Curso: Desarrollo web FrontEnd

Docente: David Alcolea

DETALLE Y DESCRIPCPIÓN

Nombre de la actividad: **Creación de una galeria de fotos dinámica**

Objetivos de la actividad:

- Carga dinámica de elementos utilizando arrays
- Activación de listeners estáticos y dinámicos
- Sincronización de arrays
- Comprobar existencia de elementos en un array
- Uso de los selectores javascript ECMA6
- Trabajar con posicionamentos absolutos y relativos
- Identificar objetos del documento web
- Trabajar con elementos ocultos
- Recorrer elementos de un array
- Utilizar el storage de javascript para guardar y recuperar datos

Competencias asociadas a la actividad:

Competencias técnicas	Soft Skills
• Consolidar uso de listeners estáticos y dinámicos • Consolidar el uso de arrays • Trabajar con posicionamientos absolutos y relativos • Consolidar la carga dinámica de elementos • Identificar elementos de un documento web • Uso de la API de storage javascript • Uso de los métodos para convertir arrays a JSON	• Resolución de problemas • Interpretar requerimientos • Gestionar un proyecto • Búsqueda, gestión y uso de la información

Instrucciones metodológicas :

En esta actividad:

- Consolidaremos el uso de listeners, elementos dinámicos, arrays y selectores confeccionando una galería de fotos dinámica
- Utilizaremos el storage de javascript para guardar y recuperar datos

Esta actividad consta de las siguientes tareas:

1.- Crear la maquetación básica HTML y CSS que utilizaremos para mostrar la galería de imágenes

2.- Creación de la fototeca para la selección de fotos a incorporar en la galería

3.- Añadir fotos de la fototeca en la galería

4.- Permitir borrar fotos de la galería
5.- Seleccionar una foto de la galería y ampliarla dentro de un lightbox
6.- Navegar por las imágenes de la galería dentro del lightbox
7.- Guardar la galería en el storage cuando seleccionemos una foto o la damos de baja
8.- Recuperar la galería del storage al cargar la página

Explicación de la actividad (Elaboración de una galería de imágenes dinámica)

EJERCICIO 1: ELABORACIÓN DEL DOCUMENTO HTML DONDE MOSTRAREMOS LA GALERIA

Crear la maquetación del documento HTML que utilizaremos para seleccionar las fotos y mostrarlas en la galería:

La maquetación es orientativa. Cada alumno puede utilizar el esquema de color que considere oportuno. No obstante, el documento html tendría que contar con las siguientes secciones:

1.- Caja contenedora (marcada con el nombre 'contenedor' en la imagen) con las siguientes características (así aprovechamos para repasar un poco de CSS):

- Centrada
- Con un ancho de aproximadamente el 75% - 80% del ancho del navegador
- Con un margen interior (*padding*) de mínimo 30px
- Con un borde de mínimo 1px (*border*)
- Con las esquinas ligeramente redondeadas (*border-radius*)
- Con un color de fondo (*background-color*)
- Que el contenido que incorporemos se muestre centrado (*text-align*)

2.- La caja contenedora tendrá los siguientes elementos:

2.1.- Un título con el nombre 'Galería de fotos' o similar

2.2.- Debajo del título una sección galería (marcada con color naranja y el nombre 'galería' en la imagen):

Al cargar la página esta sección estará vacia. Al seleccionar fotos en la sección de debajo mostraremos aquí las fotos seleccionadas con un ancho de 260px y un alto de 200px

2.3.- Debajo de la sección anterior un segundo título con el nombre 'fototeca' o similar

2.4.- Debajo colocaremos la sección fototeca (marcada con color azul y el nombre 'fototeca' en la imagen):

En esta sección se mostrarán, al cargar la página, las fotos que guardaremos en un array que construiremos en el siguiente paso. Estas fotos tendrán un ancho de 120px y un alto de 90px

EJERCICIO 2.1: CREACIÓN DE LA FOTOTECA PARA LA SELECCIÓN DE FOTOS

Al cargar la página mostraremos en la sección de la fototeca (sección de color azul) todas les fotos que tengamos disponibles para que el usuario, al pulsar con el ratón sobre cada una de ellas, se incorporen en la galería (la sección de color naranja).

Para realizar esto:

- Crearemos un array al inicio del código javascript con el nombre de todas las fotos que tenemos en la carpeta img (este array de aquí en adelante le llamaremos array fototeca).

  NOTA: En la plataforma moddle encontraréis el zip con todas las imágenes

- Al cargar la página ejecutar una función para mostrar las fotos contenidas en el array anterior.

  Tenemos dos procedimientos para realizar esta tarea:

  Procedimiento clásico (el que hemos utilizado en actividades anteriores):

  - Crear una variable vacía para guardar las etiquetas <img>
  - Recorrer el array de fotos con un bucle for y, para cada iteración, guardar en la variable anterior una etiqueta <img> donde su atributo src sea la ruta y el nombre de la foto que tenemos en el array (ejemplo: <img src='img/casa1.jpg'>)
  - Una vez completado el bucle for, trasladamos al documento web, dentro de la sección fototeca, el contenido de esta variable utilizando innerHTML o insertAdjacentHTML

- Dado que utilizaremos la fototeca para seleccionar las fotos que incorporaremos en la sección galería, tendremos que activar un listener para cada una de las fotos. El evento que detectaremos en este caso será el dblclick

Procedimiento utilizando nodos (es el más profesional):

NOTA: si utilizáis este procedimiento es muy recomendable leer bien y entender el pdf de selectores y nodos javascript que encontrareis en la plataforma)

Recorrer el array de fotos con un bucle for y para cada iteración:

- Crear un objeto del tipo imagen

  var objetoImagen = document.**createElement**('img')

- Añadir el atributo src en el objeto creado:

  objetoImagen.**setAttribute**('src', `img/nombreFoto.jpg`)

  donde nombreFoto.jpg es el nombre de la foto del array

- Activamos el listener dblclick directamente sobre el nodo creado:

  objetoImagen.**ondblclick =** anyadirGaleria

  donde anyadirGaleria es la función que utilizaremos más adelante para que la foto seleccionada aparezca en la galería:

  objetoImagen.**ondblclick =** anyadirGaleria

- Trasladar el nodo creado al documento html

  document.querySelector('#fototeca').**append**(objetoImagen)

  donde #fototeca es el id que hemos utilizado para identificar la secciónen el documento

EJERCICIO 2.2: ACTIVACION DE LOS LISTENERS SOBRE LA FOTOTECA

Tenemos que activar un listener para cada una de las fotos de la fototeca de forma que, al realizar un doble click sobre cada foto, se ejecute la función que se encargará de añadir la foto seleccionada en la galería (esta función se explica en el ejercicio 3)

Tenemos dos opciones para ejecutar esta función:

1. Pasando como parámetro el objeto event correspondiente a la foto pulsada
2. Pasando como parámetro de entrada directamente el nombre de la foto pulsada

Opción 1: uso del objeto event

Es la opción tradicional que hemos usado hasta ahora en otros ejercicios. AL pulsar sobre la foto el objeto event nos dirá sobre que foto se ha pulsado para extraer el atributo src con el nombre de la misma utilizando event.target.getAttribute('src')

Opción 2: nombre de la foto como parámetro

Una segunda opción es pasar directamente a la función del ejercicio 3 el índice de la foto pulsada. Para ello realizaremos lo siguiente:

SI hemos confeccionado la fototeca utilizando nodos:

```
for (let i=0; i<arrayFotos.length; i++) {
        let img = document.createElement('img')
        img.setAttribute('src', 'img/'+arrayFotos[i])
        img.ondblclick = function() {mostrarFoto(arrayFotos[i])}
        document.querySelector('#fotos').appendChild(img)
}
```

SI hemos confeccionado la fototeca utilizando innerHTML:

```
for (let i=0; i<arrayNodos.length; i++) {
        arrayNodos[i].ondblclick = function() {
                mostrarFoto(arrayFotos[i])
        }
}
```

Al utilizar en el bucle for una variable local con let, en cada iteración estamos fijando el valor de esta variable para pasar directamente el nombre de la foto que corresponde al índice del array de fotos. Observar que si, en lugar de let utilizamos var y, por tanto, una variable global, vemos que a la función del ejercicio 3 le llegará el valor undefined ya que el índice que se enviará es el 8 y solo tenemos índices de 0 al 7.

EJERCICIO 3: AÑADIR LAS FOTOS SELECCIONADAS DE LA FOTOTECA EN LA GALERIA

En este ejercicio definiremos la función que se ejecutará al realizar doble click sobre cada una de las fotos de la fototeca que hemos confeccionado en el apartado anterior y que servirá para añadir las fotos seleccionadas en la sección galería.

Para realizarlo:

- Crear un array vacío al inicio del código javascript y que utilizaremos para guardar el nombre de las fotos seleccionadas de la fototeca (este array de aquí en adelante le llamaremos array galeria).
- Recuperar la foto seleccionada en la fototeca (recordad que si hemos utilizado la opción del objeto event, éste nos indicará sobre que foto se ha ejecutado el evento)

- Antes de mostrar la foto en la galería tendremos de verificar que no la hemos incorporado anteriormente (no permitiremos fotos duplicadas en la galería). Para hacer ésto:
  - Si hemos utilizado el objeto event necesitamos leer el atributo src de la foto seleccionada

    var rutaFoto = event.target.**getAttribute**('src')
  - El atributo src contiene la ruta y el nombre de la foto (img/nomfoto.jpg) pero nos interesa solo el nombre para poder comprobar si éste existe en el array galería

    var slash = rutaFoto.lastIndexOf('/') buscamos la posición del último '/' dentro de la ruta

    var nombreFoto = rutaFoto.substring(slash+1) el nombre de la foto lo encontraremos desde la posición del último '/' hasta al final del string
  - Una vez tenemos el nombre de la foto verificamos con indexOf o includes si éste existe en el array galería:

    Si existe, mostramos una alerta y no incorporamos la foto en la galería

    Si no existe, guardamos el nombre de la foto en el array galería (con push) y la mostramos en la sección galería del documento html
- Si la foto no existe en la galería, y una vez hemos guardado su nombre en el array, la mostraremos en la sección galería. Para realizar esto podemos utilizar una función específica crearNodo(). Tenemos dos opciones:

  Opción 1: Construcción de las fotos con innerHTML o insertAdjacentHTML

  Esta función realizará las siguientes tareas:
  - Crear una variable vacía para guardar las etiquetas <img>
  - Recorrer el array de galería con un bucle for y, para cada iteración, guardar en la variable anterior una etiqueta <img> donde su atributo src sea la ruta y el nombre de la foto que tenemos en el array (ejemplo: <img src='img/casa1.jpg'>)
  - Una vez completado el bucle for, trasladamos al documento web, dentro de la sección galería, el contenido de esta variable utilizando innerHTML:

    document.querySelector('#galeria').innerHTML = imagenes;

  Opción 2: Construcción de las fotos con el sistema de nodos

  Esta función recorrerá el array de galería, y, para cada iteración. realizará las siguientes tareas:
  - Crear el nodo de la etiqueta <img>

    let imagen = document.createElement('img');
  - Añadir el atributo src y un atributo class para uso futuro

    imagen.setAttribute('src', 'img/'+nombreFoto);

    imagen.classList.add('imagenGaleria);
  - Trasladar el nodo a la galería:

    document.querySelector('#galeria').append(imagen);

EJERCICIO 4.1: BORRAR FOTOS DE LA GALERIA (1ª PARTE)

Permitiremos al usuario borrar fotos de la galería mediante un click sobre una pequeña cruz roja que situaremos en la esquina superior derecha de cada foto.

En esta primera parte del ejercicio veremos como situar esta cruz en cada una de las fotos utilizando los posicionamientos relativos y absolutos de CSS3. Es, por lo tanto, muy recomendable que se haya leído y entendido el documento sobre display y posicionamientos que se encuentra en recursos complementarios

Haremos lo siguiente:

- Cada una de las etiquetas <img> de las fotos de la galería ,que confeccionamos en la función comentada en el ejercicio anterior, las pondremos dentro de una caja ( ) a la que le asignaremos un atributo class especifico (por ejemplo 'imagenesGaleria')
- Dentro de esta caja, a parte de la foto de la galería, incorporaremos otra etiqueta <img> con la imagen de la cruz roja y le asignaremos también un atributo class que nos permita activar después un listener para borrar la foto correspondiente

  <img class='borrar' src='img/borrar.png'>
- De esta forma, cada imagen de la galería tendrá la siguiente estructura HTML:

  ```
  <div class='imagenesGaleria'>
      <img class = 'imagenGaleria' src='img/nombreImagen.jpg'>
      <img class='borrar' src='img/borrar.png'>
  </div>
  ```

  Cuidado porque si hemos utilizado la opción 1 con innerHTML o la opción 2 con nodos, incluir esta nueva estructura será más o menos compleja
- Si todo ha ido bien y miramos el resultado obtendremos algo parecido a:

- Y aquí es donde interviene el atributo position de CSS3:
  - Primero, por temas de usabilidad, es conveniente que el usuario sepa que puede interactuar con la imagen de la cruz (recordad que tiene asignado un atributo class para identificarla). Por tanto haremos que aparezca el icono del cursor de enlace cuando el ratón pase por encima de la imagen (cursor:pointer)
  - Después reduciremos el tamaño de la imagen de la cruz roja a 20px de ancho y 20px de alto

NOTA: Si veis que no hace caso del nuevo tamaño es que la imagen está utilizando las propiedades que hemos definido al ejercicio 1 para las imágenes de a galería (recordad que especificamos 260 x 200px). Para que la imagen de la cruz se muestre con el tamaño 20 x 20px utilizad un selector combinado en el css con la finalidad de sobre escribir el selector genérico definido en el ejercicio 1.

Por ejemplo, si la sección de la galería la habéis identificado con el id 'galeria' utilizad en el css:

```
#galeria img.borrar {width:20px; height:20px}
```

  - Ahora haremos que la imagen se posicione a 5px por debajo del margen superior de la caja contenedora y a 5px a la izquierda del margen derecho. Para realizar esto es donde interviene la propiedad position junto con las propiedades top y right:

```
position: absolute; top: 5px; right: 5px;
```

  - Si todo ha ido bien veremos que la cosa empeora por momentos y obtendremos algo parecido a:

La cruz de todas las fotos se ha situado a 5px del extremo superior y 5px del extremo derecho de la pantalla y en el de la caja contenedora

Para solucionarlo tenemos que asignar a la caja contenedora un posicionamiento diferente de static (que es el posicionamiento por defecto de todos los elementos de un documento html). Lo que haremos es cambiar el atributo position de la caja contenedora (la caja con el class imagenesGaleria) a relative:

```
position: relative;
```

  - Vemos que la cosa mejora algo pero no para lanzar cohetes.

Ahora la cruz si que se ha posicionado a 5px del margen superior y 5px del margen derecho de la caja contenedora, pero es que, por defecto la etiqueta tiene un tipo de display llamado block que ocupa el 100% del ancho de su caja contenedora (y que en nuestro ejercicio es la sección galería). Si utilizáis provisionalmente un background-color en las cajas imagenesGaleria lo podréis ver mucho mejor.

- ¿Qué tenemos que hacer? Pues cambiar el tipo de display de las cajas de las imágenes de la galería por inline-block:

  display: inline-block;

EJERCICIO 4.2: BORRAR FOTOS DE LA GALERIA (2ª PARTE)

Una vez tenemos la estructura correcta de las fotos de la galería, tendremos que activar el listener de las imágenes de la cruz roja (las imágenes con class='borrar')

Para realizar ésto:

- Dentro de la función donde creamos los nodos de la galería (crearNodo) para mostrar las fotos en la pantalla, una vez hemos trasladado éstos a la sección galería, tendremos que activar el listener parar detectar cuando el usuario hace click en la cruz de color rojo. Tenemos tres opciones:

  Opción 1: si creamos la galería utilizando nodos:

  Es la más sencilla ya que las tareas a realizar se limitan a añadir el evento click en el nodo de la imagen de borrar (la cruz roja):

  borrar.onclick = borrarFotoGaleria

Opción 2: activación estática si creamos la galeria utilizando innerHTML:

También podemos activar el listener de la imagen de borrar foto utilizando un elemento estático que ya sepamos que exista cuando cargamos la página, por ejemplo la propia sección galería.

```
document.querySelector('#galeria).onclick = function(event) {
    let elemento = event.target; //recuperamos el elemento sobre el que se ha activado el evento dentro de la sección galería
    if (elemento.nodeName.toUpperCase = 'IMG' &&
    elemento.classList.includes('borrar')) { //si el elemento corresponde a la etiqueta img con class borrar ejecutamos la función de borrar la foto de la galería
        borrarFotoGaleria(event.target) //pasamos como parámetro a la función el elemento sobre el cual se ha activado el evento (la etiqueta de la cruz roja pulsada)
    }
}
```

- Al hacer click sobre la cruz ejecutaremos una función que se encargará de borrar la foto seleccionada del array galería y de la sección galería

  Necesitamos conocer que posición ocupa la foto que corresponde a la cruz que hemos pulsado dentro de la sección galería ya que esta posición coincidirá exactamente con la que ocupa el nombre de la foto en el array galería. Para conocer la posición:

  Opción 1: extraer la posición a partir del nombre de la foto y el array galería

  Buscaremos la posición de la foto a partir del nombre de la imagen a borrar dentro del array galería:

  - Tendremos que buscar la caja contenedora de esta imagen utilizando parentNode (es la caja con class imagenesGaleria

    `let cajaContenedoraImagen = this.parentNode`
  - Desde esta caja buscamos la imagen de la foto que tiene class imagenGaleria

    `let imagen = cajaContenedoraImagen.querySelector('.imagenGaleria')`
  - Leemos el atributo src de la foto para extraer el nombre de la misma:

    `let nombre = imagen.getAttribute('src')`
  - El atributo que obtenemos incorpora también la ruta de la foto que tendremos que suprimir utilizando el procedimiento que se ha explicado en anteriores apartados (utilizando lastIndexOf y substring)
  - Obtenemos la posición de la foto directamente del array;

    `let indexFoto = arrayGaleria.indexOf(nombre)`
  - Borramos la foto del array con arrayGaleria.splice(indexFoto,1)
  - Y, el nodo de la foto utilizando:

    ```
    let seccion = document.querySelector('#galeria'); //buscar el nodo contenedor (div)
    let nodoABorrar = seccion.childNodes[indexFoto]; //buscar el nodo a borrar
    seccion.removeChild(nodoABorrar); //borrar el nodo
    ```

Opción 2: extraer la posición a partir del nombre de la foto informado como atributo de etiqueta

Podemos utilizar la particularidad que tiene HTML5 para crear atributos de etiqueta personalizados para crear uno dentro de la propia imagen de la cruz roja y asigarle como valor el nombre de la foto asociada:

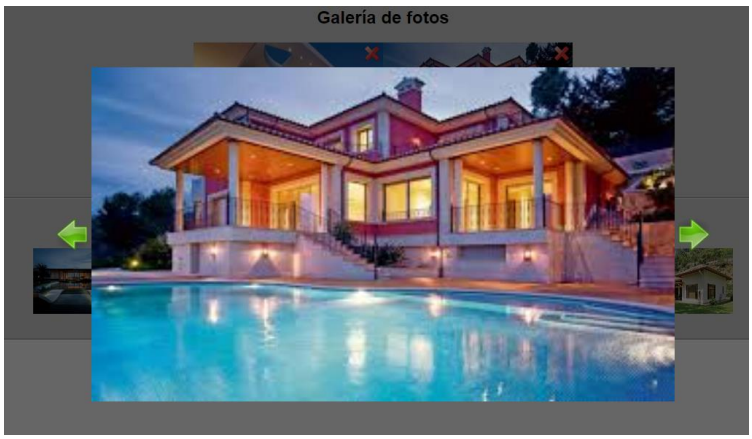
- En la función crearNodos, cuando construimos la etiqueta de borrado de foto añadimos el atributo siguiente:

  data-foto = 'nombreFoto' donde nombreFoto es el nombre de la imagen asociada a borrar
- En la función de borrar la foto obtenemos su nombre mediante:

  let nombreFoto = this.getAttribute('data-foto')
- Y, a partir de aquí, obtenemos su posición en el array y procedemos como en el apartado anterior

EJERCICIO 5.1: AMPLIACIÓN FOTO GALERIA SOBRE LA QUE CLICAMOS (1ª PARTE)

Cuando el usuario haga click sobre una de las fotos de la galería, abriremos una ventana tipo *lightbox* donde mostraremos la foto seleccionada ampliada y con un par de flechas (una a cada lado de la foto) para permitir (en el ejercicio 6) ver la foto anterior o siguiente:



Lo que haremos en esta 1ª parte es construir la caja donde mostraremos la imagen seleccionada y las flechas y que, al cargar la página, estará oculta:

- Dentro de la caja contenedora del documento HTML crear una nueva caja (a continuación de la sección de fototeca) con un id específico para poder identificarla
- Dentro de esta caja incorporar:
  - Una etiqueta <img> con la imagen de la flecha (incluida en el zip de imágenes que encontrareis en la plataforma) y con un id para poder identificarla
  - Una etiqueta <img> con un id para identificarla y con un atributo src vacío (ya que este atributo lo informaremos de forma dinámica al seleccionar una imagen de la galería)

- Otra etiqueta <img> con la flecha pero con un id diferente del que hemos utilizado en la primera imagen

- Y ahora necesitamos un poco de CSS para definir los tamaños y color de la caja y ocultarla inicialmente
  - Primero vemos que las flechas son enormes. Reduciremos su tamaño para que tengan un ancho de 50px (propiedad width). Si no ponemos height el alto se ajustará automáticamente
  - Queremos que la caja ocupe el 100% del ancho de la pantalla y el 100% de alto. Para el ancho podemos utilizar width:100%, pero para el alto no podemos utilizar porcentajes, en su lugar utilizaremos la unidad vh (viewport height): height:100vh
  - Utilizaremos un color de fondo semitransparente para que, cuando se muestre la caja al seleccionar una imagen de la galería, el usuario no tenga la sensación que ha abandonado la página. Para hacer esto utilizaremos la propiedad background-color: rgba(0,0,0,0.6).

    Los tres primeros valores corresponden con el código rgb del color (en este caso negro para el 0,0,0) y el último es el canal alfa o transparencia (en este caso 0.6)
  - Vemos que la caja, como es de esperar, aparece a continuación de la sección de la fototeca, pero queremos que aparezca superpuesta con el resto de secciones. Para ello utilizaremos position:absolute en la caja del *lightbox* junto con los atributos top:0px y left:0px para obligarla a posicionarse en las esquina superior izquierda de la pantalla del navegador
  - Vemos que, como las dos imágenes de les flechas son realmente la misma imagen, las dos aparecen orientadas hacia la izquierda

    Queremos que la flecha derecha se oriente en esa dirección. Para ello utilizaremos, sobre esta flecha, la propiedad transform: rotate(180deg)
  - Y, por último, haremos que la caja de entrada quede oculta utilizando display:none

Faltan, todavía, algunas propiedades CSS que incorporaremos más adelante.

EJERCICIO 5.2: AMPLIACIÓN FOTO GALERIA SOBRE LA QUE HACEMOS CLICK (2ª PARTE)

Una vez tenemos la caja del *lightbox* confeccionada, tendremos que activar un listener para cada una de las imágenes que mostramos a la galería.

Para realizar ésto:

- Dentro de la función donde leemos el array galería para mostrar las fotos en la pantalla, una vez hemos trasladado éstas a la sección galería, tendremos que activar el listener para detectar cuando el usuario clica en cada una de ellas (utilizad un atributo class para identificarlas).
- Al hacer click sobre una imagen ejecutaremos una función que se encargará de abrir el *lightbox* y mostrar la foto seleccionada
  - Necesitamos conocer que posición ocupa la foto sobre la que hemos pulsado dentro del array galería. Para ello:
    - Buscamos que posición ocupa dentro de la sección (tal como hicimos en el ejercicio 4.2):
    - O bien recuperamos el atributo src de la foto seleccionada utilizando el objeto event y buscamos el nombre de la foto en el array (como hemos hecho en el ejercicio 3)
  - Guardamos esta posición en una variable global (es muy importante que sea global, veremos el porqué en el ejercicio 6)
  - Informaremos el atributo src de la foto de la caja del *lightbox* (que, recordad, lo habíamos dejado vacío)
  - Y, finalmente, mostramos la caja (de momento podemos utilizar display:block para ver el resultado)

    `document.getElementById('id_del_lightbox').style.display='block'`
  - Podemos ver que el resultado final no es del todo satisfactorio (foto muy pequeña y alineada en la parte superior de la pantalla)

- Podemos hacer que la foto tenga un ancho del 50% (atributo width aplicado a la foto del *lightbox*)
- Y para que todo el conjunto aparezca centrado vertical y horizontalmente podemos utilizar para mostrar el *lightbox* ,en vez de display:block, el magnífico display:flex
- No nos asustemos cuando veamos el resultado de cambiar block por flex porque tenemos que añadir algunas propiedades css que van muy ligadas a este tipo de display

Por defecto, este display alinea los elementos a la derecha. Para centrarlos horizontalmente utilizaremos (en la caja que hemos creado para el *lightbox*):

justify-content: center;

Y también estira los componentes para que ocupen el 100% del alto de la caja. Para centrarlos verticalmente:

align-items: center

EJERCICIO 5.3: AMPLIACIÓN FOTO GALERIA SOBRE LA QUE HACEMOS CLICK (3ª PARTE)

Por último, una vez mostramos el *lightbox* con la foto seleccionada, necesitamos poder cerrarlo. Para ello:

- Activamos un *listener* al cargar la página para detectar el *click* sobre la caja *lightbox* (la caja es un elemento estático aunque el nombre de la imagen lo informamos dinámicamente)
- Este evento ejecutará una función que lo único que hará será volver a cambiar el display de la caja de flex a none

EJERCICIO 5.4: AMPLIACIÓN FOTO GALERIA SOBRE LA QUE HACEMOS CLICK (EXTRA)

Opcionalmente, podemos utilizar un pequeño efecto de transición al abrir y cerrar el *lightbox*. Para realizar estos efectos se acostumbra a utilizar la librería de ***JQuery*** que, aunque no sea el objetivo del curso, va muy bien para realizar estos efectos.

- La librería la tenemos que incorporar en el head del documento y siempre antes del código javascript que hemos utilizado en el ejercicio:

  <script src='https://code.jquery.com/jquery-3.4.1.min.js '></script>

  No pongáis nunca código javascript entre estas etiquetas script
- Para abrir el *lightbox*, antes de cambiar el display a flex, ponemos la siguiente línea de código

  $('#id_de_lightbox').fadeIn(1000, 'linear')

  $('selector) → elemento sobre el cual queremos que se ejecute el efecto

  fadein(p1, p2) → efecto que queremos ejecutar donde p1 es la duración del efecto en mili segundos y p2 es el tipo de efecto: linear o swing
- Para cerrar el *lightbox* substituimos el cambio de display de flex a none por la siguiente expresión:

  $('#id_de_lightbox').fadeOut(1000, 'linear')

EJERCICIO 6: NAVEGAR POR LAS IMAGENES DE LA GALERIA DENTRO DEL LIGHTBOX

Y para finalizar este ejercicio haremos que, cuando el usuario pulse sobre las flechas del *lightbox*, se muestre la foto anterior o posterior de la galería.

Para ello:

- Al cargar la página activaremos dos listeners, uno para cada flecha, para detectar cuando el usuario hace click en la flecha derecha (mostrar imagen siguiente) o izquierda (mostrar imagen anterior)
- Definiremos la función para mostrar la imagen siguiente:
  - Necesitamos conocer que posición ocupa en el array galería la imagen que aparece en el *lightbox*. Es por este motivo que, cuando seleccionamos una imagen en la galería, guardamos su posición en una variable global: para poder utilizar esta variable dentro de esta función.
  - Cada vez que el usuario pulse sobre la flecha de siguiente imagen incrementaremos en una unidad esta variable

    Pero, hemos de tener presente que, en caso de encontrarnos ya en el último índice del array galería (bien porque la imagen seleccionada para mostrar al lightbox ya era la última o bien porque al pulsar la flecha siguiente ya llegamos a la última) , actualizaremos la variable con el valor 0 (mostraremos la primera imagen)

    NOTA: para saber si la imagen es la última tenemos que preguntar si indiceFoto == arrayGaleria.length-1
  - Por último recuperamos del array galería el nombre de la imagen que ocupa la posición que nos indique esta variable y cambiamos el atributo src de la imagen del *lightbox*
- Definiremos la función para mostrar la imagen anterior. El procedimiento es exactamente el mismo pero aquí:
  - Cada vez que el usuario pulse sobre la flecha de imagen anterior decrementaremos en una unidad esta variable

    Pero, hemos de tener presente que, en caso de encontrarnos ya en el primer índice del array galería (indice 0) actualizaremos la variable con el índice que ocupa la última imagen del array galería (arrayGaleria.length-1)

¿Funciona todo correctamente?

Pues no!. Ahora vemos que, al pulsar sobre las flechas anterior o siguiente estamos cerrando el *lightbox*. Si hemos utilizado el efecto de transición con jquery, veremos como la imagen efectivamente cambia mientras el *lightbox* se cierra. Si no hemos utilizado el efecto veremos como el *lightbox* se cierra sin cambiar de imagen.

Esto es debido a que tenemos otro listener definido sobre la caja contenedora del *lightbox* (el que utilizamos para cerrarlo) y puesto que las flechas se encuentran dentro de esta caja contenedora, están disparando dos eventos:

- evento click de la flecha para cambiar la imagen anterior o siguiente
- evento click de la caja contenedora para cerrar el *lightbox*

Nos interesa que, cuando el usuario pulse sobre las flechas, este evento no se propague a la caja contenedora de forma que solo se ejecuten las funciones que correspondan al cambio de imagen

Para realizar esto, en las funciones para cambiar a imagen anterior o siguiente utilizaremos la siguiente instrucción solo entrar en la función:

```
function imagenAnterior(event) {
        event.stopPropagation()
        .../...
}
```

stopPropagation() impide que un evento se propague hacia los elementos contenedores en caso que estos tengan activo un evento del mismo tipo

EJERCICIO 7: GUARDAR GALERIA AL STORAGE AL AÑADIR O BORRAR UNA FOTO

Cada vez que añadimos una foto a la galería o la borramos guardaremos el array galería en el *storage* de javascript con la finalidad de, al cargar la página en una nueva sesión, conservemos las fotos que, en la sesión anterior, teníamos en la galería.

NOTA: Tenemos dos tipos de storage:

sessionStorage() → los datos se conservan mientras no cerremos el navegador

localStorage() → los datos se conservan aunque que cerremos el navegador y el ordenador

1.- Crearemos una función que se encargue de guardar el array galería en el storage.

- En el *storage* no podemos guardar directamente un array , por tanto el primer paso será convertir el array galería a formato texto utilizando el método join():

  var arrayTexto = arrayGaleria.**join**('//') donde '//' es el carácter o caracteres que queremos utilizar como separador de cada uno de los elementos del array

- Una vez convertido a texto guardamos la variable arrayTexto en el storage:

  localStorage.setItem('galeria', arrayTexto) donde 'galeria' es el nombre que queremos dar al fichero del storage

- Antes de guardar el array en el storage seria deseable comprobar que no no se encuentra vacío (imaginad que hemos borrado la última foto de la galería y no queremos guardar un array vacío en el storage que después nos dará problemas al intentar recuperarlo al cargar la página):

  Si el array está vacío (lo podemos comprobar con array.length) borraremos el storage:

  localStorage.removeItem('galeria')

2.- Cada vez que añadamos una foto de la fototeca en la galería llamaremos a la función anterior para actualizar el *storage*

3.- Cada vez que borremos una foto de la galería llamaremos a la función anterior para actualizar el *storage*:

EJERCICIO 8: RECUPERAR STORAGE AL CARGAR LA PÀGINA

Al cargar la página comprobaremos si existen datos en el storage para actualizar el array galería con las fotos añadidas en una sesión o conexión anterior.

Para ello:

- Comprobaremos si existe el storage:

  localStorage.getItem('galeria')!=undefined
- Recuperamos el contenido del storage y lo guardamos en una variable

  var arrayTexto = localStorage.getItem('galeria') recordad que recuperemos un texto
- Actualizamos el array galería con el contenido del storage. Tenemos que convertir el texto a array utilizando el método split():

  arrayGaleria = arrayTexto.split('//') recordad que '//' son los caracteres separadores que utilizamos para convertir el array a texto en la función del ejercicio anterior
- Una vez actualizado el array galería tenemos que llamar a la función que confeccionamos en el ejercicio 3 para mostrar las fotos de la galería a partir del array