



JONAS.IO
SCHMEDTMANN

Desafíos de codificación para todas las secciones

El completo Curso JavaScript



@jonasschmedtman

JS

Tabla de contenido

Instrucciones.....	4
Fundamentos de JavaScript – Parte 1	5
Desafío de codificación n.º 1	5
Desafío de codificación n.º 2	6
Desafío de codificación n.º 3	7
Desafío de codificación #4	8
Fundamentos de JavaScript – Parte 2	9
Desafío de codificación n.º 1	9
Desafío de codificación n.º 2	10
Desafío de codificación n.º 3	11
Desafío de codificación #4	12
Habilidades del desarrollador y configuración del editor	13
Desafío de codificación n.º 1	13
JavaScript en el Navegador: DOM y Eventos	14
Desafío de codificación n.º 1	14
Estructuras de datos, operadores modernos y cadenas	15
Desafío de codificación n.º 1	15
Desafío de codificación n.º 2	17
Desafío de codificación n.º 3	18
Desafío de codificación #4	19
Una mirada más cercana a las funciones	20
Desafío de codificación n.º 1	20
Desafío de codificación n.º 2	22

Trabajando con arreglos	23
Desafío de codificación n.º 1	23
Desafío de codificación n.º 2	24
Desafío de codificación n.º 3	24
Desafío de codificación #4	25
 Programación Orientada a Objetos (POO)	 27
Desafío de codificación n.º 1	27
Desafío de codificación n.º 2	28
Desafío de codificación n.º 3	28
Desafío de codificación #4	29
 JavaScript asíncrono	 30
Desafío de codificación n.º 1	30
Desafío de codificación n.º 2	32
Desafío de codificación n.º 3	33

Instrucciones

- § El Curso completo de JavaScript contiene más de 25 desafíos de codificación para que practiques los conceptos que aprendemos en cada sección, por tu cuenta. **Esta es una parte fundamental de su viaje de aprendizaje.** 🧐
- § Este documento contiene todos los desafíos de codificación de este curso, para que pueda leer el texto del desafío sin ver el video (todavía recomiendo ver los videos porque podría darte algunas pistas adicionales)
- § Algunos desafíos requieren un código de inicio, que puede obtener de este documento en lugar de copiarlo de los videos
- § La solución de cada desafío está en los propios videos, por lo que todavía tienes que ver a verlos 😊
- § Algunos de los últimos desafíos son bastante difíciles a propósito (después de todo, se llaman desafíos, ¿no?). Si alguno de ellos es demasiado difícil, no te preocupes, ¡es completamente normal! Simplemente comience a ver la solución, luego pause el video y siga trabajando por su cuenta. Luchar es parte del viaje, pero puedes hacerlo 🚀

Fundamentos de JavaScript - Parte 1

Desafío de codificación #1

Mark y John están tratando de comparar su IMC (índice de masa corporal), que se calcula mediante la fórmula:

$$\text{IMC} = \text{masa} / \text{altura}^2 = \text{masa} / (\text{altura} * \text{altura}) \text{ (masa en kg y altura en metros).}$$

Sus tareas:

1. Almacene la masa y la altura de Mark y John en variables
2. Calcule el IMC de ambos usando la fórmula (incluso puede implementar ambos versiones)
3. Cree una variable booleana 'markHigherBMI' que contenga información sobre si Mark tiene un IMC más alto que John.

Datos de prueba:

§ Dato 1: Mark pesa 78 kg y mide 1,69 m de altura. John pesa 92 kg y mide 1,95 m de altura

§ Dato 2: La marca pesa 95 kg y mide 1,88 m de altura. Juan pesa 85 kg y mide 1,76 m de altura

BUENA SUERTE 😊

Desafío de codificación #2

Use el ejemplo de BMI del Desafío #1 y el código que ya escribió, y mejórelo.

Sus tareas:

1. Imprima un buen resultado en la consola, diciendo quién tiene el IMC más alto. El mensaje es *"¡El IMC de Mark es más alto que el de John!"* o *"¡El IMC de John es más alto que el de Mark!"*
2. Utilice un literal de plantilla para incluir los valores de IMC en los resultados. Ejemplo: *"Marcos ¡El IMC (28,3) es más alto que el de John (23,9)!"*

Sugerencia: use una declaración if/else



BUENA SUERTE 😊

Desafío de codificación #3

Hay dos equipos de gimnasia, **Dolphins** y **Koalas**. Compiten entre sí 3 veces. ¡El ganador con el puntaje promedio más alto gana un trofeo!

Sus tareas:

1. Calcule el puntaje promedio para cada equipo, usando los datos de prueba a continuación
2. Compare los puntajes promedio del equipo para determinar el ganador de la competencia e imprímalo en la consola. No olvides que puede haber un empate, así que prueba eso también (empatar significa que tienen el mismo puntaje promedio)
3. **Bono 1:** Incluya un requisito para una puntuación mínima de 100. Con esta regla, un
El equipo solo gana si tiene una puntuación más alta que el otro equipo y, al mismo tiempo, una puntuación de al menos 100 puntos. **Sugerencia:** use un operador lógico para probar la puntuación mínima, así como múltiples bloques else-if 😊
4. **Bonificación 2:** ¡La puntuación mínima también se aplica a un empate! Entonces, un empate solo ocurre cuando ambos equipos tienen la misma puntuación y ambos tienen una puntuación mayor o igual a 100 puntos. De lo contrario, ningún equipo gana el trofeo.

Datos de prueba:

§ Dato 1: Los delfines puntúan 96, 108 y 89. Los koalas puntúan 88, 91 y 110

§ Bonificación de datos 1: los delfines obtienen 97, 112 y 101. Los koalas obtienen 109, 95 y 123


§ Bonificación de datos 2: los delfines obtienen 97, 112 y 101. Los koalas obtienen 109, 95 y 106

BUENA SUERTE 😊

Desafío de codificación #4

Steven quiere construir una calculadora de propinas muy simple para cada vez que vaya a comer a un restaurante. En su país, lo habitual es dejar una propina del 15% si el valor del billete está entre 50 y 300. Si el valor es diferente, la propina es del 20%.

Sus tareas:

1. Calcula la propina, según el valor del billete. Cree una variable llamada 'consejo' para esto. No está permitido usar una declaración if/else (si le resulta más fácil, puede comenzar con  declaración if/else y luego intentar convertirla en un operador ternario).
2. Imprima una cadena en la consola que contenga el valor de la factura, la propina y el valor final (factura + propina).
Ejemplo: *"El billete fue 275, la propina fue 41,25 y el valor total 316,25"*

Datos de prueba:

§ Dato 1: Test para valores de billete 275, 40 y 430

Sugerencias:

§ Para calcular el 20% de un valor, simplemente multiplíquelo por $20/100 = 0,2$

§ El valor X está entre 50 y 300, si es $\geq 50 \ \&\& \leq 300$



BUENA SUERTE 

Fundamentos de JavaScript - Parte 2

Desafío de codificación #1

¡Volvamos a los dos equipos de gimnasia, los delfines y los koalas! Hay una nueva disciplina de gimnasia, que funciona de manera diferente.

Cada equipo compite 3 veces y luego se calcula el promedio de los 3 puntajes (es decir, un puntaje promedio por equipo).

Un equipo **solo** gana si tiene al menos **el doble** de la puntuación media del otro equipo.

De lo contrario, ¡ningún equipo gana!

Sus tareas:

1. Cree una función de flecha 'calcAverage' para calcular el promedio de 3 puntajes
2. Usa la función para calcular el promedio de ambos equipos
3. Cree una función 'checkWinner' que tome el puntaje promedio de cada equipo como parámetros ('avgDolphins' y 'avgKoalas'), y luego registre el ganador en la consola, junto con los puntos de victoria, de acuerdo con la regla anterior.

Ejemplo: *"Koalas ganan (30 vs. 13)"*
4. Utilice la función 'checkWinner' para determinar el ganador de los Datos 1 y Datos 2
5. Ignora los sorteos esta vez

Datos de prueba:

§ Dato 1: Delfines puntúan 44, 23 y 71. Koalas puntúan 65, 54 y 49

§ Dato 2: Delfines puntúan 85, 54 y 41. Koalas puntúan 23, 34 y 27

Sugerencias:

§ Para calcular el promedio de 3 valores, sumarlos todos juntos y dividir por 3

§ Para verificar si el número A es al menos el doble del número B, verifique que $A \geq 2 * B$.

Aplique esto a los puntajes promedio del equipo



BUENA SUERTE 😊

Desafío de codificación #2

Steven todavía está construyendo su calculadora de propinas, usando las mismas reglas que antes: dé una propina del 15 % de la factura si el valor de la factura está entre 50 y 300, y si el valor es diferente, la propina es del 20 %.

Sus tareas:

1. Escriba una función 'calcTip' que tome cualquier valor de factura como entrada y devuelva la propina correspondiente, calculada según las reglas anteriores (puede consultar el código del desafío de la calculadora de la primera propina si es necesario). Utilice el tipo de función que más le guste. Pruebe la función utilizando un valor de factura de 100
2. ¡Y ahora usemos arreglos! Entonces cree una matriz 'facturas' que contenga los datos de prueba abajo
3. Cree una matriz de 'propinas' que contenga el valor de la propina para cada factura, calculado a partir de la función que creó antes
4. **Bonificación:** cree una matriz 'total' que contenga los valores totales, por lo que la factura + propina

Datos de prueba: 125, 555 y 44

Sugerencia: ¡Recuerde que una matriz necesita un valor en cada posición, y ese valor en realidad puede ser el valor devuelto por una función! Por lo tanto, puede llamar a una función como valores de matriz (así que no almacene primero los valores de punta en variables separadas, sino directamente en la nueva matriz)



BUENA SUERTE 😊

Desafío de codificación #3

¡Volvamos a Mark y John comparando sus IMC! ¡Esta vez, usemos objetos para implementar los cálculos! Recuerda: $IMC = masa / altura^{**}2 = masa / (altura * altura)$ (masa en kg y altura en metros)

Sus tareas:

1. Para cada uno de ellos, crea un objeto con propiedades para su nombre completo, masa y altura (Mark Miller y John Smith)
2. Cree un método 'calcBMI' en cada objeto para calcular el IMC (el mismo método en ambos objetos). Almacene el valor de IMC en una propiedad y también devuélvalo desde el método
3. Registrar en la consola quién tiene el IMC más alto, junto con el nombre completo y el IMC respectivo. Ejemplo: *"¡El IMC de John (28,3) es más alto que el de Mark (23,9)!"*

Datos de la prueba: Mark pesa 78 kg y mide 1,69 m de altura. Juan pesa 92 kg y mide 1,95 m.

BUENA SUERTE 😊

Desafío de codificación #4

¡Mejoremos aún más la calculadora de propinas de Steven, esta vez usando bucles!

Sus tareas:

1. Cree una matriz 'facturas' que contenga los 10 valores de facturas de prueba
2. Cree matrices vacías para las propinas y los totales ('propinas' y 'totales')
3. Use la función 'calcTip' que escribimos antes (no es necesario repetirla) para calcular las propinas y los valores totales (factura + propina) para cada valor de factura en la matriz de facturas. Usa un `for` bucle para realizar los 10 cálculos!

Datos de prueba: 22, 295, 176, 440, 37, 105, 10, 1100, 86 y 52

Sugerencias: llame a 'calcTip' en el ciclo y use el método `push` para agregar valores a las matrices de propinas y totales 🤔

Prima:

4. **Bonificación:** escriba una función 'calcAverage' que tome una matriz llamada 'arr' como argumento. Esta función calcula el promedio de todos los números en la matriz dada. ¡Este es un **desafío difícil** (no lo hemos hecho antes)! Aquí está cómo resolverlo:
 - 4.1. Primero, deberá sumar todos los valores en la matriz. Para hacer la suma, comience creando una variable 'suma' que comience en 0. Luego recorra la matriz usando un bucle `for`. En cada iteración, agregue el valor actual a la variable 'suma'. De esta manera, al final del bucle, todos los valores se suman
 - 4.2. Para calcular el promedio, divida la suma que calculó antes por la longitud de la matriz (porque esa es la cantidad de elementos)
 - 4.3. Llame a la función con la matriz 'totales'

BUENA SUERTE 😊

Habilidades de desarrollador y configuración del editor

Desafío de codificación #1

Dada una serie de temperaturas máximas pronosticadas, el termómetro muestra una cadena con las temperaturas dadas. **Ejemplo:** [17, 21, 23] imprimirá "... 17°C en 1 día... 21°C en 2 días... 23°C en 3 días..."

Sus tareas:

1. Cree una función 'printForecast' que tome una matriz 'arr' y registre un cadena como la anterior a la consola. Pruébalo con ambos conjuntos de datos de prueba.
2. Utilice el marco de resolución de problemas: comprenda el problema y divídalo en sub-problemas!

Datos de prueba:

§ Dato 1: [17, 21, 23]

§ Dato 2: [12, 5, -5, 0, 4]

BUENA SUERTE 😊

JavaScript en el Navegador: DOM y Eventos

Desafío de codificación #1

¡Implemente una funcionalidad de descanso del juego, para que el jugador pueda hacer una nueva suposición!

Sus tareas:

1. Seleccione el elemento con la clase 'otra vez' y adjunte un controlador de eventos de clic
2. En la función de controlador, restaurar los valores iniciales de la 'puntuación' y variables 'número secreto'
3. Restaurar las condiciones iniciales del mensaje, número, puntaje y entrada de conjetura campos
4. Restaure también el color de fondo original (n.º 222) y el ancho del número (15 rem)

BUENA SUERTE 😊

Estructuras de datos, operadores modernos y cadenas

Desafío de codificación #1

¡Estamos construyendo una aplicación de apuestas de fútbol (fútbol para mis amigos estadounidenses)! 🤖

Supongamos que obtenemos datos de un servicio web sobre un determinado juego (variable 'juego' en la página siguiente). En este desafío vamos a trabajar con esos datos.

Sus tareas:

1. Cree una matriz de jugadores para cada equipo (variables 'players1' y 'players2')
2. El primer jugador en cualquier grupo de jugadores es el portero y los demás son jugadores de campo. Para el Bayern de Múnich (equipo 1), cree una variable ('gk') con el nombre del portero y una matriz ('fieldPlayers') con los 10 jugadores de campo restantes.
3. Cree una matriz 'allPlayers' que contenga todos los jugadores de ambos equipos (22 jugadores)
4. Durante el partido, Bayern Munich (equipo 1) usó 3 jugadores suplentes. Así que cree una nueva matriz ('players1Final') que contenga todos los jugadores originales del equipo 1 más 'Thiago', 'Coutinho' y 'Perisic'
5. Basado en el objeto game.odds, cree una variable para cada probabilidad (llamada 'team1', 'draw' y 'team2')
6. Escriba una función ('printGoals') que reciba una cantidad arbitraria de nombres de jugadores (**no** una matriz) e imprima cada uno de ellos en la consola, junto con la cantidad de goles que se anotaron en total (cantidad de nombres de jugadores pasados en)
7. El equipo con la cuota más baja tiene más posibilidades de ganar. Imprimir en la consola que es más probable que el equipo gane, **sin** usar una declaración if/else o el operador ternario.

Datos de prueba para 6.: Primero, use los jugadores 'Davies', 'Muller', 'Lewandowski' y 'Kimmich'.

Luego, vuelve a llamar a la función con los jugadores de game.scored

BUENA SUERTE 😊

```
const partido =  
  { equipo1: 'Bayern Munich', equipo2:  
    'Borussia Dortmund', jugadores: [ [  
  
      'Noticias',  
      'Adoquín',  
      'Martínez',  
      'Álaba',  
      'Davies',  
      'Kimmich',  
      'Goretzka',  
      'Comán',  
      'Müller',  
      'Gnarby',  
      'Lewandowski', ], [  
  
      'Burki',  
      'Schulz',  
      'Hummels',  
      'Akanji',  
      'Hakimi',  
      'Weigl', 'Witsel',  
      'Hazard',  
      'Brandt',  
      'Sancho',  
      'Gotze', ], ],  
    puntuación:  
      '4:0', anotó:  
        ['Lewandowski',  
        'Gnarby',  
        'Lewandowski', 'Hummels'], fecha: '9 de noviembre de 2037', probabilidades:  
        { equipo1: 1.33, x: 3.25, equipo2: 6.5, }, };
```


Desafío de codificación #2

¡Sigamos con nuestra app de apuestas de fútbol! Sigue usando la variable 'juego' de antes.

Sus tareas:

1. Recorra la matriz `game.scored` e imprima el nombre de cada jugador en la consola, junto con el número de gol (Ejemplo: *"Gol 1: Lewandowski"*)
2. Use un bucle para calcular la cuota promedio y regístrelo en la consola (Ya estudiamos cómo calcular los promedios, puede ir a verificar si no lo recuerda)
3. Imprime las 3 cuotas en la consola, pero con un formato agradable, exactamente así:

Cuota de victoria Bayern Munich: 1.33

Cuota de empate: 3.25

Probabilidad de victoria Borussia Dortmund: 6.5

Obtén los nombres de los equipos directamente desde el objeto del juego, no los codifiques (excepto *"dibujar"*). **Sugerencia:** observe cómo las probabilidades y los objetos del juego tienen la mismos nombres de 🤔

propiedad 4. **Bonificación:** Cree un objeto llamado 'puntuadores' que contenga los nombres de los jugadores que marcaron como propiedades, y el número de goles como valor. En este juego, se verá así:

```
{
  Gnarby: 1,
  Hummel: 1,
  Lewandowski: 2
}
```

BUENA SUERTE 😊

Desafío de codificación #3

¡Sigamos con nuestra app de apuestas de fútbol! Esta vez, tenemos un mapa llamado 'gameEvents' (ver más abajo) con un registro de los eventos que sucedieron durante el juego. Los valores son los eventos en sí, y las claves son los minutos en los que ocurrió cada evento (un partido de fútbol tiene 90 minutos más algún tiempo extra).

Sus tareas:

1. Cree una matriz de 'eventos' de los diferentes eventos del juego que sucedieron (no duplicados)
2. Una vez finalizado el partido, se constata la tarjeta amarilla desde el minuto 64 fue injusto Así que elimine este evento del registro de eventos del juego.
3. Calcule y registre la siguiente cadena en la consola: *"Sucedió un evento, en promedio, cada 9 minutos"* (tenga en cuenta que un juego tiene 90 minutos)
4. Recorra 'gameEvents' y registre cada elemento en la consola, marcando ya sea en la primera mitad o en la segunda mitad (después de 45 min) del juego, así:

[PRIMERA MITAD] 17:  META

BUENA SUERTE 😊

```
const gameEvents = nuevo mapa ([
  [17, , ''],
  [36, 'Sustitución', ],
  [47, , ''],
  [61, 'Sustitución', ],
  [64, 'Tarjeta amarilla', ],
  [69, ' Tarjeta roja'],
  [70, 'Sustitución', ],
  [72, 'Sustitución', ],
  [76, , ''],
  [80, , ''],
  [92, 'Tarjeta amarilla', ],
]);
```

Desafío de codificación #4

Escriba un programa que reciba una lista de nombres de variables escritos en underscore_case y conviértalos a camelCase.

La entrada provendrá de un área de texto insertada en el DOM (consulte el código a continuación para insertar los elementos), y la conversión se realizará cuando se presione el botón.

Datos de prueba (pegados al área de texto, incluidos los espacios):

```
subrayado_caso
primer nombre
alguna_variable
  calcular_EDAD
salida_retrasada
```

Debería producir esta salida (5 salidas separadas de console.log):

```
guión           ✓
primer nombre   ✓✓
alguna          ✓✓✓
calcularEdad    ✓✓✓✓
retrasadaSalida ✓✓✓✓✓
```

Sugerencias:

- § Recordar qué carácter define una nueva línea en el área de texto 🤔
- § La solución solo necesita funcionar para una variable formada por 2 palabras, como a_b
- § Empieza sin preocuparte por el ÿ. Aborrecí eso solo después de que tenga la variable
trabajo de conversión de nombre 😓
- § Este desafío es difícil a propósito, así que empieza a ver la solución por si acaso
estas atorado. ¡Luego haz una pausa y continúa!

¡Después, pruebe con sus propios datos de prueba!

BUENA SUERTE 🍀

```
document.body.append(document.createElement('textarea'));
documento.cuerpo.append(documento.createElement('botón'));
```

Una mirada más cercana a las funciones

Desafío de codificación #1

¡Construyamos una aplicación de encuesta simple!

Una encuesta tiene una pregunta, una serie de opciones entre las que la gente puede elegir y una serie con el número de respuestas para cada opción. Estos datos se almacenan en el objeto de "encuesta" de inicio a continuación.

Sus tareas:

1. Cree un método llamado 'registerNewAnswer' en el objeto 'poll'. El método hace 2 cosas:

1.1. Muestra una ventana de solicitud para que el usuario ingrese el número de la opción seleccionada. El mensaje debería verse así:

¿Cuál es tu lenguaje de programación favorito?

0: Javascript

1: pitón

2: óxido

3: C++

(Escribir número de opción)

1.2. Según el número de entrada, actualice la propiedad de matriz 'respuestas'. Para

Por ejemplo, si la opción es 3, aumente el valor **en la posición 3** de la matriz en 1. Asegúrese de verificar si la entrada es un número y si el número tiene sentido (por ejemplo, la respuesta 52 no tendría sentido, ¿verdad?)

2. Llame a este método cada vez que el usuario haga clic en el botón *"Responder encuesta"*.

3. Cree un método 'displayResults' que muestre los resultados de la encuesta. los

El método toma una cadena como entrada (llamada 'tipo'), que puede ser 'cadena'

o 'matriz'. Si el tipo es 'matriz', simplemente muestre la matriz de resultados tal como está,

usando console.log(). Esta debería ser la opción predeterminada. Si el tipo es 'cadena', muestra una cadena como *"Los resultados de la encuesta son 13, 2, 4, 1"*.

4. Ejecute el método 'displayResults' al final de cada

Llamada al método 'registrarseNuevaRespuesta'.

5. **Bonificación:** use el método 'displayResults' para mostrar las 2 matrices en los datos de prueba. Utilice tanto la opción 'matriz' como la 'cadena'. ¡No coloque las matrices en el objeto de encuesta! Entonces, ¿cómo debería verse esta palabra clave en esta situación?

Datos de prueba para bonificación:

§ Dato 1: [5, 2, 3]

§ Dato 2: [1, 5, 3, 9, 6, 1]

Sugerencias: use muchas de las herramientas que aprendió en esta y la última sección 😊

BUENA SUERTE 😊

```
encuesta constante = {  
  pregunta: "¿Cuál es tu lenguaje de programación favorito?",  
  opciones: ["0: JavaScript", "1: Python", "2: Rust", "3: C++"],  
  
  // Esto genera [0, 0, 0, 0]. ¡Más en la siguiente sección!  
  respuestas: nuevo Array(4).fill(0),  
};
```

Desafío de codificación #2

Esto es más un desafío de *pensamiento* que un desafío de *codificación*. 🤔

Sus tareas:

1. Tome el IIFE a continuación y, al final de la función, adjunte un detector de eventos que cambie el color del elemento h1 seleccionado ('encabezado') a azul, cada vez que se haga clic en el elemento del cuerpo. ¡No vuelva a seleccionar el elemento h1!
2. ¡Y ahora explícate a **ti mismo** (o a alguien a tu alrededor) **por qué** funcionó! Toma todo el tiempo que necesites. Piense en **cuándo** se ejecuta exactamente la función de devolución de llamada y qué significa eso para las variables involucradas en este ejemplo.

```
(función () {  
  encabezado const = documento.querySelector('h1');  
  encabezado.estilo.color = 'rojo';  
})();
```

BUENA SUERTE 😊

Trabajar con arreglos

Desafío de codificación #1

Julia y Kate están haciendo un estudio sobre perros. Entonces, cada uno de ellos preguntó a 5 dueños de perros sobre la edad de su perro y almacenó los datos en una matriz (una matriz para cada uno). Por ahora, solo les interesa saber si un perro es adulto o cachorro. Un perro es adulto si tiene al menos 3 años y es cachorro si tiene menos de 3 años.

Sus tareas:

Cree una función 'checkDogs', que acepte 2 conjuntos de edades de perros ('dogsJulia' y 'dogsKate'), y haga lo siguiente:

1. Julia descubrió que los dueños del **primero** y los **dos últimos** perros en realidad tienen gatos, ¡no perros! Así que cree una copia superficial de la matriz de Julia y elimine las edades de gato de esa matriz copiada (porque es una mala práctica mutar los parámetros de la función)
2. Cree una matriz con los datos de Julia (corregidos) y Kate
3. Para cada perro restante, registre en la consola si es un adulto (*"El perro número 1 es un adulto y tiene 5 años"*) o un cachorro (*"El perro número 2 todavía es un cachorro 🐶"*)
4. Ejecute la función para ambos conjuntos de datos de prueba

Datos de prueba:

§ Datos 1: datos de Julia [3, 5, 2, 12, 7], datos de Kate [4, 1, 15, 8, 3]

§ Datos 2: datos de Julia [9, 16, 6, 8, 3], datos de Kate [10, 5, 6, 1, 4]

Sugerencias: utilice las herramientas de todas las conferencias de esta sección hasta el momento. 🤔

BUENA SUERTE 😊

Desafío de codificación #2

Volvamos al estudio de Julia y Kate sobre perros. Esta vez, quieren convertir las edades de los perros en edades humanas y calcular la edad promedio de los perros en su estudio.

Sus tareas:

Cree una función 'calcAverageHumanAge', que acepte una serie de edades de perros ('edades'), y haga lo siguiente en orden:

1. Calcule la edad del perro en años humanos usando la siguiente fórmula: si el perro tiene ≤ 2 años, $\text{edadhumana} = 2 * \text{edadperro}$. Si el perro tiene > 2 años, $\text{humanAge} = 16 + \text{dogAge} * 4$
2. Excluya todos los perros que tengan menos de 18 años humanos (que es lo mismo que tener perros que tengan al menos 18 años)
3. Calcule la edad humana promedio de todos los perros adultos (ya deberías saberlo) de otros desafíos cómo calculamos los promedios) 🤔
4. Ejecute la función para ambos conjuntos de datos de prueba

Datos de prueba:

§ Dato 1: [5, 2, 4, 1, 15, 8, 3]

§ Dato 2: [16, 6, 10, 5, 6, 1, 4]

BUENA SUERTE 😊

Desafío de codificación #3

Vuelva a escribir la función 'calcAverageHumanAge' del Desafío n.º 2, pero esta vez como una función de flecha, ¡y usando encadenamiento!

Datos de prueba:

§ Dato 1: [5, 2, 4, 1, 15, 8, 3]

§ Dato 2: [16, 6, 10, 5, 6, 1, 4]

BUENA SUERTE 😊

Desafío de codificación #4

Julia y Kate todavía están estudiando perros, y esta vez están estudiando si los perros comen demasiado o muy poco.

Comer demasiado significa que la porción actual de comida del perro es mayor que la porción recomendada, y comer muy poco es lo contrario.

Comer una cantidad aceptable significa que la porción de comida actual del perro está dentro de un rango de 10 % por encima y 10 % por debajo de la porción recomendada (ver sugerencia).

Sus tareas:

1. Recorra la matriz de 'perros' que contiene objetos de perro y, para cada perro, calcule la porción de comida recomendada y agréguela al objeto como una nueva propiedad. No **cre** una nueva matriz, simplemente recorra la matriz. Foro: comida recomendada = peso ** 0,75 * 28. (El resultado es en gramos de comida y el peso debe estar en kg)
2. Encuentra el perro de Sarah e inicia sesión en la consola, ya sea que esté comiendo demasiado o demasiado poco. **Sugerencia:** algunos perros tienen varios dueños, por lo que primero debe encontrar a Sarah en la matriz de propietarios, por lo que este es un poco complicado (a propósito) 🤔
3. Cree una matriz que contenga a todos los dueños de perros que comen demasiado ('propietariosComenDemasiado') y una matriz con todos los dueños de perros que comen muy poco ('propietariosComenDemasiadoPoco').
4. Registre una cadena en la consola para cada matriz creada en 3., así: *"¡Matilda y Alice y los perros de Bob comen demasiado!"* y *"¡Los perros de Sarah, John y Michael comen muy poco!"*
5. Registre en la consola si hay algún perro comiendo **exactamente** la cantidad de comida que se recomienda (solo verdadero o falso)
6. Registre en la consola si hay algún perro comiendo una **cantidad adecuada** de comida (solo verdadero o falso)
7. Cree una matriz que contenga los perros que comen una **buena** cantidad de comida (intente reutilizar la condición utilizada en 6).
8. Cree una copia superficial de la matriz 'perros' y ordénela por porción de comida recomendada en orden ascendente (tenga en cuenta que las porciones están dentro de los objetos de la matriz)



Sugerencias:

§ Use muchas herramientas diferentes para resolver estos desafíos, puede usar el resumen conferencia para elegir entre ellos 😊

§ Estar dentro de un rango 10% arriba y abajo de la porción recomendada significa: $\text{actual} > (\text{recomendado} * 0.90)$ && $\text{actual} < (\text{recomendado} * 1.10)$. Básicamente, la porción actual debe estar entre el 90% y el 110% de la porción recomendada.

Datos de prueba:

```
const perros = [  
  { peso: 22, curFood: 250, dueños: ['Alice', 'Bob'] },  
  { peso: 8, curFood: 200, propietarios: ['Matilda'] },  
  { peso: 13, curFood: 275, propietarios: ['Sarah', 'John'] },  
  { peso: 32, curFood: 340, propietarios: ['Michael'] },  
];
```

BUENA SUERTE 😊

Programación Orientada a Objetos (POO)

Desafío de codificación #1

Sus tareas:

1. Use una función constructora para implementar un 'Auto'. Un automóvil tiene una propiedad de 'marca' y 'velocidad'. La propiedad 'velocidad' es la velocidad actual del automóvil en km/h
2. Implemente un método de 'aceleración' que aumentará la velocidad del automóvil en 10 y registrará la nueva velocidad en la consola.
3. Implemente un método de 'freno' que reduzca la velocidad del automóvil en 5, y registre la nueva velocidad a la consola
4. Cree 2 objetos 'Car' y experimente llamando 'acelerar' y 'frenar' varias veces en cada uno de ellos

Datos de prueba:

§ Datos coche 1: 'BMW' yendo a 120 km/h

§ Datos coche 2: 'Mercedes' yendo a 95 km/h

BUENA SUERTE 😊

Desafío de codificación #2

Sus tareas:

1. Vuelva a crear el Desafío n.º 1, pero esta vez usando una clase ES6 (llámela 'CarCl')
2. Agregue un getter llamado 'speedUS' que devuelve la velocidad actual en mi/h (divida por 1.6)
3. Agregue un setter llamado 'speedUS' que establece la velocidad actual en mi/h (pero lo convierte a km/h antes de almacenar el valor, multiplicando la entrada por 1,6)
4. Crea un auto nuevo y experimenta con 'acelerar' y 'frenar' métodos, y con getter y setter.

Datos de prueba:

§ Datos coche 1: 'Ford' yendo a 120 km/h

BUENA SUERTE 😊

Desafío de codificación #3

Sus tareas:

1. Use una función constructora para implementar un automóvil eléctrico (llamado 'EV') como un **niño** "clase" de 'Coche'. Además de la marca y la velocidad actual, el 'EV' también tiene la carga actual de la batería en % (propiedad de 'carga')
2. Implemente un método 'chargeBattery' que tome un argumento 'chargeTo' y establezca la carga de la batería en 'chargeTo'
3. Implemente un método de 'aceleración' que aumentará la velocidad del automóvil en 20, y disminuir la carga en un 1%. Luego registre un mensaje como este: 'Tesla va a 140 km/h, con una carga del 22%'
4. Cree un objeto de automóvil eléctrico y experimente llamando 'acelerar', 'brake' y 'chargeBattery' (carga al 90%). ¡Observe lo que sucede cuando 'acelera'! **Sugerencia:** revise la definición de polimorfismo 😊

Datos de prueba:

§ Coche de datos 1: 'Tesla' yendo a 120 km/h, con una carga del 23%

BUENA SUERTE 😊

Desafío de codificación #4

Sus tareas:

1. Vuelva a crear el Desafío n.º 3, pero esta vez usando clases ES6: cree una clase secundaria 'EVCi' de la clase 'CarCI'
2. Hacer que la propiedad 'carga' sea privada 3.

Implementar la capacidad de encadenar 'acelerar' y 'cargar batería'

métodos de esta clase, y también actualice el método 'brake' en la clase 'CarCI'. ¡Entonces experimenta con el encadenamiento!

Datos de prueba:

§ Data car 1: 'Rivian' yendo a 120 km/h, con una tasa del 23%

BUENA SUERTE 😊

JavaScript asíncrono

Desafío de codificación #1

En este desafío, creará una función 'whereAml' que representa un país **solo** en función de las coordenadas GPS. Para eso, utilizará una segunda API para geocodificar las coordenadas. Entonces, en este desafío, usará una API por tu cuenta por primera vez 😊

Sus tareas:

PARTE 1

1. Cree una función 'whereAml' que tome como entradas un valor de latitud ('lat') y un valor de longitud ('lng') (estas son coordenadas GPS, los ejemplos se encuentran en los datos de prueba a continuación).
2. Realice una "geocodificación inversa" de las coordenadas proporcionadas. La geocodificación inversa significa convertir las coordenadas en una ubicación significativa, como el nombre de una ciudad y un país. Utilice esta API para realizar la geocodificación inversa: <https://geocode.xyz/api>. La llamada AJAX se hará a una URL con este formato:
<https://geocode.xyz/52.508,13.381?geoit=json>. Use la API de búsqueda y las promesas para obtener los datos. No use la función 'getJSON' que creamos, eso es hacer trampa 😊
3. Una vez que tenga los datos, mírelos en la consola para ver todos los atributos que recibió sobre la ubicación proporcionada. Luego, utilizando estos datos, registre un mensaje como este en la consola:
"Estás en Berlín, Alemania"
4. Encadene un método .catch al final de la cadena de promesas y registre los errores en el consola
5. Esta API te permite realizar solo 3 solicitudes por segundo. Si recarga rápido, obtendrá este error con el código 403. Este es un error con la solicitud. Recuerda, fetch() **no** rechaza la promesa en este caso. Así que cree un error para rechazar la promesa usted mismo, con un mensaje de error significativo

PARTE 2

6. Ahora es el momento de usar los datos recibidos para representar un país. Así que tome el atributo relevante del resultado de la API de geocodificación y conéctelo a la API de países que hemos estado usando.
7. Representa el país y detecta cualquier error, tal como lo hicimos en la última lección (incluso puedes copiar este código, no es necesario que escribas el mismo código)

Datos de prueba:

§ Coordenadas 1: 52.508, 13.381 (Latitud, Longitud)

§ Coordenadas 2: 19.037, 72.873

§ Coordenadas 3: -33.933, 18.474

BUENA SUERTE 😊

Desafío de codificación #2

¡Para este desafío tendrás que ver el video! Luego, crea la función de carga de imágenes que te acabo de mostrar en la pantalla.

Sus tareas:

Las tareas no son muy descriptivas esta vez, por lo que puedes resolver algunas cosas por ti mismo. Finge que estás trabajando por tu cuenta 🤔

PARTE 1

1. Cree una función 'createImage' que reciba 'imgPath' como entrada.

Esta función devuelve una promesa que crea una nueva imagen (use `document.createElement('img')`) y establece el atributo `.src` en la ruta de la imagen proporcionada

2. Cuando la imagen termine de cargarse, agréguela al elemento DOM con el clase 'images' y resuelve la promesa. El valor cumplido debe ser el propio elemento de la imagen. En caso de que haya un error al cargar la imagen (escuche el evento 'error'), rechace la promesa

3. Si esta parte es demasiado complicada para ti, mira la primera parte de la solución.

PARTE 2

4. Consuma la promesa usando `.then` y también agregue un controlador de errores
5. Después de que se haya cargado la imagen, pause la ejecución durante 2 segundos usando el botón 'esperar' función que creamos anteriormente
6. Después de que hayan pasado los 2 segundos, oculte la imagen actual (establezca la propiedad CSS de visualización en 'ninguna') y cargue una segunda imagen (**Sugerencia:** use el elemento de imagen devuelto por la promesa 'createImage' para ocultar la imagen actual. necesitará una variable global para eso)



7. Después de que se haya cargado la segunda imagen, vuelva a pausar la ejecución durante 2 segundos.
8. Después de que hayan pasado los 2 segundos, oculta la imagen actual

Datos de prueba: Imágenes en la carpeta img. Pruebe el controlador de errores pasando una ruta de imagen incorrecta. Establezca la velocidad de la red en "Fast 3G" en la pestaña Red de las herramientas de desarrollo; de lo contrario, las imágenes se cargarán demasiado rápido

BUENA SUERTE 😊


Desafío de codificación #3

Sus tareas:

PARTE 1

1. Escriba una función asíncrona 'loadNPause' que recree el Desafío #2, esta vez usando async/await (solo la parte donde se consume la promesa, reutilice la función 'createImage' de antes)
2. Compare las dos versiones, piense en las grandes diferencias y vea cuál te gusta mas
3. No olvide probar el controlador de errores y configurar la velocidad de la red en "Fast 3G" en la pestaña Red de herramientas de desarrollo

PARTE 2

1. Cree una función asíncrona 'loadAll' que reciba una matriz de rutas de imágenes 'imgArr'
2. Use .map para recorrer la matriz, para cargar todas las imágenes con la función 'createImage' (llame a la matriz resultante 'imgs')
3. ¡Mira la matriz 'imgs' en la consola! ¿Es como esperabas?
4. Use una función combinada de promesa para obtener las imágenes de la matriz 5. Agregue la clase  'paralela' a todas las imágenes (tiene algunos estilos CSS)

Datos de prueba Parte 2: ['img/img-1.jpg', 'img/img-2.jpg', 'img/img 3.jpg']. Para probar, apague la función 'loadNPause'

BUENA SUERTE 