

## Ejercicio 2: galería carrusel de imágenes

### 1. Información General

Puntuación total: 10 puntos (sobre 20 total del examen).

- Archivo a modificar: **ej02\_NombreApellido\_galeria.js**
- No modifiques los archivos .css ni el .html (solo en el html la línea `<script src="ej02_NombreApellido_galeria.js"></script>`)
- Renombra el archivo de Javascript para que coincida con tu nombre.

### 2. Descripción del funcionamiento

**MUCHO TEXTO, DON'T PANIC! → CADA APARTADO PRETENDE "GUIARTE" EN LA IMPLEMENTACIÓN**



En la página HTML se muestran varias imágenes dentro de un contenedor. El carrusel de imágenes debe comportarse de la siguiente manera (la implementación se detalla en cada apartado):

- En cada momento **solo una imagen debe estar visible**, identificada mediante la clase CSS **active**.
- Los botones izquierdo y derecho permiten navegar entre las imágenes:
  - Al pulsar el botón derecho, se muestra la siguiente imagen.
  - Al pulsar el botón izquierdo, se muestra la imagen anterior.
  - La navegación es circular: si se avanza desde la última imagen, vuelve a la primera; si se retrocede desde la primera, se muestra la última.
- Se deben generar dinámicamente con js indicadores (círculos) debajo del carrusel, uno por cada imagen:
  - El indicador correspondiente a la imagen visible debe mostrarse como activo.
  - Al pulsar sobre cualquier indicador, el carrusel debe mostrar la imagen asociada.
- El teclado también permite controlar el carrusel:
  - Tecla **flecha derecha (ArrowRight)** → siguiente imagen.
  - Tecla **flecha izquierda (ArrowLeft)** → imagen anterior.
- El carrusel incluye un botón de **reproducción automática**:

- Al pulsarlo, las imágenes avanzan automáticamente cada segundo.
- Al volver a pulsarlo, se detiene la reproducción automática.
- El texto del botón debe cambiar según la acción que produzca.

### 3. Instrucciones de implementación

#### 3.1. Creación de variables globales (1 punto)

Crea las siguientes variables globales donde almacenar los siguientes elementos del DOM:

- **prevBtn**: El elemento del DOM con el botón izquierdo.
- **nextBtn**: El elemento del DOM con el botón derecho.
- **indicadoresContainer**: El elemento div del DOM padre de los indicadores a generar dinámicamente más tarde.
- **autoBtn**: El botón de reproducción automática.
- **imagenes**: Un array (o mejor aún, un NodeList) con cada elemento del DOM que contiene cada imagen de la galería/carrusel.

**NOTA: puedes utilizar o no estas variables en los siguientes apartados**

#### 3.2. Función para mostrar imágenes: muestraImg() (2,5 puntos)

Para mostrar una imagen tienes que emplear la clase CSS "active" en su elemento del DOM correspondiente (por defecto el .html muestra la imagen 'img1.jpg').

Crea una función llamada **muestraImg** que:

- Recibe un parámetro que indica la imagen que queremos mostrar
- Añade la clase **active** solo al elemento DOM de esa imagen y la elimina del resto.
- Llama a la función **activaIndicador(x)** para actualizar los indicadores (los "circuititos").
  - **x** hará referencia al indicador correspondiente a la imagen visible.
  - **activaIndicador()** **SE IMPLEMENTARÁ EN OTRO APARTADO**. Para que tu código no falle, puedes usar temporalmente:

```
function activaIndicador(x) {
    console.log("FALTA POR IMPLEMENTAR! activaIndicador");
}
```

#### 3.3. Control del carrusel mediante botones y teclado (2 puntos)

##### 3.3.1. Funciones para imagen "siguiente" y "anterior" (0,75 puntos)

Antes de programar los botones y el teclado, debes crear **dos funciones** que gestionen el avance y retroceso de las imágenes:

- **nextImage()** (0,5 punto): muestra la siguiente imagen del carrusel. Si no hay imagen siguiente que mostrar, debe mostrarse la primera.
- **prevImage()** (0,5 punto): muestra la imagen anterior. Si se retrocede desde la primera imagen, debe mostrarse la última.

**Ambas funciones deben reutilizar la función muestraImg()**

##### 3.3.2. Botones izquierdo y derecho (0,5 puntos)

- Programa el botón derecho (**nextBtn**) para que muestre la siguiente imagen llamando a

la función **nextImage()** cada vez que se pulse.

- Programa el botón izquierdo (**prevBtn**) para que muestre la imagen anterior llamando a la función **prevImage()** cada vez que se pulse.

### 3.3.3. Control mediante teclado (0,75 puntos)

Programa la navegación mediante el teclado. Al pulsar con el teclado las flechas izquierda o derecha debe de suceder lo siguiente:

- Tecla **ArrowRight**: mostrar la siguiente imagen usando **nextImage()**.
- Tecla **ArrowLeft**: mostrar la anterior usando **prevImage()**.

### 3.4. Reproducción automática del carrusel o auto-play (2,5 puntos)

Programa el botón de auto-play para conseguir la reproducción automática de imágenes, actuando como un interruptor con la siguiente funcionalidad:

- Si la reproducción automática está desactivada, debe iniciarse, haciendo que el carrusel avance automáticamente a la siguiente imagen cada segundo llamando a **nextImage()**.
- Si la reproducción automática está activa, se detiene.

El texto del botón debe reflejar la acción disponible:

- **Inicia auto-play** cuando está desactivada, para poder iniciarla.
- **Stop** cuando está activa, para poder pararla.

### 3.5. Indicadores o "circulitos" (2 puntos)

Crea tantos indicadores (los "circulitos") como imágenes tiene el carrusel (con JavaScript sin modificar el HTML).

Cada indicador representa o se corresponde con una imagen del carrusel. Implementa la siguiente funcionalidad:

1. Cada indicador un elemento **span** que está dentro del **<div class="indicadores-container"></div>**.
2. Si haces click en un indicador, tiene que mostrar su imagen correspondiente llamando a **muestraImg()**.
3. Para cambiar el aspecto del indicador correspondiente a la **imagen visible** debe tener la clase CSS **active** (y el resto no deben de tenerla).

Para implementar el punto c), hazlo implementando **activaIndicador(x)** que se llamará desde la función **muestraImg()**.