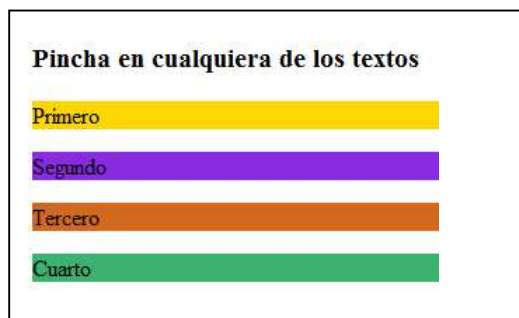


## EJERCICIOS SOBRE ANIMACIONES

1. Realiza una animación que muestre cada uno de los textos de la imagen siguiente, cada uno con un retraso de 1 segundo respecto al anterior. Al pinchar en cualquiera de los textos se vuelve a reproducir la animación desde cero ocultando los textos que se hayan mostrado previamente. Cuando ha finalizado la secuencia, se puede volver a reproducir haciendo click igualmente en cada uno de los textos.



2. Crea una cuenta atrás teniendo en cuenta la imagen de debajo y las consideraciones que se indican a continuación:

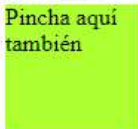
Número de segundos

(Contador)

- El número de segundos es un campo que acepta números enteros mayores de 0.
  - Una vez seleccionado el número de segundos, el botón “Iniciar” comienza una cuenta atrás que se muestra sobrescribiendo el texto “Contador”.
  - Cuando termina la cuenta atrás se muestra el valor en rojo.
  - El usuario puede interrumpir el contador con el botón “Parar”.
3. Modifica el ejercicio de la carpeta “Ejercicio 3-4” para cumplir las siguientes restricciones:

« » ↺ ↻ 50

Pincha aquí también




- Los botones de arriba a la izquierda permiten mover el cuadrado verde en las direcciones que indican. La cantidad de píxeles del desplazamiento es la que se indica en el cuadro de texto.
- Cada desplazamiento tiene una duración fija de un segundo.
- De momento puedes ignorar el cuadrado gris.

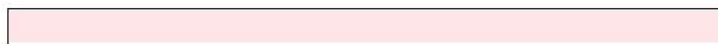
4. Sobre el ejercicio anterior, añade la funcionalidad de arrastrar y soltar el cuadrado verde en cualquier lugar del contenedor principal. Aquí se explica más en detalle: [https://www.w3schools.com/html/html5\\_draganddrop.asp](https://www.w3schools.com/html/html5_draganddrop.asp). Además, si el cuadrado verde se arrastra sobre el cuadrado gris ocurrirá lo siguiente.
  - Si el cuadrado verde se encuentra dentro del cuadrado gris más grande, entonces cambiará su color a rojo y desaparece el texto.
  - Cuando el bloque toque algún límite del cuadrado o esté fuera volverá a su color original.
  - Esta comprobación también debe añadirse cuando el cuadrado se desplaza con las flechas como en el ejercicio 3.
5. Realiza las siguientes modificaciones en el ejercicio 1:
  - La secuencia se volverá a ejecutar automáticamente también cuando termine y no solo cuando el usuario hace click en cualquiera de los textos.
  - Para ello, debes usar promesas y los métodos **then** combinados con los **setTimeout**.
6. El método **fetch** permite cargar un fichero vía JavaScript. Tienes más información sobre su funcionamiento en el sitio: [https://developer.mozilla.org/es/docs/Web/API/Fetch\\_API/Using\\_Fetch](https://developer.mozilla.org/es/docs/Web/API/Fetch_API/Using_Fetch). Su uso es sencillo y basta con pasar una URL absoluta o relativa donde está el fichero. El método devuelve una promesa cuyo método **resolve** tiene un objeto **Response** que contiene una serie de métodos y propiedades que en este caso vamos a ignorar. Lo único que nos interesa de momento es controlar la respuesta que devuelve **fetch**. Por tanto, la estructura sería: `fetch(URL).then((respuesta)=>{...})`. Teniendo en cuenta esta información, realiza la siguiente página teniendo en cuenta las consideraciones que se indican debajo.

## Carga de ficheros

Aquí se realiza el seguimiento

- El select contiene tres valores: “Un fichero” (opción por defecto), “Los dos ficheros”, “El que primero cargue”. En función de la opción seleccionada se realizará la siguiente:
  - La opción “Un fichero” realiza un **fetch** normal en la URL <https://reqres.in/api/users>
  - La opción “Los dos ficheros” realizará un **Promise.all** y el resultado será correcto solo si cargan las URL <https://reqres.in/api/users> y <https://reqres.in/api/login>. El parámetro del **Promise.all** es un array con tantos métodos **fetch** como sea necesario.
  - La opción “El que primero cargue” llamará a un **Promise.race** con un formato similar al anterior, pero en este caso el resultado es la respuesta con el fichero que primero carga.
- El fichero tiene un tamaño tan pequeño que en realidad carga inmediatamente. Pero vamos a simular una barra de progreso en el rectángulo que hay al final del todo de la página. Inicialmente en el texto que hay arriba mostramos el texto “Cargando”. Cuando haya terminado el **fetch** (método **then**) se genera una animación que carga la barra de progreso en 2 segundos. Esta animación se ejecuta en una función que devuelve una promesa para indicar que todo ha ido correctamente en el método **resolve**. El resultado al finalizar será el siguiente:

El fichero cargó bien



- En caso de error, no se llega a rellenar la barra y se muestra el mensaje que devuelva el método **reject** de la promesa de la siguiente manera.

Error: TypeError: Failed to fetch



NOTA: La barra de progreso se puede crear fácilmente con un contenedor que contiene un elemento vacío (por ejemplo un `div`). Este elemento podría tener un ancho inicial de 0% y va creciendo gradualmente durante 2 segundos hasta el 100%. Puedes emplear **requestAnimationFrame** o **setInterval**.

7. Realiza el ejercicio 5 con **async/await** reemplazando los **then** empleados para las promesas.
8. Realiza el ejercicio 6 con **async/await** reemplazando los **then** empleados para las promesas.