

# Unidad 4. Ejercicios

---

## Contenidos

Ejercicio 1 (1,5 puntos).....	2
Ejercicio 2 (1,5 puntos).....	2
Ejercicio 3 (3,4 puntos).....	3
Ejercicio 4 (3,6 puntos).....	5

Entrega los ejercicios en un fichero llamado ***ud4ejs\_tunombre\_tuprimerapellido.zip***.

### NOTAS:

- Si al ejecutar un ejercicio aparece algún error por consola, el ejercicio contará sobre la mitad de su valor.
- Utilizar el método **trim()** para “limpiar” los valores introducidos por el usuario en los cuadros de texto, cuadros de tipo numérico, etc.
- Si no se especifica lo contrario, se asume que el usuario introducirá datos correctos.
- En cada ejercicio, los **manejadores de eventos** deben asignarse correctamente y, si deben asignarse como manejadores semánticos, hay que asegurarse de que la página se haya cargado por completo antes de asignarlos a los eventos correspondientes. En otro caso, **se restará** al menos 1 punto.

## Ejercicio 1 (1,5 puntos)

A partir del contenido del archivo **ud4ej1.html**, crea un script con una función manejadora llamada **muestraOculto(enlace)** que haga lo siguiente:

1. (0,4) Al pulsar sobre el enlace de una caja “**.accordion-item**”, se oculte su sección asociada (el párrafo al que apunta el atributo **href** del enlace).
2. (0,4) Al volver a pulsar sobre el mismo enlace, se muestre de nuevo su sección asociada.
3. (0,3) Cuando una sección se oculte, cambie la imagen incluida en el enlace correspondiente, de modo que cuando la sección esté visible, la imagen sea “**close.png**” y, cuando la sección esté oculta, la imagen sea “**open.png**”.
4. (0,4) Incluye en el código HTML las llamadas pertinentes a la función manejadora.

**NOTA:** Para ocultar/mostrar contenidos puedes usar la siguiente propiedad CSS:

```
elemento.style.display = "block";
```

```
elemento.style.display = "none";
```

**NOTA:** Puede que el navegador, al cargar la página, asigne la cadena vacía como valor de la propiedad **display** y lo mantenga hasta la segunda pulsación del enlace. Teniendo esto en cuenta, el script debe funcionar correctamente en la primera pulsación del enlace.

## Ejercicio 2 (1,5 puntos)

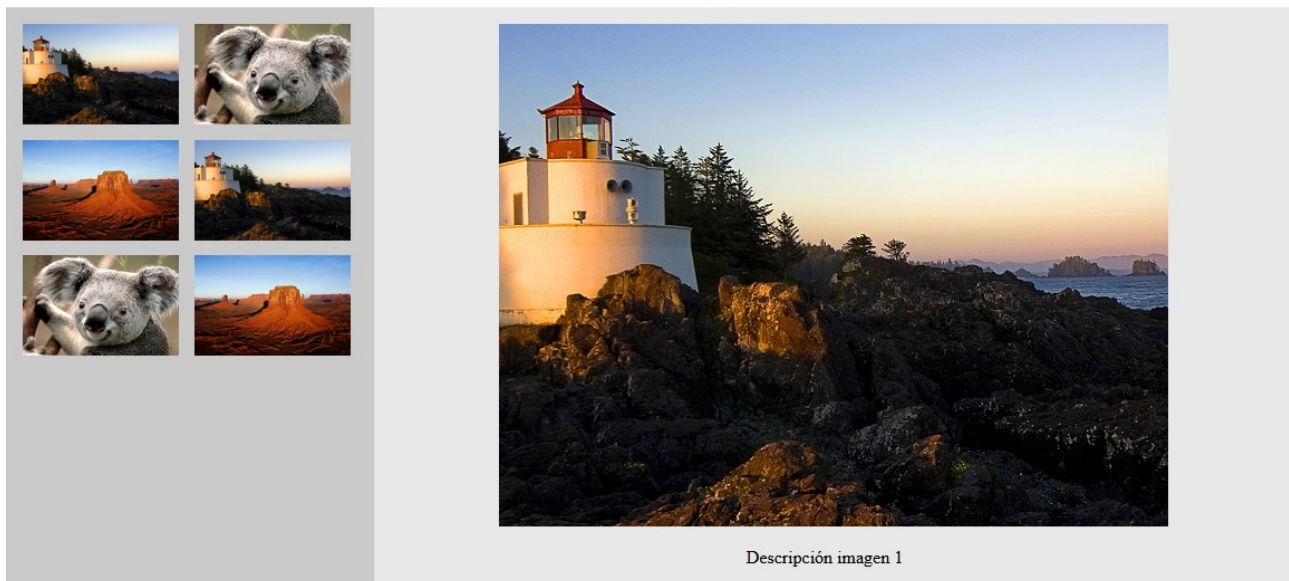
A partir del contenido del archivo **ud4ej2.html**, descarga de Internet 4 o 5 imágenes de un tamaño de 600x450 píxeles y crea un script que gestione un visor de imágenes básico. Para ello:

1. (0,7) Crear una función **main()** como función principal y, en ella, seleccionar todas la imágenes de “**#parte\_lateral**” (las imágenes en miniatura) y asignar a su evento “**click**” un manejador llamado **mostrarEnVisor()**, que realizará las operaciones del visor.

Asigna los manejadores de eventos como **manejadores semánticos**. Asegúrate de que la página se haya cargado por completo antes de llamar a **main()**.

2. Crear el manejador **mostrarEnVisor()**, que se ejecutará al pulsar sobre una imagen en miniatura. En él:
  1. (0,4) Cargar en la imagen de la parte central de la página la imagen pulsada (atributo **src**) con el mismo valor en su atributo **alt** que tiene en la imagen en miniatura.
  2. (0,4) Mostrar, como contenido del elemento **<figcaption>**, el contenido del atributo **alt** de la imagen pulsada. La primera imagen de la parte central debe ser la correspondiente a la primera imagen en miniatura (en el HTML, incluye las imágenes “a mano”).

## Visor de imágenes



## Ejercicio 3 (3,4 puntos)

A partir del contenido del archivo **ud4ej3.html**, crear un script que gestione una cesta de la compra básica. Para ello:

1. Se proporciona un array global llamado **productos** con los productos disponibles. Cada producto tiene las propiedades **id** (id del producto), **name** (su nombre), **price** (su precio), **image** (la imagen del producto) y **qty** (cantidad del producto comprada en la cesta).

La propiedad **qty** deberá reflejar en todo momento la cantidad de un producto en la cesta (al agregar el primer producto de un tipo a la cesta, su valor será **1**). En otro caso, **se restará**.

2. (0,4) Crear una función **main()** como función principal y, en ella, seleccionar todos los elementos con la clase **“.addToCart”** y asignarle a su evento **“click”** un manejador llamado **agregarProductoACesta()**. Al pulsar sobre estos elementos, el manejador realizará las operaciones que constituyen agregar un producto a la cesta (descritas en el punto siguiente).

Asigna los manejadores de eventos como **manejadores semánticos** utilizando el método **addEventListener()**. Asegúrate de que la página se haya cargado por completo antes de llamar a **main()**.

3. (3) Al pulsar sobre el enlace de compra de un producto (con la clase **“.addToCard”**):
  - Obtener del array global **productos** los datos que corresponden al producto sobre el que se ha pulsado. El valor del atributo **id** del enlace con la clase **“.addToCard”** sobre el que se ha pulsado corresponde a la propiedad **id** del producto en el array **productos**.
  - Si el producto está en la cesta, no se hará nada. Si no está en la cesta (puede saberse, por ejemplo, consultando la propiedad **qty** del producto, que será **0** si no está en la cesta):

- Incrementar el valor de su propiedad **qty** en 1.
- Generar un nodo DOM que represente en la cesta al producto, con el marcado:

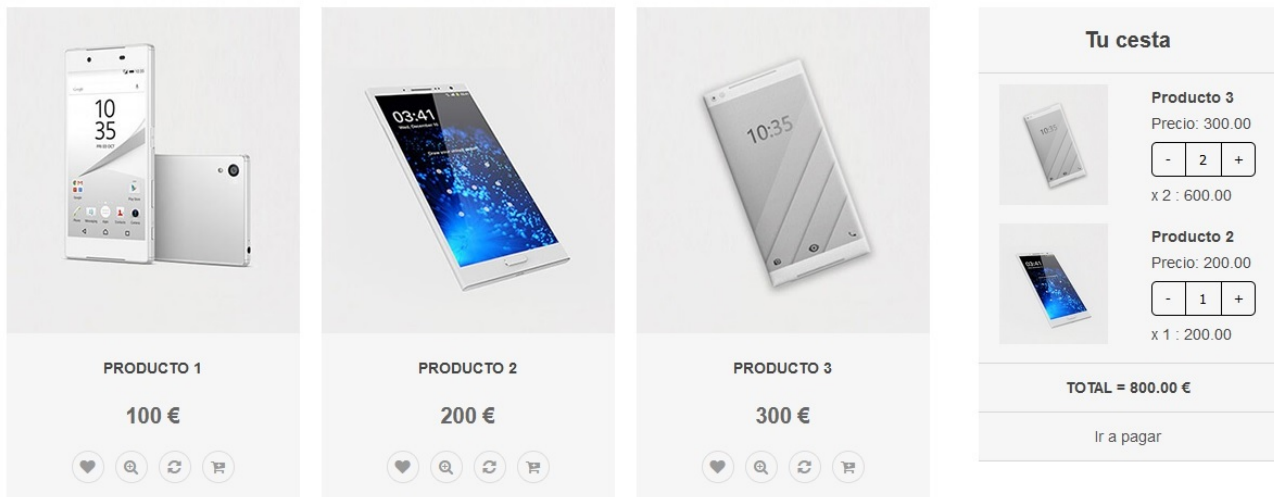
```
<div class="cart-product" data-id-product="">
  <div class="cart-product-image">
    <a href="#"><img src="" alt="..."></a>
  </div>
  <div class="cart-product-data">
    <h6 class="cart-product-name"></h6>
    <p>Precio: <span class="cart-product-price"></span></p>
    <div class="counter" data-product-id="">
      <button class="btnTotal btnDec">-</button>
      <button class="btnQty"></button>
      <button class="btnTotal btnInc">+</button>
    </div>
    <p>x <span class="cart-product-plus"></span> : <span
class="cart-product-total"></span> €</p>
  </div>
</div>
```

En el marcado anterior deberán incluirse los datos del producto: en “**data-id-product**” y en “**data-product-id**”, el valor de la propiedad **id** del producto, en el atributo **src** de la imagen, el valor de la propiedad **image** del producto con su ruta, en “**.cart-product-name**”, el valor de la propiedad **name** del producto, en “**.cart-product-price**”, el valor de la propiedad **price** del producto (con 2 decimales), en “**.btnQty**” y en “**.cart-product-plus**”, el número de productos adquiridos como el seleccionado (disponible en la propiedad **qty** del producto) y, en “**.cart-product-total**”, el total del producto en la cesta (cantidad comprada del producto \* su precio, con 2 decimales).

- Al evento “**click**” del botón “**.btnDec**” del producto generado, asignarle un manejador que **disminuya en 1** la cantidad de productos del mismo tipo comprados en el array **productos** (sin que este valor pueda disminuir por debajo de **1**) y actualice los datos del producto en la cesta (“**.btnQty**”, “**.cart-product-plus**”, “**.cart-product-total**”), y el total de la cesta (“**.cart-product-total**”).
- Al evento “**click**” del botón “**.btnInc**” del producto generado, asignarle un manejador que haga lo mismo que en el anterior, pero **aumentando en 1** la cantidad de productos comprados en el array **productos**.
- Agregar el nodo anterior tras el último producto de la cesta (en la caja “**.cart-items**”).

En la cesta completa, en “**.cart-total-price**”, deberá actualizarse el precio total correspondiente para todos los productos de la cesta, mostrándose con 2 decimales.

**NOTA:** Además de los manejadores descritos, pueden crearse otras funciones de apoyo para llevar a cabo las funcionalidades del script.



## Ejercicio 4 (3,6 puntos)

A partir del contenido del archivo **ud4ej4.html**, crea un script con el código necesario para gestionar el envío de un formulario. Para ello:

**Apartado 1)** (0,75) Crear una función **main()** como función principal y, en ella,

1. Establecer el foco en el campo **#txtNombre**.
2. Asignar como manejador para el evento **onblur** del campo **#txtNombre** una función llamada **validarNombre()**.
3. Asignar como manejador para el evento **onkeypress** del campo **#txtEdad** una **función anónima** que no permita introducir más de 2 caracteres en el campo (cancelando el evento **onkeypress**). La función recibirá como parámetro el objeto **Event** del evento producido.
4. En la lista **#lstComunidades**, establecer como **<option>** seleccionado aquel cuyo atributo **value="0"** (para esto, basta con asignar “0” a la propiedad **value** de la lista).
5. Asignar como manejador para el evento **onchange** de la lista **#lstComunidades** una función llamada **mostrarProvincias()**.
6. Asignar como manejador para el evento **onchange** del campo de archivo **#fileFoto** una función llamada **mostrarFoto()**.
7. Asignar como manejador para el evento **onclick** del botón **#btnVerDatos** una función llamada **mostrarDatos()**.

8. Asignar como manejador para el evento **onsubmit** del formulario una función llamada **validarFormulario(e)**.

**NOTA:** Asigna los manejadores de eventos como **manejadores semánticos**. Asegúrate de que la página se haya cargado por completo antes de llamar a **main()**.

**Apartado 2)** (0,55) Crear una función llamada **validarNombre()**, que será el manejador asignado al campo **txtNombre** y que:

1. (0,25) Si el campo **#txtNombre** está vacío, asigne “Debe escribir su nombre” como contenido del elemento **#errorNombre** y agregue a la lista de clases (atributo “**class**”) de dicho elemento la clase “**campo-error**”. Además, agregue la clase “**error**” a la lista de clases del campo **#txtNombre**.
2. (0,3) Si el campo no está vacío, si el elemento **#errorNombre** tiene la clase “**campo-error**”, asigne como contenido de este elemento la cadena vacía, y elimine de dicho elemento la clase “**campo-error**”. Además, elimine la clase “**error**” de la lista de clases del elemento **#txtNombre**.

**Apartado 3)** (0,8) Crear las siguientes funciones, que serán los manejadores asignados a la lista **lstComunidades** y al campo de archivo **fileFoto**:

1. Una función llamada **mostrarProvincias()** que:
  - a) (0,15) Obtenga el **value** del **<option>** seleccionado en la lista **#lstComunidades** y, a partir de él, las provincias que le corresponden en la variable global **comunidades**, que es un objeto individual con una entrada por cada comunidad autónoma.
  - b) (0,4) Rellene la lista **#lstProvincias** (vaciando su contenido previamente) con un **<option>** por cada provincia obtenida, donde el **value** del **<option>** está en el campo **clave** del objeto correspondiente, y el texto del **<option>** está en el campo **texto**.
2. (0,25) Una función llamada **mostrarFoto(e)** que cargue en la imagen **#imgFoto** (como valor de su atributo **src**) la imagen cargada en el campo de archivo **#fileFoto**.

Un modo de obtener la imagen cargada consiste en acceder a la propiedad **files** de este tipo de elementos, que es un “array” con los archivos cargados por el usuario. Si sólo se carga un archivo, se encontrará en la posición “0” del “array” **files** (**campo.files[0]**).


Los archivos en la propiedad **files** son de un tipo llamado BLOB y, para convertirlos a un URL (que puede cargarse en un elemento **<img>**), puede crearse un **UrlObject** con el método **URL.createObjectURL(archivo)** y asignar el resultado al atributo **src** del **<img>**.

**Apartado 4)** (0,75) Crear una función llamada **validarDatos()**, que validará algunos datos del formulario y mostrará los errores encontrados. La función devolverá **true** si todos los datos son correctos, o **false** en otro caso, y debe realizar las siguientes comprobaciones:

1. (0,35) Si el campo **#txtNombre** está vacío (puede comprobarse la longitud del valor del campo), se añadirá al campo la clase **“error”** (y el campo no pasa la validación). Si no lo está, se eliminará la clase **“error”** del campo.
2. (0,4) Si el **value** de la lista **#lstComunidades** es **“0”**, se añadirá a la lista la clase **“error”** (y el campo no pasa la validación). Si no lo es, se eliminará la clase **“error”** de la lista.

**Apartado 5)** (0,75) Crear las siguientes funciones, que serán los manejadores asignados al botón **#btnVerDatos** y al formulario:

1. (0,5) Una función llamada **mostrarDatos()** que llame a la función **validarDatos()** y, si ésta devuelve **true**, mostrar todos los datos del formulario agregándolos al **<div id=“datos”>**. Cada dato se incluirá en un párrafo. Si se ha cargado una foto, mostrar el mensaje “Foto cargada”, si no, el mensaje “No se cargó foto”. Si se ha marcado **#chkPublicidad**, mostrar el mensaje “Recibirá publicidad”, si no, el mensaje “No recibirá publicidad”.
2. (0,25) Una función llamada **validarFormulario(e)** que llame a la función **validarDatos()** y, si ésta devuelve **false**, cancele la acción por defecto del evento (el envío del formulario).

FORMULARIO	RESUMEN DE DATOS
Nombre: <input type="text" value="Ana"/>	Nombre: Ana
Apellidos: <input type="text" value="Apellidos"/>	Apellidos:
E-mail: <input type="text" value="email@email.com"/>	E-mail: email@email.com
Edad: <input type="text" value="18"/>	Edad: 18
Sexo: <input type="radio"/> Hombre <input checked="" type="radio"/> Mujer	Sexo: Mujer
Incluir mi foto: <input type="button" value="Examinar..."/> photo-3.jpg	Foto cargada
	No recibirá publicidad
Enviar publicidad: <input type="checkbox"/>	Comunidad autónoma: Castilla - La Mancha
Comunidad autónoma: <input type="text" value="Castilla - La Mancha"/>	Provincia: Toledo
Provincia: <input type="text" value="Toledo"/>	Observaciones: Lorem ipsum dolor sit amet, consectetur adipiscing elit. Pellentesque commodo, metus quis rutrum malesuada, lectus dolor consequat est, sed sollicitudin est mi ut nibh.
Observaciones: <input type="text" value="Lorem ipsum dolor sit amet, consectetur adipiscing elit. Pellentesque commodo, metus quis rutrum malesuada, lectus dolor consequat est, sed sollicitudin est mi ut nibh."/>	
<input type="button" value="ENVIAR DATOS"/>	<input type="button" value="VER DATOS"/>
<input type="button" value="BORRAR DATOS"/>	