor que recibe la llamada. Las líneas que comienzan con // son comentarios: explican el código y no se ejecutan. En este módulo, la página también copia la salida a un panel para que puedas compararla con el libro. Ese panel pertenece al material de apoyo.

Resultado y comprobación

Resultado visual

CLUB PROGRAMADOR / JAVASCRIPT

01. Mi primer programa

Mostrá un mensaje de bienvenida y el nombre del curso. Ejecutá el programa, cambiá el mensaje y comprobá el resultado al recargar.

Salida del programa / script.js

```
¡Hola, mundo!
Estoy aprendiendo JavaScript.
```

Editá `script.js`, guardá y recargá con Ctrl+R. También podés consultar la consola del navegador con F12.

El panel y sus estilos son material de apoyo. La solución está en `script.js` y en los archivos complementarios indicados en el libro.

Comprobación

Abrí index.html y verificá las dos líneas en el mismo orden. Abrí también la consola del navegador con F12. Cambiá mundo por tu nombre en script.js, guardá y recargá: ambas vistas deben reflejar el cambio.

> **Error frecuente**
>
> Olvidar cerrar una comilla o un paréntesis produce un error de sintaxis y evita que este archivo se ejecute. Revisá el mensaje de la consola y el número de línea.

> **Desafío adicional**
>
> Agregá una tercera línea con tu objetivo de aprendizaje y volvé a ejecutar.

02. Guardar datos con const y let

El problema

Guardá el nombre de una estudiante y la cantidad de ejercicios resueltos. Aumentá esa cantidad en uno y mostrá ambos datos.

Solución JavaScript / script.js

```
01  const nombre = "Ana";
02  let resueltos = 2;
03  resueltos = resueltos + 1;
04  console.log(nombre);
05  console.log(resueltos);
```

Explicación

const declara una vinculación que no se puede reasignar; let permite asignarle un valor nuevo a la variable. El nombre permanece igual, mientras resueltos pasa de 2 a 3. A la derecha de = se calcula el nuevo valor antes de guardarlo a la izquierda. Usá nombres descriptivos y preferí const cuando no necesites reasignar. const no vuelve inmutable el contenido de objetos o arrays; eso se estudiará más adelante.

Resultado y comprobación

Resultado visual



Comprobación

La primera línea debe ser Ana y la segunda 3. Cambiá el valor inicial de resueltos a 7: al recargar, el resultado debe ser 8. Restaurá el ejemplo antes de compararlo con la captura.

> **Error frecuente**
>
> Declarar resueltos con const y luego intentar reasignarlo genera un error en tiempo de ejecución. Tampoco podés declarar dos veces el mismo nombre con let en el mismo ámbito.

> **Desafío adicional**
>
> Guardá también la cantidad total de ejercicios y calculá cuántos faltan.

03. Reconocer tipos de datos

El problema

Inspeccioná los tipos de un nombre, una edad, una condición y un dato todavía sin asignar.

Solución JavaScript / script.js

```
const nombre = "Ana";
const edad = 23;
const inscrita = true;
let comentario;
console.log(typeof nombre);
console.log(typeof edad);
console.log(typeof inscrita);
console.log(typeof comentario);
```

Explicación

typeof devuelve un texto que identifica el tipo de un valor. En este ejemplo obtenemos string, number, boolean y undefined. Una variable declarada sin valor comienza con undefined. Los números se escriben sin comillas y los booleanos son true o false. No se enumeran aquí todos los tipos del lenguaje. Un caso histórico importante es typeof null: devuelve object, aunque null representa la ausencia intencional de un valor.

Resultado y comprobación

Resultado visual

CLUB PROGRAMADOR / JAVASCRIPT

03. Reconocer tipos de datos

Inspeccioná los tipos de un nombre, una edad, una condición y un dato todavía sin asignar.

Salida del programa / script.js

```
string
number
boolean
undefined
```

Editá `script.js`, guardá y recargá con Ctrl+R. También podés consultar la consola del navegador con F12.

El panel y sus estilos son material de apoyo. La solución está en `script.js` y en los archivos complementarios indicados en el libro.

Comprobación

Compará las cuatro líneas con la salida esperada. Poné la edad entre comillas: su tipo debe cambiar a string. Luego probá typeof null en la consola y verificá el caso explicado.

> **Error frecuente**
>
> Confundir el contenido de un texto con su tipo: "23" contiene dígitos, pero sigue siendo un string.

> **Desafío adicional**
>
> Asigná un texto a comentario y comprobá cómo cambia el resultado de typeof.

04. Crear mensajes con plantillas

El problema

Armá un saludo que incorpore un nombre y la cantidad de ejercicios completados sin concatenar varios fragmentos.

Solución JavaScript / script.js

```
01  const nombre = "Ana";
02  const resueltos = 3;
03  const mensaje = `${nombre} completó ${resueltos} ejercicios.`;
04  console.log(mensaje);
```

Explicación

Una plantilla literal se delimita con acentos graves, también llamados backticks. Dentro de ${...}, JavaScript evalúa una expresión e incorpora su valor al texto. La variable mensaje guarda el resultado completo. También podés incluir cálculos, pero conviene separar las operaciones complejas para que el programa sea fácil de leer.

Resultado y comprobación

Resultado visual



Comprobación

Cambiá nombre a Leo y resueltos a 8. La salida debe decir Leo completó 8 ejercicios. Restaurá los valores para recuperar la captura del libro.

> **Error frecuente**
>
> Usar comillas simples o dobles en lugar de backticks: en ese caso ${nombre} se muestra literalmente y no se reemplaza.

> **Desafío adicional**
>
> Agregá el total de desafíos y mostrale a la estudiante cuántos le quedan.

05. Operaciones y prioridades

El problema

Calculá el costo de tres cuadernos de 1200 y dos lápices de 300. Compará el efecto de agrupar una suma entre paréntesis.

Solución JavaScript / script.js

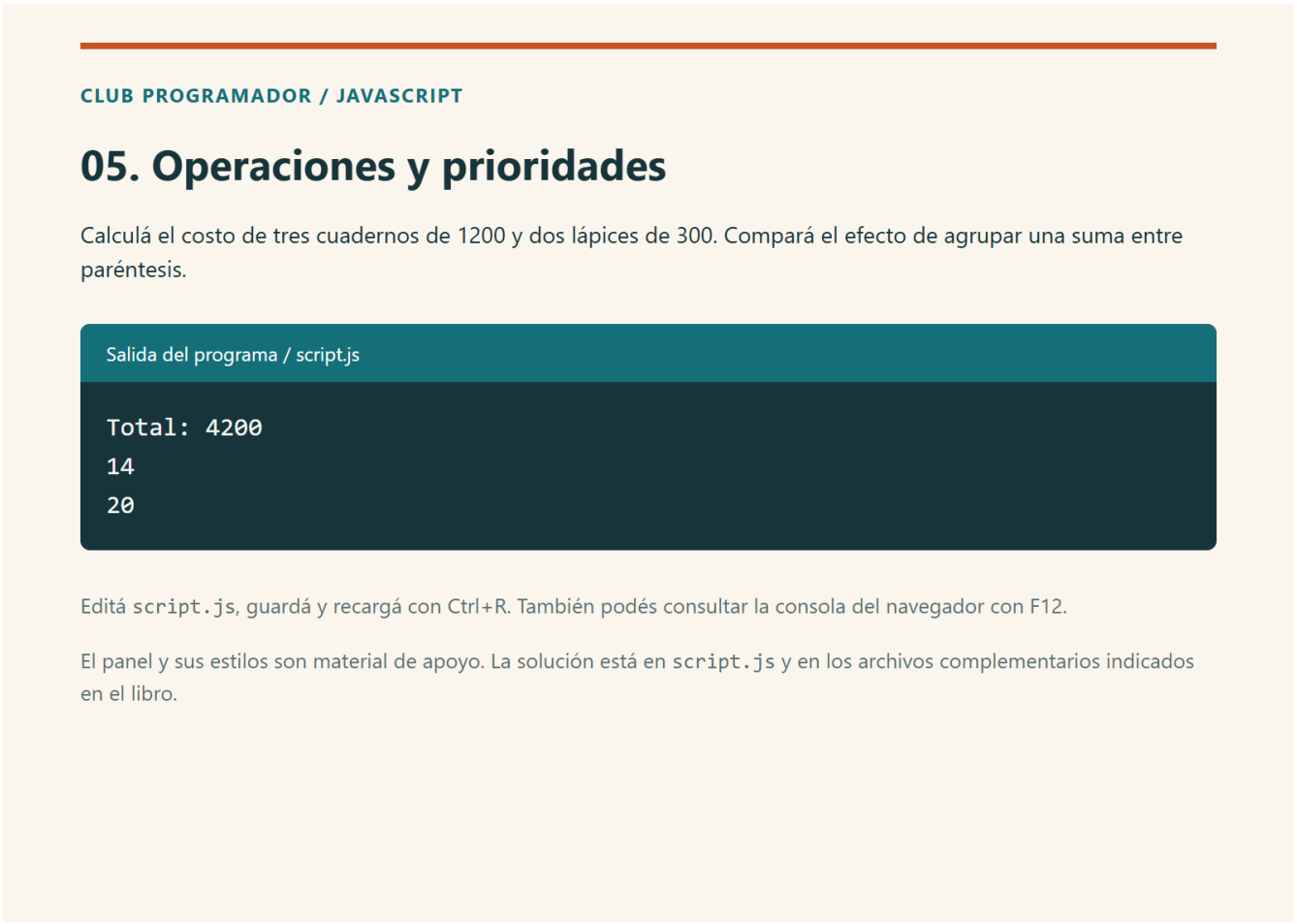
```
const cuadernos = 3 * 1200;
const lapices = 2 * 300;
const total = cuadernos + lapices;
console.log(`Total: ${total}`);
console.log(2 + 3 * 4);
console.log((2 + 3) * 4);
```

Explicación

Los operadores * y / tienen prioridad sobre + y -. Los paréntesis permiten cambiar el orden o hacerlo explícito. El total suma dos subtotales calculados por separado. Las dos expresiones finales usan los mismos números: la primera multiplica antes de sumar; la segunda suma antes de multiplicar. = asigna un valor: no expresa una comparación.

Resultado y comprobación

Resultado visual



Comprobación

Verificá las tres líneas. Cambiá la cantidad de lápices a 4: el total debe pasar a 4800. Calculá a mano ambas expresiones antes de ejecutar.

> **Error frecuente**
>
> Escribir una fórmula como si JavaScript la leyera de izquierda a derecha sin prioridades. Agregá paréntesis cuando la intención pueda resultar confusa.

> **Desafío adicional**
>
> Sumá un costo de envío de 500 y mostrale al usuario los subtotales por separado.

06. Calcular un descuento

El problema

Calculá el ahorro y el precio final de un producto de 25000 con un descuento del 15 %. Aplicá el porcentaje una sola vez.

Solución JavaScript / script.js

```
01  const precio = 25000;
02  const porcentaje = 15;
03  const ahorro = precio * porcentaje / 100;
04  const final = precio - ahorro;
05  console.log(`Ahorro: ${ahorro}`);
06  console.log(`Precio final: ${final}`);
```

Explicación

Un porcentaje indica una cantidad por cada cien unidades. precio * porcentaje / 100 convierte el 15 % en el importe que se descuenta. Luego restamos ese ahorro al precio original. Los valores de este ejercicio son datos conocidos y válidos; la validación de porcentajes fuera del rango 0 a 100 se trabajará con condicionales.

Resultado y comprobación

Resultado visual



Comprobación

Con 15 %, verificá ahorro 3750 y precio final 21250. Probá 0 % y 100 %: el precio final debe ser 25000 y 0, respectivamente.

> **Error frecuente**
>
> Restar 15 directamente al precio o multiplicar por 15 sin dividir por 100. El porcentaje y el importe son magnitudes distintas.

> **Desafío adicional**
>
> Calculá el precio final de dos productos con el mismo porcentaje y compará sus ahorros.

07. Convertir texto en números

El problema

Sumá dos cantidades recibidas como texto. Mostrá también qué ocurre si sumás las cadenas sin convertirlas.

Solución JavaScript / script.js

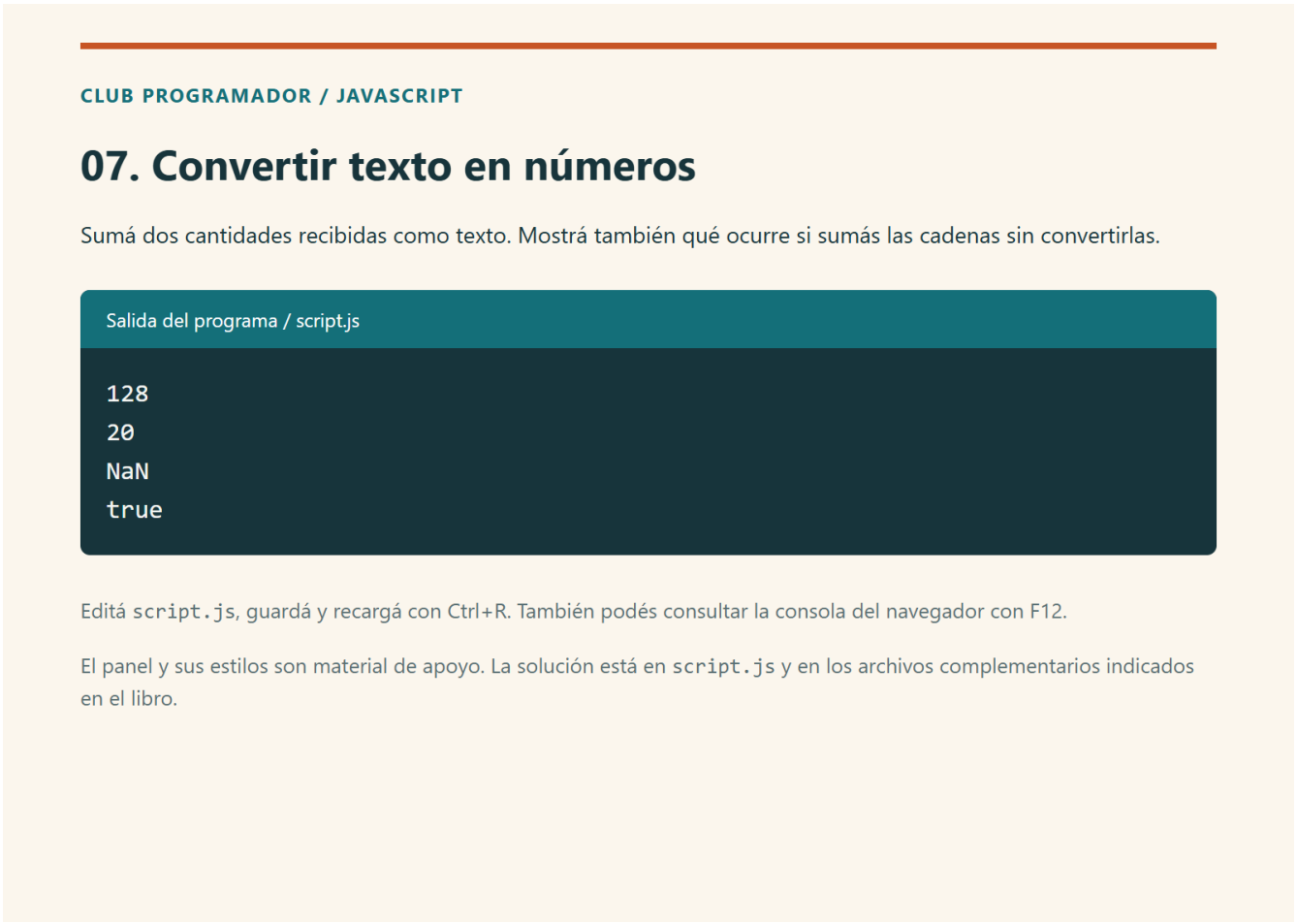
```
const entradaA = "12";
const entradaB = "8";
console.log(entradaA + entradaB);
const a = Number(entradaA);
const b = Number(entradaB);
console.log(a + b);
console.log(Number("doce"));
console.log(Number.isNaN(Number("doce")));
```

Explicación

Cuando + recibe cadenas, puede concatenarlas: "12" + "8" produce "128". Number intenta convertir el texto completo en un número; con los datos válidos obtenemos 20 al sumar. Una conversión fallida produce NaN. Number.isNaN permite detectarlo. Atención: Number("") y Number(" ") producen 0, por lo que una futura validación de formularios deberá distinguir campos vacíos de un cero escrito por el usuario.

Resultado y comprobación

Resultado visual



Comprobación

Verificá las cuatro líneas. Probá "12.5" y "8": la suma numérica debe ser 20.5. Luego probá "12,5": Number devuelve NaN porque la coma no es el separador decimal de esta sintaxis.

> **Error frecuente**
>
> Suponer que cualquier texto con dígitos se suma como número o que NaN se detecta comparándolo con === NaN. Usá Number.isNaN.

> **Desafío adicional**
>
> Probá Number("") y Number(" ") y escribí por qué hará falta validar las entradas.

08. Comparar valores y tipos

El problema

Compará una edad numérica con la misma edad escrita como texto. Observá la diferencia entre igualdad estricta e igualdad con conversión.

Solución JavaScript / script.js

```
const edad = 18;
const entrada = "18";
console.log(edad === entrada);
console.log(edad == entrada);
console.log(edad === Number(entrada));
console.log(edad >= 18);
console.log(edad !== 21);
```

Explicación

=== compara sin convertir tipos: 18 y "18" son distintos. == aplica reglas de conversión y en este caso devuelve true. Para que la intención sea clara, este libro usa === y !== y convierte las entradas de forma explícita cuando corresponde. Los operadores >=, >, <= y < comparan valores; acá comprobamos si una edad numérica alcanza el mínimo.

Resultado y comprobación

Resultado visual

CLUB PROGRAMADOR / JAVASCRIPT

08. Comparar valores y tipos

Compará una edad numérica con la misma edad escrita como texto. Observá la diferencia entre igualdad estricta e igualdad con conversión.

Salida del programa / script.js

```
false
true
true
true
true
```

Editá `script.js`, guardá y recargá con Ctrl+R. También podés consultar la consola del navegador con F12.

El panel y sus estilos son material de apoyo. La solución está en `script.js` y en los archivos complementarios indicados en el libro.

Comprobación

Comprobá las cinco respuestas. Cambiá edad a 17: la comparación edad >= 18 debe ser false. No confundas el resultado booleano con el texto "false".

> **Error frecuente**
>
> Usar = dentro de una comparación: ese operador asigna. Otro error es esperar que === convierta automáticamente un string en number.

> **Desafío adicional**
>
> Compará 0, "0" y false con === y ==. Anotá los resultados antes de ejecutarlos.

09. Combinar condiciones lógicas

El problema

Determiná si una persona puede participar de un taller: debe ser mayor de edad y tener inscripción. Mostrá también si posee algún beneficio.

Solución JavaScript / script.js

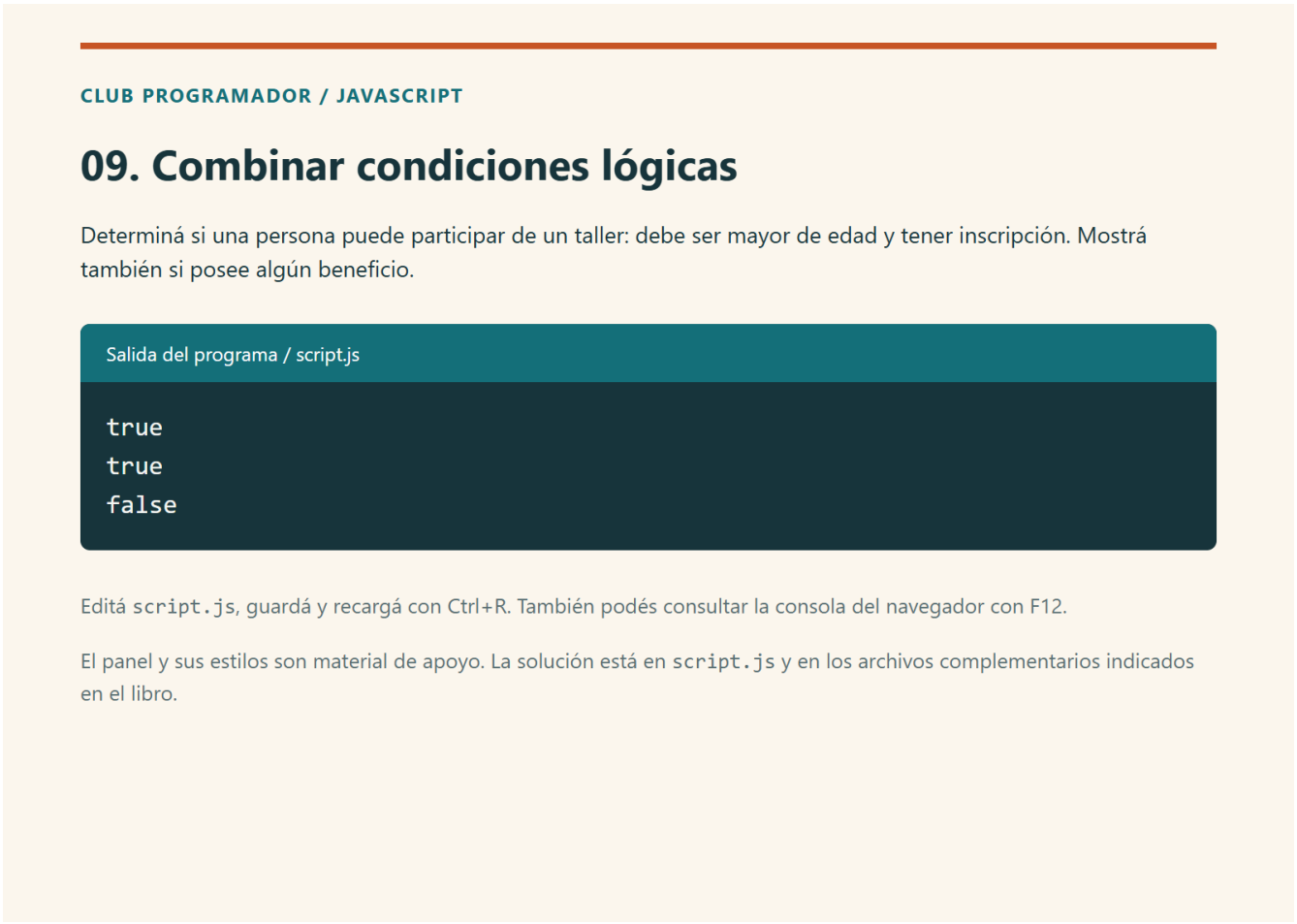
```
const edad = 20;
const inscrita = true;
const tieneBeca = false;
const esSocia = true;
console.log(edad >= 18 && inscrita);
console.log(tieneBeca || esSocia);
console.log(!inscrita);
```

Explicación

&& exige ambas condiciones y || al menos una; ! niega el valor lógico. Con estos operandos booleanos obtenemos booleanos. En condiciones, false, 0, -0, "", null, undefined y NaN se evalúan como falsos; strings no vacíos, arrays y objetos se evalúan como verdaderos. && devuelve el primer operando falso o el último; || devuelve el primero verdadero o el último. Evalúan con cortocircuito: omiten lo que ya no necesitan comprobar. Así pueden devolver valores distintos de true o false.

Resultado y comprobación

Resultado visual



Comprobación

Con los valores iniciales, las respuestas son true, true y false. Cambiá inscrita a false: la primera debe ser false y la tercera true. Cambiá esSocia a false: el beneficio debe desaparecer.

> **Error frecuente**
>
> Usar || cuando se necesitan dos requisitos simultáneos. Una persona menor de edad no debe quedar habilitada solo por estar inscripta.

> **Desafío adicional**
>
> Agregá una condición booleana hayCupo al requisito de participación.

10. Repartir unidades y calcular el resto

El problema

Repartí 23 libros entre 5 cajas con igual cantidad de libros en cada una. Calculá cuántos entran por caja y cuántos sobran.

Solución JavaScript / script.js

```
const libros = 23;
const cajas = 5;
const porCaja = Math.floor(libros / cajas);
const sobrantes = libros % cajas;
console.log(`Por caja: ${porCaja}`);
console.log(`Sobran: ${sobrantes}`);
```

Explicación

La división / puede devolver decimales. Math.floor redondea hacia abajo, por lo que 23 / 5 permite obtener cuatro libros completos por caja. % calcula el resto de la división. Para cantidades enteras no negativas y un divisor positivo, se cumple libros = porCaja * cajas + sobrantes. Estas restricciones importan: no estamos resolviendo todavía divisiones por cero ni repartos de cantidades negativas.

Resultado y comprobación

Resultado visual

CLUB PROGRAMADOR / JAVASCRIPT

10. Repartir unidades y calcular el resto

Repartí 23 libros entre 5 cajas con igual cantidad de libros en cada una. Calculá cuántos entran por caja y cuántos sobran.

Salida del programa / script.js

```
Por caja: 4
Sobran: 3
```

Editá `script.js`, guardá y recargá con Ctrl+R. También podés consultar la consola del navegador con F12.

El panel y sus estilos son material de apoyo. La solución está en `script.js` y en los archivos complementarios indicados en el libro.

Comprobación

Comprobá que 4 * 5 + 3 sea 23. Probá 25 libros: deben entrar 5 por caja y sobrar 0. Con 3 libros y 5 cajas deben entrar 0 por caja y sobrar 3.

> **Error frecuente**
>
> Redondear con Math.round puede asignar más libros de los disponibles. Tampoco debe usarse % como operador de porcentaje.

> **Desafío adicional**
>
> Usá la misma idea para convertir una cantidad entera de minutos en horas y minutos restantes.