

# Tutoriales HTML5, CSS3 y JavaScript

**Formación Sopra Steria**

Formación HTML5, CSS3 y JavaScript

Tutoriales HTML5, CSS3 y JavaScript

Versión 1.1 del martes, 08 de enero de 2019

# Historial

Versión	Fecha	Origen de la actualización	Realizado por	Validado por
1.0	14/12/2018		José Pedro Checa Gutiérrez, Gabriel Patricio Del Rey Apollonio, Juan Ignacio García Nicolás	
1.1	08/01/2019		José Pedro Checa Gutiérrez, Gabriel Patricio Del Rey Apollonio, Juan Ignacio García Nicolás	



# Contenidos

1.	Tutorial Capítulo 2: Construyendo la Interfaz de Usuario(UI) Usando HTML5: Textos, Gráficos y Multimedia	5
1.1.	<b>Entendiendo los Conceptos Esenciales de HTML</b>	5
1.1.1.	Etiquetado Básico y Estructura de la Página	5
1.2.	<b>Selección y Configuración de Etiquetas HTML5 para Mostrar el Contenido del Texto</b>	8
1.2.1.	Elementos de Texto de HTML4 con Nuevos Significados y Nuevas Funcionalidades	8
1.2.2.	Nuevos Elementos de HTML5	9
1.3.	<b>Elementos de Texto no Usados en HTML5</b>	9
1.4.	<b>Selección y Configuración de Etiquetas en HTML5 para Mostrar Gráficos</b>	10
1.4.1.	Usando los elementos figure y el figcaption	10
1.5.	<b>Creación de Gráficos con Canvas</b>	13
1.5.1.	Canvas Básico	13
1.5.2.	Uso de Canvas para la Creación de Contornos	15
1.5.3.	Alternativa Hacer Imágenes y Textos para Viejos Navegadores	16
1.6.	<b>Creando Gráficos con SVG</b>	17
1.7.	<b>Entendiendo las Etiquetas de Vídeo y Audio</b>	18
1.7.1.	Trabajando con el Elemento Vídeo	18
1.7.2.	Trabajando con el Elemento Audio	19
2.	Tutorial Capítulo 3: Construyendo la Interfaz de Usuario mediante HTML5: Organización, Valores de Entrada y Validación	21
2.1.	<b>Eliendo y Configurando las Etiquetas de HTML5 para Organizar Contenidos y Formularios</b>	21
2.1.1.	Creando un Documento HTML con Header, Section y Footer	21
2.1.2.	Añadiendo el Elemento Nav a un Documento HTML	22
2.1.3.	Añadiendo el Elemento Aside a un Documento HTML	23
2.1.4.	Creando una Tabla	25
2.1.5.	Creando una Lista Ordenada	26
2.2.	<b>Eliendo y Configurando las Etiquetas de HTML5 para Valores de Entrada y Validación:</b>	27
2.2.1.	Entendiendo los Elementos de Entrada y Formularios	27
3.	Tutorial Capítulo 4: Entendiendo CSS Básico: Contenido Flotante, Posicionamiento y Estilos	31
3.1.	<b>Explorando la Conexión entre HTML y CSS</b>	31
3.2.	<b>Entendiendo Fuentes y Familias de Fuentes</b>	32
3.3.	<b>Manejando Contenido Flotante</b>	33
3.4.	<b>Posicionando Elementos Individuales</b>	35
3.4.1.	Aplicando Posicionamiento Flotante	35

3.4.2.	Aplicando Posicionamiento Absoluto	36
<b>3.5.</b>	<b>Manejando Contenido Desbordante</b>	<b>37</b>
3.5.1.	Entendiendo Scrolling Desbordante	37
3.5.2.	Entendiendo el Desbordamiento Visible y Oculto	39
<b>4.</b>	<b>Tutorial Capítulo 5: Entendiendo CSS Básico: Layouts</b>	<b>40</b>
<b>4.1.</b>	<b>Usando una Flexible Box para Establecer Contenido de Alineación, Dirección y Orientación</b>	<b>40</b>
4.1.1.	Trabajando con Flexbox y Flexbox Items	40
<b>5.</b>	<b>Tutorial Capítulo 6: Manejando Texto Flotante usando CSS</b>	<b>46</b>
<b>5.1.</b>	<b>Entendiendo y Usando Regions para Contenido de Texto entre Múltiples Secciones</b>	<b>46</b>
5.1.1.	Usando Columnas y Unión con Guión para Optimizar la Lectura del Texto	46
<b>6.</b>	<b>Tutorial Capítulo 7: Entendiendo JavaScript y Fundamentos de Programación</b>	<b>49</b>
<b>6.1.</b>	<b>Gestión y Mantenimiento JavaScript</b>	<b>49</b>
6.1.1.	Creación y Uso de Funciones	52
6.1.2.	Usando jQuery y Otras Librerías	55
<b>6.2.</b>	<b>Actualizando la UI Usando JavaScript</b>	<b>56</b>
6.2.1.	Localizando y Accediendo a Elementos	58
6.2.2.	Escuchando y Respondiendo a Eventos	60
6.2.3.	Mostrando Elementos Ocultos	62
6.2.4.	Actualizando el Contenido de Elementos	64
6.2.5.	Añadiendo Elementos	66
<b>7.</b>	<b>Tutorial Capítulo 8: Trabajando con Gráficos y Accediendo a Datos</b>	<b>68</b>
<b>7.1.</b>	<b>Trabajando con Imágenes, Formas y otros Gráficos</b>	<b>68</b>
<b>7.2.</b>	<b>Enviando y Recibiendo Datos</b>	<b>72</b>
7.2.1.	Transmitiendo objetos complejos y Parsing	73
<b>7.3.</b>	<b>Cargando y Guardando Archivos</b>	<b>74</b>
7.3.1.	Usando AppCache	76
<b>7.4.</b>	<b>Usando JavaScript para Validar Campos de Formularios Introducidos por Usuarios</b>	<b>77</b>
<b>7.5.</b>	<b>Entendiendo y Usando Cookies</b>	<b>79</b>
<b>7.6.</b>	<b>Entendiendo y Usando Almacenamiento Local</b>	<b>81</b>



# 1. Tutorial Capítulo 2: Construyendo la Interfaz de Usuario (UI) Usando HTML5: Textos, Gráficos y Multimedia

---

## 1.1. Entendiendo los Conceptos Esenciales de HTML

### 1.1.1. Etiquetado Básico y Estructura de la Página

#### CREANDO UNA PÁGINA WEB SIMPLE

Para crear una página Web sencilla y ver el efecto de la falta de etiquetas, anidamiento y entidades, se pueden realizar los siguientes pasos:

1. Se crea una subcarpeta dentro de la carpeta Mis documentos que contendrá los archivos en los que se trabaja a lo largo de este tutorial. Esta será la carpeta de trabajo. Se puede nombrar la subcarpeta *HTML5* o algo similar.
2. Se abre un editor de páginas Web, una herramienta de desarrollo de aplicaciones o incluso un simple editor de texto como *Notepad* y se escribe lo siguiente:

```
<!doctype html>
<html>
<head>
  <title>78704 Servicio de Mascotas</title>
</head>
<body>
  <h1>Cuidado y alimentación</h1>
  <p>
    Su perro es un amigo de por vida. ¿Por qué no ofrecer el mejor cuidado posible?
  </p>
  <p>
    Asegúrese de que su mascota tenga suficiente <i><b>agua fresca</b></i> en tiempo
    caluroso. Cuando lleve a su perro a dar largos paseos, lleve un recipiente para el agua
    donde pueda beber el perro. Usted puede encontrar estos recipientes en muchas tiendas de
    mascotas por 10 euros o menos.
  </p>
</body>
</html>
```

En la etiqueta `<title>` se ha hecho el título de la página que se va a crear y eso se pondrá generalmente, en el `<head>`. A continuación, se añade un párrafo con un texto en el `<body>` en el que se usa el tipo de letra cursiva `<i>` y la negrita `<b>` para agua fresca y con un encabezado de tipo `<h1>`.

3. Se guarda el archivo como *L2-mascotas-original.html* en la carpeta de trabajo que se creó en Mis Documentos.
4. Se va a la carpeta de trabajo y se abre la página *HTML* en un navegador Web. Debería parecerse a la figura siguiente:



# Cuidado y alimentación

Su perro es un amigo de por vida. ¿Por qué no ofrecer el mejor cuidado posible?

Asegúrese de que su mascota tenga suficiente **agua fresca** en tiempo caluroso. Cuando lleve a su perro a dar largos paseos, lleve un recipiente para el agua donde pueda beber el perro. Usted puede encontrar estos recipientes en muchas tiendas de mascotas por 10 euros o menos.

5. Para ver el efecto de eliminar una etiqueta, se borra la etiqueta `</b>` final, después de "agua fresca". Se crea un nuevo archivo para probar los cambios guardándolo como `L2-mascotas-prueba.html` y se abre en el navegador. Ahora todo el contenido desde "agua fresca" hasta el final está en negrita.

# Cuidado y alimentación

Su perro es un amigo de por vida. ¿Por qué no ofrecer el mejor cuidado posible?

Asegúrese de que su mascota tenga suficiente **agua fresca en tiempo caluroso. Cuando lleve a su perro a dar largos paseos, lleve un recipiente para el agua donde pueda beber el perro. Usted puede encontrar estos recipientes en muchas tiendas de mascotas por 10 euros o menos.**

6. Para ver el efecto de un anidamiento inadecuado, se mueve la etiqueta final `</i>` para que aparezca después de la última etiqueta `</p>`. Se guarda `L2-mascotas-prueba.html` de nuevo y se abre en un navegador. Ahora todo el contenido desde "agua fresca" hasta el final del documento está en negrita y cursiva, como se ve en la siguiente figura.

# Cuidado y alimentación

Su perro es un amigo de por vida. ¿Por qué no ofrecer el mejor cuidado posible?

Asegúrese de que su mascota tenga suficiente ***agua fresca en tiempo caluroso. Cuando lleve a su perro a dar largos paseos, lleve un recipiente para el agua donde pueda beber el perro. Usted puede encontrar estos recipientes en muchas tiendas de mascotas por 10 euros o menos.***



7. Se cierra el archivo *L2-mascotas-prueba.html* en el editor y se abre *L2-mascotas-original.html*.
8. Se agrega una línea de derechos de autor en la parte inferior de la página pulsando **Intro** unas cuantas veces después de la etiqueta de cierre `</p>` y se escribe `<p>&copy; 2018</p>`. Se sustituye el año en curso para "2018", si es necesario. Se pulsa **Intro** para añadir una línea en blanco. Se confirma que la línea de copyright está por encima de las etiquetas finales `</body>` y `</html>`.
9. Se crea un nuevo archivo guardando *L2-mascotas-prueba.html* como *L2-mascotas-copyright.html* y se abre en el navegador. ¿El símbolo del círculo C aparece como se muestra en la figura siguiente? Si no, se cambia `&copy;` a `&#169;`; se guarda el archivo y luego se ve de nuevo.

## Cuidado y alimentación

Su perro es un amigo de por vida. ¿Por qué no ofrecer el mejor cuidado posible?

Asegúrese de que su mascota tenga suficiente *agua fresca* en tiempo caluroso. Cuando lleve a su perro a dar largos paseos, lleve un recipiente para el agua donde pueda beber el perro. Usted puede encontrar estos recipientes en muchas tiendas de mascotas por 10 euros o menos.

© 2018

10. Se va a la página web del Servicio de validación de marcas del W3C en <http://validator.w3.org>. Se sube *L2-mascotas-copyright.html* y se hace clic en **Check** para que el servicio lo compruebe. Se arregla cualquier error reportado por el inspector que se relacionan con etiquetas o errores tipográficos que faltan, si los hubiera.
11. Probablemente se reciba un mensaje de error sobre la codificación de caracteres. Para solucionar esto, se abre *L2-mascotas-copyright.html* en la herramienta de edición, se inserta `<meta charset="UTF-8">` en el elemento de cabecera, en su propia línea, justo antes del título.

```
<head>
  <meta charset="UTF-8">
  <title>78704 Servicio de Mascotas</title>
</head>
```

12. Se guarda el archivo, se carga de nuevo al verificador de validación y se comprueba.
13. Se deja abierta la herramienta de edición y el explorador web si se va a continuar con la siguiente sección.



## 1.2. Selección y Configuración de Etiquetas HTML5 para Mostrar el Contenido del Texto

### 1.2.1. Elementos de Texto de HTML4 con Nuevos Significados y Nuevas Funcionalidades

## MODIFICAR LAS ETIQUETAS RELACIONADAS CON EL TEXTO EN UNA PÁGINA WEB

Para modificar las etiquetas de una página Web, se pueden realizar los siguientes pasos:

1. En una herramienta de edición, se abre *L2-mascotas-copyright.html*.
2. En el párrafo siguiente, se reemplazan las etiquetas en cursiva `<i>` y negrita `<b>` por el elemento `strong`.

Asegúrese de que su mascota tenga suficiente ***agua fresca*** en tiempo caluroso.

El código resultante se verá así:

Asegúrese de que su mascota tenga suficiente **agua fresca** en tiempo caluroso.

3. Se debe tener en cuenta que el elemento `strong` se verá como el elemento en negrita. El W3C prefiere que se use `<strong>` sobre `<b>`, aunque parecen producir casi resultados idénticos. Se añaden etiquetas `<small>` de inicio y final a la línea de copyright, anidándolas correctamente en las etiquetas de párrafo.
4. Se guarda el archivo como *L2-mascotas-modificado.html* y se visualiza en un navegador Web. Aparecerá como se puede ver en la figura siguiente.

## Cuidado y alimentación

Su perro es un amigo de por vida. ¿Por qué no ofrecer el mejor cuidado posible?

Asegúrese de que su mascota tenga suficiente **agua fresca** en tiempo caluroso. Cuando lleve a su perro a dar largos paseos, lleve un recipiente para el agua donde pueda beber el perro. Usted puede encontrar estos recipientes en muchas tiendas de mascotas por 10 euros o menos.

© 2018

5. Se deja abierta la herramienta de edición y el explorador web si se va a continuar con la siguiente sección.





### 1.2.2. Nuevos Elementos de HTML5

#### USAR EL ELEMENTO MARK

Para utilizar el elemento de marca para resaltar texto, se pueden realizar los siguientes pasos:

1. En una herramienta de edición, se abre *L2-mascotas-modificado.html*.
2. Se modifica el siguiente párrafo insertando el elemento `<mark>` alrededor del texto "amigo de por vida". Quedando así: `<p>Su perro es un <mark style="background-color:orange;"> amigo de por vida </mark>.`
3. Se crea un nuevo archivo guardándolo como *L2-mascotas-mark.html* y se visualiza en un navegador Web. La figura siguiente muestra el texto resaltado.

## Cuidado y alimentación

Su perro es un **amigo de por vida**. ¿Por qué no ofrecer el mejor cuidado posible?

Asegúrese de que su mascota tenga suficiente *agua fresca* en tiempo caluroso. Cuando lleve a su perro a dar largos paseos, lleve un recipiente para el agua donde pueda beber el perro. Usted puede encontrar estos recipientes en muchas tiendas de mascotas por 10 euros o menos.

© 2018

4. Se deja abierta la herramienta de edición y el explorador web si se va a con la siguiente sección.

### 1.3. Elementos de Texto no Usados en HTML5

#### VER LOS EFECTOS DE LOS ELEMENTOS OBSOLETOS:

Para ver los efectos de los elementos obsoletos en una página Web *HTML5*, ejecute los siguientes pasos:

1. En una herramienta de edición, se abre *L2-mascotas-mark.html*.
2. Se modifica el encabezado `<h1>` para incorporar el elemento `<center>`, como se muestra:

```
<h1><center>Cuidado y Alimentación</center></h1>
```

3. Se crea un nuevo archivo guardándolo como *L2-mascotas-temp.html* y se visualiza en un navegador Web. Se comprueba que el elemento se centró en el navegador.
4. Se debe añadir el elemento `<big>` al siguiente contenido, como se muestra, que servirá para hacer la letra más grande:

```
<p>Su perro es un <mark style="background-color:orange;">  
<big>amigo de por vida </big></mark>
```



5. Se guarda el documento y se abre en un navegador. Debería mostrarse como en la imagen siguiente:

## Cuidado y Alimentación

Su perro es un **amigo de por vida**. ¿Por qué no ofrecer el mejor cuidado posible?

Asegúrese de que su mascota tenga suficiente *agua fresca* en tiempo caluroso. Cuando lleve a su perro a dar largos paseos, lleve un recipiente para el agua donde pueda beber el perro. Usted puede encontrar estos recipientes en muchas tiendas de mascotas por 10 euros o menos.

© 2018

6. Se va a la página web del Servicio de validación de marcas del W3C en <http://validator.w3.org>. Se sube *L2-mascotas-temp.html* y se hace clic en *Check* para que el servicio lo compruebe.
7. Se observa que el validador muestra errores sobre el uso de los elementos obsoletos. (No validan, pero muchos de ellos aún se renderizan correctamente en un navegador Web.)
8. Se cierra *L2-mascotas-temp.html* y se deja la herramienta de edición y el explorador web abiertos si se va a continuar con la siguiente sección.

### 1.4. Selección y Configuración de Etiquetas en HTML5 para Mostrar Gráficos

#### 1.4.1. Usando los elementos `figure` y el `figcaption`

##### a. Mostrar una Imagen en una Página Web

Para mostrar una imagen en una página Web, se pueden realizar los siguientes pasos:

1. Se busca un archivo *JPG*, *PNG*, *GIF* o *BMP* en el ordenador para utilizarlo en este ejercicio, el cual deberá estar guardado en la misma carpeta en la que se esté haciendo el HTML.
2. Se abre el archivo *L2-mascotas-mark.html* en la herramienta de edición.
3. Se eliminan las etiquetas `<mark>` del primer párrafo.
4. Se inserta el siguiente marcado después del elemento `h1`, dejando una línea en blanco antes y después del elemento `h1` y se reemplaza por *captura.jpg*, que será el archivo de imagen:

```
<figure>
  
  <figcaption>
    Perros Felices Son Perros Buenos
  </figcaption>
```



```
</figure>
```

En `<img>` se añade la imagen que se quiere mostrar y en `<figcaption>` sirve para añadir un texto al pie de la foto.

5. Se crea un nuevo archivo guardándolo como *L2-mascotas-imagen.html* y se visualiza en un navegador Web. La página debe tener un aspecto similar al de la figura siguiente.



6. Se deja abierta la herramienta de edición y el explorador web si se va a continuar con la siguiente lección.

#### b. Uso de Fig y Figcaption

Para mostrar una imagen en una página Web, se pueden realizar los siguientes pasos:

1. Se localizan dos archivos *JPG*, *PNG*, *GIF* o *BMP* adicionales para utilizarlos en este ejercicio, los cuales deberán estar guardados en la misma carpeta en la que se esté guardando el HTML.  
Se deben tener tres imágenes para continuar con los siguientes pasos.
2. En la herramienta de edición, se abre *L2-mascotas-imagen.html* si aún no se ha hecho.
3. Se reemplaza el marcado de la figura que sigue al elemento `h1` por el siguiente, reemplazando los nombres de los archivos de imagen por los nombres de sus propios archivos de imagen:

```
<!doctype html>
<html>
<body>
  <h1>
```

```
<center>Cuidado y Alimentación</center>

</h1>

<figure>

  
  
  
  <figcaption>Perros felices son perros buenos</figcaption>

</figure>

<p>
  Su perro es un amigo de por vida. ¿Por qué no ofrecer el mejor cuidado posible?
</p>

<p>
  Asegúrese de que su mascota tenga suficiente agua fresca.
  en tiempo caluroso. Cuando lleve a su perro a dar largos paseos,
  lleve un recipiente para el agua donde pueda beber el perro.
  Usted puede encontrar estos recipientes en muchas tiendas de mascotas
  por 10 euros o menos.
</p>
</body>
</html>
```



- Se guarda el archivo como *L2-mascotas-multipimage.html* y se visualiza en un navegador Web.

## Cuidado y Alimentación



Perros felices son perros buenos

Su perro es un amigo de por vida. ¿Por qué no ofrecer el mejor cuidado posible?

Asegúrese de que su mascota tenga suficiente agua fresca, en tiempo caluroso. Cuando lleve a su perro a dar largos paseos, lleve un recipiente para el agua donde pueda beber el perro. Usted puede encontrar estos recipientes en muchas tiendas de mascotas por 10 euros o menos.

- Se cierra el navegador y el archivo si se va a continuar con la siguiente sección.

Lo que se ha hecho en este apartado es simplemente concatenar más imágenes, cada una con la etiqueta `<img>` y todo bajo el contexto de la etiqueta `<figure>`.

## 1.5. Creación de Gráficos con Canvas

### 1.5.1. Canvas Básico

#### a. Uso de Canvas para Formas

Para utilizar el elemento de lienzo para crear una forma, realice los siguientes pasos:

- En una herramienta de edición, se escribe el siguiente código:

```
<!doctype html>
<html>
<head>
  <meta charset="UTF-8">
  <title>Prueba Canvas</title>
  <script>
```

```
function f1
{
    var canvas =document.getElementById("smlRectangle");
    context = canvas.getContext("2d");
    context.fillStyle = "rgb(0,0,255)";
    context.fillRect(10, 20, 200, 100);
}
</script>
</head>
<body onload = "f1();">
    <canvas id="smlRectangle" height="100" width="200 ">
    </canvas>
</body>
</html>
```

El atributo **onload** hace que se ejecute la función JavaScript (que veremos en capítulos posteriores) en el script. Primero, se encuentra el elemento con el id **smlRectangle**:

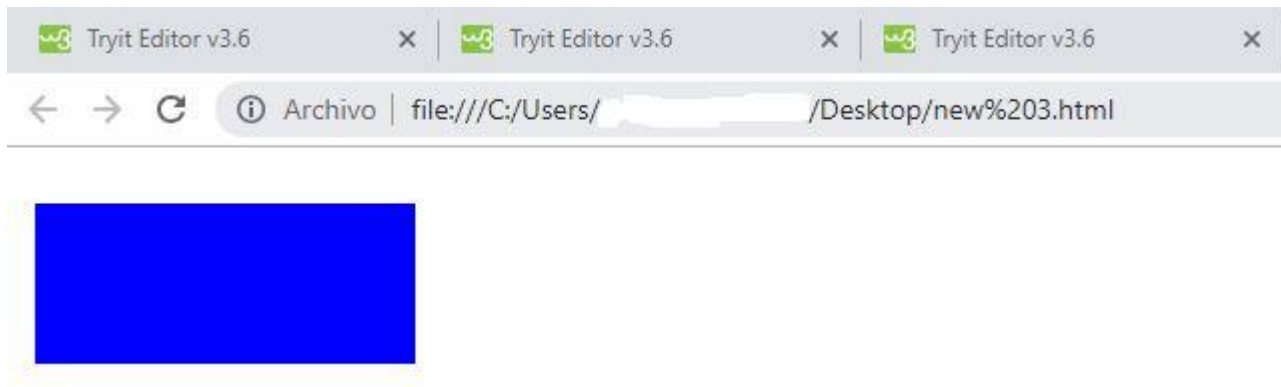
```
var canvas = document.getElementById("smlRectangle");
```

El método **context.fillStyle** rellena el rectángulo con un color azul utilizando el método **rgb** valores 0, 0, 255. El método **context.fillRect** crea un rectángulo de 200 x 100 píxeles, situado 10 píxeles hacia abajo y a 20 píxeles hacia arriba desde la esquina superior izquierda del canvas y lo rellena usando el color especificado por **fillStyle**.

Lo que hace la función primeramente es cargar el **body**, una vez se ha cargado, llama a la función **f1**, que lo que hace es guardar en la variable **canvas** el elemento que tiene por Id, **smlRectangle**, a continuación, en la segunda línea de la función se indica que tiene que ser una figura de dos dimensiones. En la tercera línea se indica, con el método RGB, el color que debe tener el canvas, para aclarar más lo que es **rgb**, el siguiente enlace: [https://www.w3schools.com/colors/colors\\_rgb.asp](https://www.w3schools.com/colors/colors_rgb.asp). Y en la cuarta se indica donde se debe situar el canvas que se ha creado.



2. Se guarda el archivo como *L2-canvas.html* y se visualiza en un navegador. Se debe mostrar como en la figura siguiente.



3. Si no aparece un rectángulo azul, se debe ir a la Web del Servicio de validación de marcas del W3C. (<http://validator.w3.org>). Se sube *L2-canvas.html* y se hace clic en *Comprobar* para tener el servicio, se comprueba. Se corrige cualquier error reportado por el verificador. Se vuelve a guardar el archivo y se ve en un navegador.
4. Se deja abierto el archivo, la herramienta de edición y el explorador web si desea continuar con el siguiente ejercicio.

### 1.5.2. Uso de Canvas para la Creación de Contornos

#### a. Uso de Canvas

Para utilizar el elemento de canvas para crear el contorno de una forma, realice los siguientes pasos:

1. En una herramienta de edición, se guarda *L2-canvas.html* como *L2-canvas-nuevo.html*.
2. Se reemplazan las líneas de código `fillStyle` y `fillRect` por las siguientes:

```
context.strokeStyle = "#FF0000"; //Otra forma de dar los colores en el canvas  
context.strokeRect(10,20,200,100);
```

3. Se eliminan los atributos de anchura y altura del elemento de lienzo en el cuerpo (después de `id="smlRectangle"`).
4. Se guarda el fichero y se visualiza en un navegador Web. Debería mostrarse como en figura siguiente:



5. Se cierra el archivo, pero se deja la herramienta de edición y el explorador web abiertos si desea continuar con el siguiente ejercicio.

### 1.5.3. Alternativa Hacer Imágenes y Textos para Viejos Navegadores

#### a. Añadir Fallback a HTML

Para añadir una copia de seguridad a un documento *HTML*, se pueden realizar los siguientes pasos:

1. En una herramienta de edición, se abre *L2-canvas.html* y se guarda como *L2-canvas-canvas-fallback.html*.
2. Se sustituye el elemento canvas por el siguiente:

```
<canvas id="sm1Rectangle" height="100" width="200">  
    Su navegador no soporta la etiqueta de canvas.  
</canvas>
```

3. Se guarda el archivo y se visualiza en el navegador de *Internet Explorer 9*. Se debería ver el icono de dibujo en canvas.
4. Se pulsa *F12* para entrar en el modo navegador, se hace clic en la pestaña de *Emulación* en la barra de menús y se seleccionan los estándares de *Internet Explorer 7*.
5. Se vuelve a presionar *F12*. Aparece un mensaje de error que indica que no reconoce a una propiedad o método. Se cierra el mensaje de error. La ventana del navegador muestra un texto alternativo, como se muestra en la figura siguiente.

---

Su navegador no soporta la etiqueta de canvas.



6. Se deja abierto el archivo, la herramienta de edición y el explorador web si se desea continuar con el siguiente ejercicio.

## 1.6. Creando Gráficos con SVG

### CREANDO UN GRÁFICO DE VECTORES CON SVG:

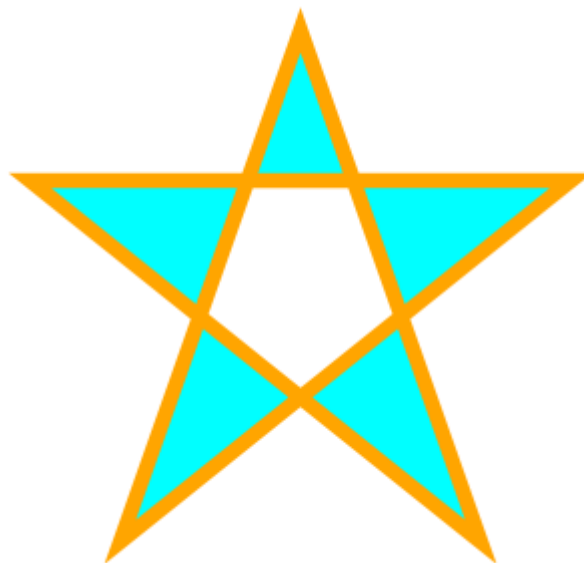
Para crear un gráfico vectorial SVG simple, realice los siguientes pasos:

1. En una herramienta de edición, se escribe el siguiente código:

```
<!doctype html>
<html>
<head>
  <meta charset="UTF-8">
  <title>SVG Estrella</title>
</head>
<body>
  <svg xmlns="http://www.w3.org/2000/svg" version="1.1">
    <polygon points="100,10 40,180 190,60 10,60 160,180"
    style="fill:aqua;stroke:orange;stroke-width:5; fill-rule:evenodd;" />
  </svg>
</body>
</html>
```

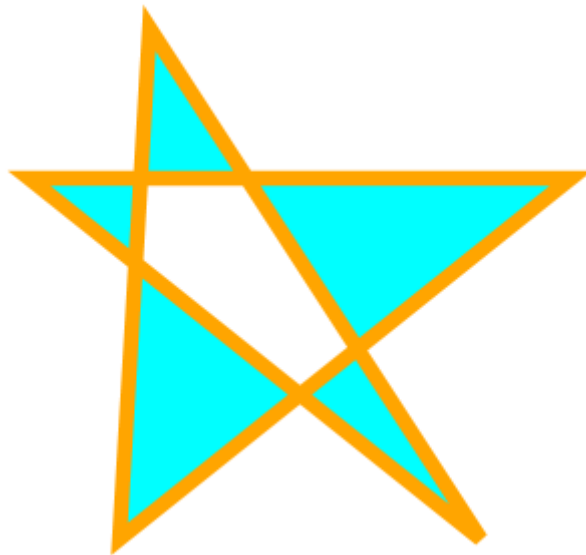
El atributo **polygon points** define las coordenadas x e y para cada esquina, o "punto", del polígono. La regla de relleno determina cómo se rellena el interior del polígono.

**Fill:aqua** sirve para darle ese color turquesa a la figura; con el **stroke** y **orange** se colorean los bordes de color naranja, de anchura 5. Y con el **fill-rule:evenodd** se consigue pintar todo salvo el centro del polígono. Se guarda el archivo como *L2-SVG.html* y se visualiza en un navegador Web. La página debe tener el aspecto de similar al de la figura siguiente.



Si la página no aparece, se comprueba con la marca de revisión de *W3C Servicio de Validación* en <http://validator.w3.org> y se corrige cualquier error.

2. Se cambian algunos de los valores de los parámetros de los puntos poligonales. Se guarda el archivo como *L2-SVG-test.html* y se visualiza en un navegador web. Por ejemplo, cambiando el primer valor del parámetro de 100 a 50 produce el polígono mostrado en la figura siguiente:



4. Se comparan los gráficos que se han generado.
5. Se cierran todos los archivos de datos abiertos. Se deja la herramienta de edición y el explorador web abiertos si se pretende continuar con el siguiente ejercicio.

## 1.7. Entendiendo las Etiquetas de Vídeo y Audio

### 1.7.1. Trabajando con el Elemento Vídeo

Para trabajar con el elemento de vídeo *HTML5*, siga estos pasos:

1. Se busca un clip de vídeo y un archivo de imagen para utilizarlo como póster. Si no se tiene un vídeo, se puede buscar y descargar un archivo *MP4* de dominio público en la Web. Se guarda el vídeo y el archivo de imagen a una carpeta *HTML5*.

2. En una herramienta de edición, se crea un archivo *HTML* con el siguiente código. Se modifica el atributo *type*, si es necesario, y se reemplaza *ejemplo.mp4* y *ejemplo.jpg* con el nombre del archivo de vídeo e imagen correspondientes.

```
<!doctype html>
<html>
<head>
  <meta charset="UTF-8">
  <title>Video Prueba</title>
</head>
<body>
  <video
    width="400" height="300"
    poster="ejemplo.jpg"
    autoplay="autoplay"
    controls="controls">
    <source src="ejemplo.mp4" type="video/mp4" />
  </video>
</body>
</html>
```

3. Se guarda el archivo como *L2-video.html*.
4. Se entra en la página web del *Servicio de validación de marcas del W3C* en <http://validator.w3.org>. Se sube *L2-video.html* y se hace clic en *Check* para que el servicio lo compruebe. Se corrige cualquier error reportado, si los hay.
5. Se abre el archivo *HTML* en un navegador Web.
6. En una herramienta de edición, se elimina la línea de reproducción automática y se reemplaza **controls="controls"** por simplemente **controls**.
7. Se guarda el archivo de nuevo y se valida. Debe validarse sin errores. Eso indica que *HTML5* le permite utilizar un método abreviado para especificar el atributo *controls*. El mismo principio se aplica a los atributos de reproducción automática y bucle.
8. Se deja abierta la herramienta de edición y el explorador web si se va a continuar con el próximo ejercicio.

### 1.7.2. Trabajando con el Elemento Audio

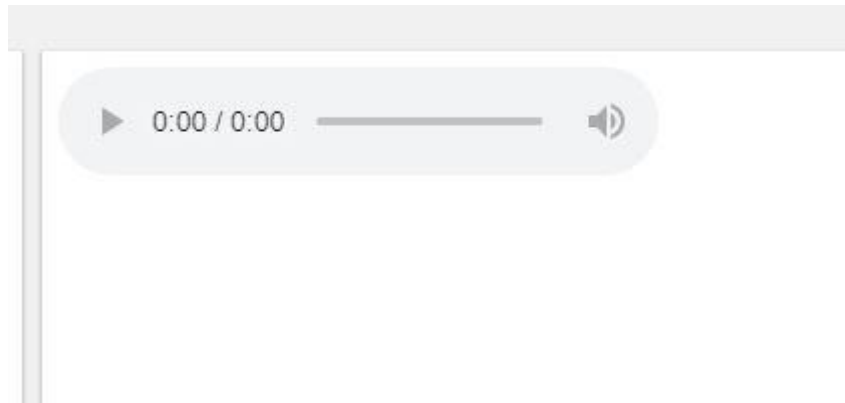
Para trabajar con el elemento de audio HTML5, siga estos pasos:

1. Se localiza un clip de audio.
2. En una herramienta de edición, se crea un archivo *HTML* con el siguiente código:



```
<!doctype html>
<html>
<head>
  <meta charset="UTF-8">
  <title>Audio Prueba</title>
</head>
<body>
  <audio src="ejemplo.mp3" controls="controls">
  </audio>
</body>
</html>
```

3. Se guarda el archivo como *L2-audio.html* y se visualiza en un navegador. Se debería ver algo similar a la figura siguiente. Como no se ha incluido el atributo de reproducción automática en este archivo, se debe hacer clic en el botón Reproducir para iniciar el clip de audio.



4. Si los controles de audio no aparecen, se debe ir a la Web del *Servicio de validación de marcas del W3C*, <http://validator.w3.org>. Se sube *L2-audio.html* y se hace clic en *Check* para que se muestre el archivo de servicio, se comprueba. Se corrige cualquier error reportado, si los hubiera.
5. Se guarda el archivo de nuevo y se abre en un navegador Web. Se reproduce el clip de audio.
6. Se cierran todos los archivos abiertos, se incluyendo la herramienta de edición y el explorador web.

## 2. Tutorial Capítulo 3: Construyendo la Interfaz de Usuario mediante HTML5: Organización, Valores de Entrada y Validación

---

### 2.1. Eligiendo y Configurando las Etiquetas de HTML5 para Organizar Contenidos y Formularios

#### 2.1.1. Creando un Documento HTML con Header, Section y Footer

Para crear un documento *HTML* utilizando el encabezado(**header**), la sección(**section**) y el pie de página(**footer**) en *HTML5* se pueden realizar los siguientes pasos:

1. Se usa un editor *HTML* o una herramienta de desarrollo de aplicaciones y un navegador Web. Se crea un archivo Documento *HTML* que incorpore las etiquetas **<header>**, **<section>**, y **<footer>**. Se incluyen dos secciones, y se asegura de incluir al menos un elemento **h1** dentro de las secciones. Se pueden incluir imágenes si se desea. El HTML, se puede parecer al siguiente:

```
<!doctype html>
<html>
<head>
  <meta charset="utf-8" />
  <title>Mi Página</title>
</head>
<body>
  <header>
    <h1>Selección de estilos de bailes</h1>
  </header>
  <section>
    <h1>Bachata</h1>
    <p>La bachata es un género musical bailable originario de la República Dominicana, dentro de lo que se denomina folclore urbano. Está considerado como un derivado del bolero rítmico, influenciado por otros estilos como el son cubano y el merengue. </p>
  </section>
  <section>
    <h1>Kizomba</h1>
    <p>La base musical de la kizomba es fácilmente reconocible. Tocándolo con las palmas de la mano se correspondería con la siguiente secuencia: Una palmada fuerte (que correspondería con el tiempo 1 de la música), seguida de dos palmadas más suaves y muy juntas en el tiempo (la primera de ellas sería una nota sincopada y la segunda correspondería con el tiempo 2 de la música) y por último una tercera palmada también más suave separada de las anteriores (que también sería sincopada).</p>
  </section>
  <footer>
    <p>Author: Anónimo</p>
  </footer>
</body>
</html>
```

Esta marca de revisión se mostraría en una página Web como se muestra en la siguiente figura.

# Selección de estilos de bailes

## Bachata

La bachata es un género musicalailable originario de la República Dominicana, dentro de lo que se denomina folclore urbano. Está considerado como un derivado del bolero rítmico, influenciado por otros estilos como el son cubano y el merengue.

## Kizomba

La base musical de la kizomba es fácilmente reconocible. Tocándolo con las palmas de la mano se correspondería con la siguiente secuencia: Una palmada fuerte (que correspondería con el tiempo 1 de la música), seguida de dos palmadas más suaves y muy juntas en el tiempo (la primera de ellas sería una nota sincopada y la segunda correspondería con el tiempo 2 de la música) y por último una tercera palmada también más suave separada de las anteriores (que también sería sincopada).

Author: Anónimo

2. Se guarda el archivo como *L3-MiPagina.html*.
3. Se valida el documento utilizando el Servicio de Validación de Marcas del W3C en <http://validator.w3.org>. Si se necesita ayuda para hacerlo, se puede consultar la Lección 2.
4. Se deja la herramienta de edición y el explorador web abiertos si se pretende continuar con el siguiente ejercicio.

### 2.1.2. Añadiendo el Elemento Nav a un Documento HTML

Para añadir el elemento de navegación a un documento *HTML*, se pueden realizar lo siguiente pasos:

1. En el editor *HTML* o herramienta de desarrollo de aplicaciones, se abre el archivo *L3-MiPagina.html* (si aún no está abierto) y se guardará como *L3-MiPagina-nav.html* para crear un archivo nuevo.
2. Se incluyen las siguientes etiquetas de navegación `<nav>` y contenido dentro de la etiqueta `<header>`:

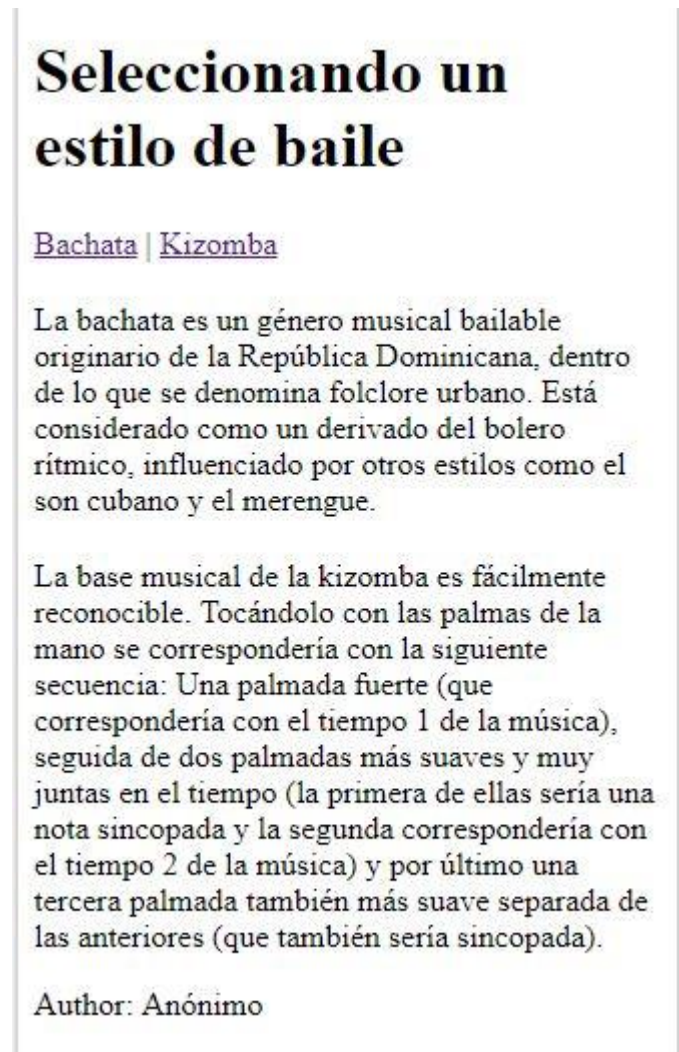
```
<header>
  <h1>Seleccionando un estilo de baile</h1>
  <nav>
    <a href="#bachata">Bachata</a> |
    <a href="#kizomba">Kizomba</a>
  </nav>
</header>
```

3. Para que los enlaces funcionen, modifique los encabezados Bachata y Kizomba `<h1>` como sigue:



```
<h1><a id=" bachata "> Bachata </a></h1>  
<h1><a id=" kizomba "> Kizomba </a></h1>
```

4. Se guarda el archivo como *L3-MiPagina-nav.html* y luego se abre en un navegador Web. En los enlaces de navegación aparecerían en una página Web como se muestra en la siguiente figura.



5. Se deja la herramienta de edición y el explorador web abiertos si se continúa con el siguiente ejercicio durante esta sesión.

### 2.1.3. Añadiendo el Elemento Aside a un Documento HTML

Para añadir el elemento **aside** a un documento HTML, se siguen los siguientes pasos:



1. En un editor HTML o herramienta de desarrollo de aplicaciones, se abre el archivo *L3-MiPagina-nav.html* (si no está abierto) y se guarda como *L3-MiPaginaArte-aside.html* para crear un nuevo archivo. Se incluye un elemento aparte, justo antes del pie de página, como se indica a continuación:

```
<aside>
    <hr/>
    <p>
        "Últimamente registramos un gran interés, pero hemos aumentado nuestras capacidades
        y, si se apunta a tiempo, podremos incluirle en un curso", dijo una representante de la escuela
        de baile "Elite", en donde enseñan a bailar tango, samba, rumba, jive, cha-cha-cha y mambo,
        entre otros ritmos.
    </p>
</aside>
```

2. Se guarda el archivo como *L3-MiPaginaArte-aside.html* y se ve en un navegador Web. La página debe ser similar a la figura siguiente.

## Seleccionando un estilo de baile

[Bachata](#) | [Kizomba](#)

La bachata es un género musical bailable originario de la República Dominicana, dentro de lo que se denomina folclore urbano. Está considerado como un derivado del bolero rítmico, influenciado por otros estilos como el son cubano y el merengue.

La base musical de la kizomba es fácilmente reconocible. Tocándolo con las palmas de la mano se correspondería con la siguiente secuencia: Una palmada fuerte (que correspondería con el tiempo 1 de la música), seguida de dos palmadas más suaves y muy juntas en el tiempo (la primera de ellas sería una nota sincopada y la segunda correspondería con el tiempo 2 de la música) y por último una tercera palmada también más suave separada de las anteriores (que también sería sincopada).

Author: Anónimo

"Últimamente registramos un gran interés, pero hemos aumentado nuestras capacidades y, si se apunta a tiempo, podremos incluirle en un curso", dijo una representante de la escuela de baile "Elite", en donde enseñan a bailar tango, samba, rumba, jive, cha-cha-cha y mambo, entre otros ritmos.

3. Se valida el documento utilizando el Servicio de Validación de Marcas del W3C en <http://validator.w3.org>.



4. Se cierra el archivo y, a continuación, se deja la herramienta de edición y el explorador web abiertos si continúa al siguiente ejercicio durante esta sesión.

#### 2.1.4. Creando una Tabla

Para crear una tabla, se pueden realizar los siguientes pasos:

1. Utilizando un editor *HTML* o una herramienta de desarrollo de aplicaciones y un navegador Web, se crea un archivo llamado *L3-PracTabla.html* con el siguiente código:

```
<!doctype html>
<html lang="en" xmlns="http://www.w3.org/1999/xhtml">
<head>
  <meta charset="utf-8" />
  <caption>Las Películas Más Caras de Producir</caption>
</head>
<body>
  <table border="1">
    <tr>
      <th>Películas</th>
      <th>Costes de Producirlas</th>
    </tr>
    <tr>
      <td><i>Avatar</i></td>
      <td>$2.7 billion</td>
    </tr>
    <tr>
      <td><i>Titanic</i></td>
      <td>$2.1 billion</td>
    </tr>
    <tr>
      <td><i>The Dark Knight</i></td>
      <td>$1.0 billion</td>
    </tr>
  </table>
</body>
</html>
```



2. Se pone en cursiva el título de cada película utilizando las etiquetas `<i>`.
3. Se modifica la tabla para añadir un título encima de la tabla que diga "Películas de alto coste", y un **footer** que incluye la palabra "Total" y "5.800 millones de dólares".
4. Se cambia el fondo de toda la tabla, desde las cabeceras de las columnas hasta el pie de página, a caqui, utilizando el código hexadecimal `#F0E68C`. Para ello, se añade la directiva siguiendo las marcas de los grupos de colores entre `<table border="1">` y `<thead>`, como sigue:

```
<thead>
  <table border="1">
    <colgroup span="2" style="background-color:#F0E68C;">
    </colgroup>
```

5. Se guarda el fichero y se visualiza en un navegador web. La tabla terminada debe tener un aspecto similar a la figura.

### Las Películas Más Caras de Producir

Películas	Costes de Producirlas
<i>Avatar</i>	\$2.7 billion
<i>Titanic</i>	\$2.1 billion
<i>The Dark Knight</i>	\$1.0 billion

Total: 5.800 millones de dólares

6. Se cierra el archivo y, se deja la herramienta de edición y el explorador web abiertos si se pretende continuar con el siguiente ejercicio.

#### 2.1.5. Creando una Lista Ordenada

Para crear una lista ordenada, se pueden realizar los siguientes pasos:

1. Utilizando un editor *HTML* o una herramienta de desarrollo de aplicaciones y un navegador Web, se introduce lo siguiente:

```
<!doctype html>
<html>
<body>
  <ol>
```

```
<li>Ordenadores</li>
<li>Portátiles</li>
<li>Tablets</li>
<li>Smartphones</li>
</ol>
</body>
</html>
```

2. Se guarda el archivo como *L3-ListaOrdenada.html* y se visualiza en un navegador Web.
3. Para cambiar los marcadores iniciales a mayúsculas, se inserta el **type="A"** en la etiqueta **<ol>**, de esta manera: **<ol type="A">**
4. Se guarda el fichero y se abre en un navegador Web.
5. Para comenzar la numeración de la lista en 5, se inserta **start="5"** en la etiqueta **<ol>**. Ahora, se reemplaza el actual **<ol>** con esto: Inicio **<ol start="5">**
6. Se guarda el fichero y se ve en un navegador Web.
7. Se cierra el archivo y, a continuación, se deja la herramienta de edición y el explorador web abiertos si se pretende continuar con el siguiente ejercicio.

## 2.2. Eligiendo y Configurando las Etiquetas de HTML5 para Valores de Entrada y Validación:

### 2.2.1. Entendiendo los Elementos de Entrada y Formularios

a. Explorando Formas de Creación, Atributos para Elementos de Entrada y Valores

#### CREANDO UN FORMULARIO WEB SENCILLO

Para crear un formulario Web simple, se realizan los siguientes pasos:

1. Utilizando un editor HTML o una herramienta de desarrollo de aplicaciones y un navegador Web, se crea un archivo Formulario web con el siguiente código:

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <caption><center>Contáctanos</center></caption>
</head>
<body>
  <div id="contact-form">
    <form id="contact" method="post" action="">
      <fieldset>
```



```

<label for="custname">Nombre</label>
<input type="text" id="custname" />
<label for="email">Correo Electrónico</label>
<input type="email" id="email" />
<label for="phone">Teléfono</label>
<input type="text" id="phone" />
<label for="message">Dudas o Comentarios</label>
<textarea name="message"></textarea>
<input type="submit" name="submit" id="submit"
      value="Enviar" />
</fieldset>
</form>
</div><!-- Fin de los datos del formulario -->
</body>
</html>

```

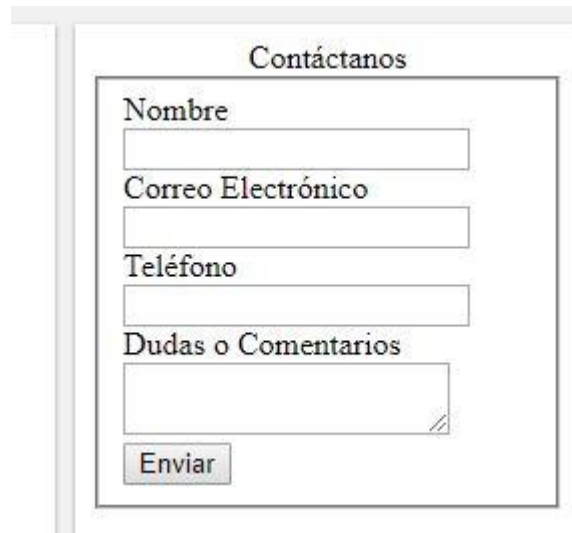
Se van añadiendo, etiquetas a través de la etiqueta **label** y botones con la etiqueta **input**, a los campos que se va creando, con un id determinado y su tipo: **text**, **email**, **phone**, **mesasge** o **submit**. El value de submit sirve para indicar lo que va a aparecer en el botón que sirve para enviar el formulario.

2. Se guarda el archivo como *L3-WebForm-orig.html*. La versión renderizada se muestra en la siguiente figura.

The screenshot shows a web browser displaying a form titled "Contáctanos". The form has four input fields: "Nombre" with the value "PEPPITO", "Correo Electrónico" with the value "pepito123@hotmail.com", "Teléfono" with the value "123456789", and "Dudas o Comentarios" with the value "Ninguno". An "Enviar" button is located at the bottom right of the form. The form is styled with a simple border and a light background.

3. El formulario Web parece desestructurado. Idealmente, se usaría **CSS** para aplicar la alineación, pero como aún no se ha aprendido **CSS**, se puede aplicar una solución para hacer alinear los campos verticalmente. Un método consiste en añadir etiquetas **<fieldset>** iniciales y finales alrededor de cada una de ellas por etiqueta de entrada y salida. Esto alinearía los campos verticalmente y añadiría cajas alrededor de ellos. Usando las etiquetas de abrir y cerrar de tipo **<p>** en lugar de **<fieldset>** se lograría lo mismo, pero sin añadir cajas. Para este ejercicio, se usan las etiquetas **<p>**. La figura muestra el mismo formulario Web con etiquetas **<p>** alrededor de los pares etiqueta, incluyendo el campo de comentarios.

4. Se agrega texto de marcador de posición a todos los campos. El resultado debe ser similar al de la figura siguiente, si en el navegador web *Mozilla Firefox*.



The image shows a web form titled "Contáctanos" (Contact Us). It contains four input fields with placeholder text: "Nombre" (Name), "Correo Electrónico" (Email), "Teléfono" (Phone), and "Dudas o Comentarios" (Questions or Comments). Below the fields is a button labeled "Enviar" (Send).

5. Se guarda el archivo como *L3-FormularioWeb.html*.
6. Se deja el archivo y la herramienta de edición abiertos si se va a continuar con el siguiente ejercicio.

#### b. Entendiendo Validaciones

### AÑADIENDO CAMPOS DE VALIDACIONES A UN FORMULARIO WEB

Para añadir campos de validación a un formulario Web, se hacen los pasos siguientes:

1. Utilizando un editor *HTML* o una herramienta de desarrollo de aplicaciones, se abre *L3-FormularioWeb.html*.
2. Se guarda el archivo como *L3-Formulario-validando.html*.
3. Se añade el atributo **required** al campo de correo electrónico, como se indica a continuación, el cual nos indica que ese campo deberá ser rellenado obligatoriamente, antes de que el formulario pueda ser enviado:

```
<p>  
  <label for="email">Email</label>  
  < input type="correo electrónico" nombre="correo electrónico" required  
    placeholder="Dirección de correo electrónico">  
</p>
```

4. Se añade el atributo de patrón al campo del teléfono. La expresión debería restringir la entrada al código de área y número de teléfono, en el formato XXX-XXX-XXXXXX y además usando el placeholder se indica que por defecto aparecerá Número de teléfono en el campo Teléfono del formulario, como sigue:

```
<p>  
    <label for="phone">Teléfono</label>  
    <input type="text" name="phone" pattern="[0-9]{3}-[0-9]{3}-[0-9]{4}" placeholder="Número  
de teléfono">  
</p>
```

5. Se guarda el fichero y, a continuación, se visualiza en un navegador Web. Se escribe el texto en cada campo de entrada excepto el campo de correo electrónico y se hace clic en el botón Enviar. ¿Se recibe un mensaje de error que le pide que introduzca una dirección de correo electrónico?
6. Se escribe el texto en cada campo de nuevo incluyendo el campo de correo electrónico, pero esta vez escriba un teléfono sin el código de área y luego haga clic en Enviar. ¿Se ha recibido un error con respecto al campo de número de teléfono?
7. Se cierra el archivo, la herramienta de edición o de desarrollo de aplicaciones y el explorador web.



## 3. Tutorial Capítulo 4: Entendiendo CSS Básico: Contenido Flotante, Posicionamiento y Estilos

### 3.1. Explorando la Conexión entre HTML y CSS

#### UN USO SIMPLE DE CSS CON HTML:

A continuación, un ejemplo simple de CSS con HTML:

1. Se usa un editor de texto o una herramienta de desarrollo para crear un fichero en el directorio llamado *e1.html* con el siguiente contenido, consta de un encabezado "Tierras Lozano", un párrafo "Te mantenemos verde" que tiene un identificador *eslogan* que cogerá estilos del fichero css y otro párrafo "Tierras Lozano puede mantener...":

```
<!doctype html>
<html>
<head>
  <meta charset="UTF-8">
  <title>Tierras Lozano.</title>
  <link href = "e1.css" rel = "stylesheet" type = "text/css">
</head>
<body>
  <h1>
    Tierras Lozano.
  </h1>
  <p id = "eslogan">
    Te mantenemos verde.
  </p>
  <p>
    Tierras Lozano puede mantener su tierra vigorosa durante toda
    la temporada. Utilizamos solamente fertilizantes naturales para
    mejorar la salud de su tierra.
  </p>
</body>
</html>
```

2. Se crea un segundo fichero, en la misma carpeta que el *.html*, llamado *e1.css* y se usa el siguiente contenido, con un tamaño de fuente (20px), un color (verde) y un estilo (cursiva):

```
#eslogan {
  font-size: 20px;
  color: green;
  font-style: italic;
}
```

3. Se abre el fichero *e1.html* en el navegador web, la página debe parecerse a la de la siguiente figura:



## Tierras Lozano.

*Te mantenemos verde.*

Tierras Lozano puede mantener su tierra vigorosa durante toda la temporada. Utilizamos solamente fertilizantes naturales para mejorar la salud de su tierra.

En este ejemplo, el archivo *HTML* (*e1.html*) define el contenido y la estructura: tiene las palabras "Tierras Lozano", que identifica dos palabras como parte de un eslogan, y así sucesivamente. El archivo *CSS* (*e1.css*) proporciona estilo a ese contenido. Para que ciertos caracteres sean verdes, la palabra "green" para comunicar al navegador cómo colorear el eslogan. Los archivos se enlazan entre sí mediante el elemento `<link>` del archivo *HTML*.

**NOTA:** Se puede especificar el color usando un nombre o un número hexadecimal. Por ejemplo, para usar el color azul, el nombre del color es "blue" y su valor hexadecimal es #000FF. Se puede obtener la lista de nombres y valores en [http://www.w3schools.com/cssref/css\\_colornames.asp](http://www.w3schools.com/cssref/css_colornames.asp).

## CREAR UNA PÁGINA WEB BÁSICA Y UN ARCHIVO CSS

A continuación, se va a crear una página web y un archivo *CSS*; se va a modificar el *CSS* para ver cómo afectan los cambios a la web. Se entenderá como *HTML* y *CSS* trabajan juntos y producen vistas en el navegador web o aplicación móvil:

1. Se crea una página web llamada *e1.html* y un archivo *CSS* llamado *e1.css* usando las marcas mostradas anteriormente.
2. Se abre con el navegador web el archivo *e1.html* para mostrar la página renderizada.
3. Se edita *e1.css* para hacer el eslogan aparecer en una fuente más grande, como **25px**. Se cambia el color de verde a **#00CC00**. Se cambia el estilo a **negrita**. Después de hacer cada cambio, se confirma que la vista correspondiente se actualiza como se espera. La página web debe quedar como el de la siguiente figura:

## Tierras Lozano.

*Te mantenemos verde.*

Tierras Lozano puede mantener su tierra vigorosa durante toda la temporada. Utilizamos solamente fertilizantes naturales para mejorar la salud de su tierra.

4. Se cierran los archivos *HTML* y *CSS*. Se deja la herramienta de edición y el navegador web abiertos si se va a continuar con la siguiente sección.

## 3.2. Entendiendo Fuentes y Familias de Fuentes

### EXPERIMENTANDO CON FAMILIAS DE FUENTES:

En el siguiente ejemplo se va a experimentar con familias de fuentes:





1. Se abren los archivos *e1.html* y *e1.css* de los ejercicios previos en la herramienta de edición y se guardan como *e3.html* y *e3.css*.
2. Se cambia *e1.css* a *e3.css* en el elemento `link` del archivo HTML.
3. Dentro de *e3.css*, se añade una nueva regla para `#eslogan` que declare la fuente `font-family: monospace;`

```
p {color: brown;}
#eslogan {
  font-family: monospace;
  font-size: 20px;
  color: green;
  font-style: italic;
}
```

4. Se guarda el archivo CSS y se mira el archivo *HTML* en el navegador. Los resultados se ven en la siguiente imagen:

## Tierras Lozano.

*Te mantenemos verde.*

Tierras Lozano puede mantener su tierra vigorosa durante toda la temporada. Utilizamos solamente fertilizantes naturales para mejorar la salud de su tierra.

5. En el archivo CSS, se debe cambiar la fuente a `font-family: sans-serif;`
6. Se guarda el archivo CSS, y se mira la vista *HTML* en el navegador. Los resultados se ven en la siguiente imagen:

## Tierras Lozano.

Te mantenemos verde.

Tierras Lozano puede mantener su tierra vigorosa durante toda la temporada. Utilizamos solamente fertilizantes naturales para mejorar la salud de su tierra.

7. Se cambia la fuente `font-family: Garamond;` se guarda el archivo CSS y se observa la vista *HTML* en el navegador.
8. Se cierran los archivos *e3.html* y *e3.css*.

### 3.3. Manejando Contenido Flotante

#### EXPLORANDO INLINE FLOW Y BLOCK FLOW:

En el siguiente ejemplo, se va a trabajar usando el elemento `<style>`:

1. Se crea el fichero *e4.html* con el siguiente contenido, con una lista desordenada *ul*:

```
<!doctype html>
```



```
<!-- Este es el contenido del archivo e4.html.-->
<html>
  <head>
    <meta charset="UTF-8">
    <title>Block e inline flow</title>
    <link href = "e4.css" rel = "stylesheet" type = "text/css">
    <style type = 'text/css'>
      .toolbar li {
      }
    </style>
  </head>
  <body>
    <h1>Block e inline flow</h1>
    <p> Estas son las opciones que puedes tomar:</p>
    <ul class = "toolbar">
      <li>Automovil</li>
      <li>Bicicleta</li>
      <li>Scooter</li>
      <li>Taxi</li>
      <li>Caminar</li>
    </ul>
  </body>
</html>
```

2. En la siguiente figura se observa el resultado que se vería en el navegador:

## Block e inline flow

Estas son las opciones que puedes tomar:

- Automovil
- Bicicleta
- Scooter
- Taxi
- Caminar

3. Se actualiza la fuente de *e4.html* tal que el segmento de `<style>`, mostrando la lista en horizontal con el comando `display: inline;`, con un color de fondo, borde y márgenes, tal cómo se muestra en el siguiente código:

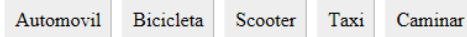
```
<style type = "text/css">
  .toolbar li {
    display: inline;
    background-color: #EEE;
    border: 1px solid;
    border-color: #F3F3F3 #BBB #BBB #F3F3F3;
    margin: 2px;
    padding: .5em;
  }
</style>
```



4. Se guarda el fichero y se refresca el navegador. La lista de actualizaciones se puede ver en la siguiente imagen.

## Block e inline flow

Estas son las opciones que puedes tomar:



### 3.4. Posicionando Elementos Individuales

#### 3.4.1. Aplicando Posicionamiento Flotante

#### USANDO POSICIONAMIENTO FLOAT CON MULTICOLUMNAS:

Para aplicar posicionamiento flotante de múltiples columnas, seguir los siguientes pasos:

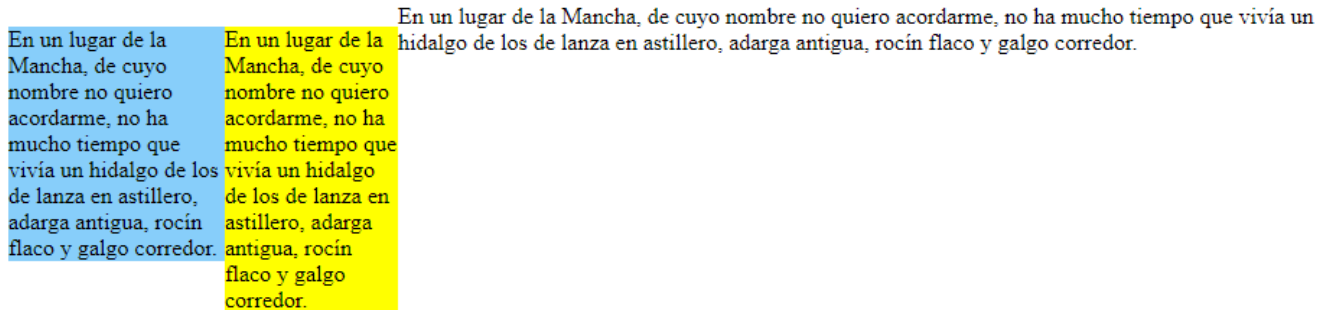
1. Se crea el fichero *e5.html* con el siguiente contenido, dentro de los estilos, se definen dos columnas con la propiedad `float: left;`, que dejará el contenido de las columnas flotando a la izquierda:

```
<!doctype html>
<!-- This is e5.html. -->
<html>
<head>
    <meta charset="UTF-8">
    <title>Posicionamiento Float</title>
    <style type = 'text/css'>
        #col1 {
            float: left;
            width: 150px;
            background-color: lightskyblue;
        }
        #col2 {
            float: left;
            width: 120px;
            background-color: yellow;
        }
    </style>
</head>
<body>
    <h1> Posicionamiento Float </h1>
    <p id = "col1">En un lugar de la Mancha, de cuyo nombre no quiero
acordarme, no ha mucho tiempo que vivía un hidalgo de los de lanza en astillero,
adarga antigua, rocín flaco y galgo corredor.
    <p id = "col2">En un lugar de la Mancha, de cuyo nombre no quiero
acordarme, no ha mucho tiempo que vivía un hidalgo de los de lanza en astillero,
adarga antigua, rocín flaco y galgo corredor.
    <p id = "col3">En un lugar de la Mancha, de cuyo nombre no quiero
acordarme, no ha mucho tiempo que vivía un hidalgo de los de lanza en astillero,
adarga antigua, rocín flaco y galgo corredor.
</body>
</html>
```



2. Se muestra *e5.html*, como se muestra en la siguiente figura.

## Posicionamiento Float



3. En el navegador, *col1* y *col2* aparecen como dos columnas con el ancho fijado, y la *col3* se rellena en el espacio restante. Si se cambiasen los dos atributos CSS *float* de izquierda a derecha, ¿Cómo se mostraría la vista?
4. Se realiza el cambio.
5. Se cierran los archivos *HTML* y *CSS*. Se deja la herramienta de edición y el navegador web abiertos si se va a continuar con la siguiente sección.

### 3.4.2. Aplicando Posicionamiento Absoluto

## USANDO POSICIONAMIENTO ABSOLUTO CON MULTICOLUMNAS:

Para aplicar posicionamiento absoluto de múltiples columnas, seguir los siguientes pasos:

1. Se crea *e6.html* abriendo *e5.html* guardando una copia de seguridad.
2. Se sustituye el comentario en el top por:

```
<!--Esto es e6.html. -->
```

Se sustituye el contenido de `<head>` con el siguiente contenido, donde se fija la posición de la *col1* con la propiedad `position: absolute`, en el fondo a 100px y a la derecha a 100px:

```
<title>Posicionamiento absoluto</title>
<meta charset="UTF-8">
<style type = 'text/css'>
#col1 {
    position: absolute;
    bottom: 100px;
    right: 100px;
    background-color: lightskyblue;
}
#col2 {
    background-color: yellow;
}
</style>
```



3. Dentro del elemento **<body>**, se cambia **<h1>** por:

# Posicionamiento absoluto </h1>

4. Se muestra *e6.html*, como se ve en la siguiente imagen:

## Posicionamiento absoluto

En un lugar de la Mancha, de cuyo nombre no quiero acordarme, no ha mucho tiempo que vivía un hidalgo de los de lanza en astillero, adarga antigua, rocín flaco y galgo corredor.

En un lugar de la Mancha, de cuyo nombre no quiero acordarme, no ha mucho tiempo que vivía un hidalgo de los de lanza en astillero, adarga antigua, rocín flaco y galgo corredor.

En un lugar de la Mancha, de cuyo nombre no quiero acordarme, no ha mucho tiempo que vivía un hidalgo de los de lanza en astillero, adarga antigua, rocín flaco y galgo corredor.

En este ejemplo, la *col2*, en amarillo, y la *col3*, sin color, aparecen normal, cerca del top de la vista, mientras que la *col3*, acaba en una posición fija en la esquina inferior izquierda, si se cambia el tamaño de la ventana, se puede observar como el párrafo se ajusta.

### 3.5. Manejando Contenido Desbordante

### 3.5.1. Entendiendo Scrolling Desbordante

## TRABAJANDO CON SCROLLING DESBORDANTE:

Para practicar scrolling desbordante, se pueden seguir los siguientes pasos:

1. Se crea *e7.html* con el siguiente contenido, donde existe un texto que se desborda del contenedor que lo contiene, por ello se especifica la propiedad **overflow: scroll**; en la sección de estilos de **#coll**:

```
<!-- This is the content of e7.html. -->  
<!doctype html>  
<html>  
    <head>  
        <meta charset="UTF-8">  
        <title>Scroll overflow</title>  
        <style type = "text/css">  
            #coll {  
                width: 200px;  
                height: 200px;  
                background-color: lightskyblue;  
                overflow: scroll;  
            }
```



```

        #col3 {
            background-color: yellow;
        }
    </style>
</head>
<body>
    <h1>Scroll overflow</h1>
    <p id = 'col1'> En un lugar de la Mancha, de cuyo nombre no quiero
    acordarme, no ha mucho tiempo que vivía un hidalgo de los de lanza en astillero,
    adarga antigua, rocín flaco y galgo corredor. Una olla de algo más vaca que
    carnero, salpicón las más noches, duelos y quebrantos los sábados, lentejas los
    viernes, algún palomino de añadidura los domingos, consuman las tres partes de su
    hacienda.

    </p>
    <p id = 'col2'> En un lugar de la Mancha, de cuyo nombre no quiero
    acordarme, no ha mucho tiempo que vivía un hidalgo de los de lanza en astillero,
    adarga antigua, rocín flaco y galgo corredor. Una olla de algo más vaca que
    carnero, salpicón las más noches, duelos y quebrantos los sábados, lentejas los
    viernes, algún palomino de añadidura los domingos, consuman las tres partes de su
    hacienda.

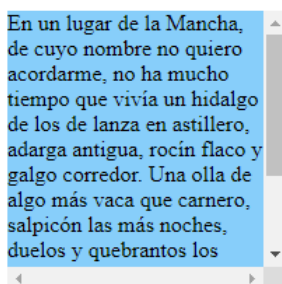
    </p>
    <p id = 'col3'> En un lugar de la Mancha, de cuyo nombre no quiero
    acordarme, no ha mucho tiempo que vivía un hidalgo de los de lanza en astillero,
    adarga antigua, rocín flaco y galgo corredor. Una olla de algo más vaca que
    carnero, salpicón las más noches, duelos y quebrantos los sábados, lentejas los
    viernes, algún palomino de añadidura los domingos, consuman las tres partes de su
    hacienda.

    </p>
</body>
</html>

```

2. Se debe mostrar una vista como la de la siguiente figura:

## Scroll overflow



En un lugar de la Mancha, de cuyo nombre no quiero acordarme, no ha mucho tiempo que vivía un hidalgo de los de lanza en astillero, adarga antigua, rocín flaco y galgo corredor. Una olla de algo más vaca que carnero, salpicón las más noches, duelos y quebrantos los sábados, lentejas los viernes, algún palomino de añadidura los domingos, consuman las tres partes de su hacienda.

En un lugar de la Mancha, de cuyo nombre no quiero acordarme, no ha mucho tiempo que vivía un hidalgo de los de lanza en astillero, adarga antigua, rocín flaco y galgo corredor. Una olla de algo más vaca que carnero, salpicón las más noches, duelos y quebrantos los sábados, lentejas los viernes, algún palomino de añadidura los domingos, consuman las tres partes de su hacienda.

3. Se debe experimentar con el código fuente para ver cómo se renderiza *HTML* en diferentes circunstancias. Por ejemplo, con una anchura de 400 px.
4. Se debe experimentar con el código borrando la mitad del texto de la *col1*.



### 3.5.2. Entendiendo el Desbordamiento Visible y Oculto

## TRABAJANDO CON DESBORDAMIENTO VISIBLE Y OCULTO

Para practicar el desbordamiento visible y oculto, seguir los siguientes pasos:

1. Se abre el archivo *e7.html*
2. Se cambia **overflow** de *e7.html* de **scroll** a **visible**.
3. Se guarda el fichero y observar en el navegador, tal como se vería en la siguiente figura:

### Visible overflow

En un lugar de la Mancha, de cuyo nombre no quiero acordarme, no ha mucho tiempo que vivía un hidalgo de los de lanza en astillero, adarga antigua, rocín flaco y galgo corredor. Una olla de algo más vaca que carnero, salpicón las más noches, duelos y quebrantos los sábados, lentejas los viernes, algún palomino de añadidura

los domingos, de la Mancha, de cuyo nombre no quiero acordarme, no ha mucho tiempo que vivía un hidalgo de los de lanza en astillero, adarga antigua, rocín flaco y galgo corredor. Una olla de algo más vaca que carnero, salpicón las más noches, duelos y quebrantos los sábados, lentejas los viernes, algún palomino de añadidura los domingos, consuman las tres partes de su hacienda.

En un lugar de la Mancha, de cuyo nombre no quiero acordarme, no ha mucho tiempo que vivía un hidalgo de los de lanza en astillero, adarga antigua, rocín flaco y galgo corredor. Una olla de algo más vaca que carnero, salpicón las más noches, duelos y quebrantos los sábados, lentejas los viernes, algún palomino de añadidura los domingos, consuman las tres partes de su hacienda.

**visible** es el valor por defecto de **overflow**. Con **visible** los elementos de la vista de *HTML* están dispuestos en orden, entonces cualquier desbordamiento simplemente solapa los elementos.

4. El **background-color** no se aplica al contenido desbordante.
5. Ahora se cambia **scroll** a **hidden**. La vista se debería ver como la siguiente imagen, ocultando el texto desbordante:

### Hidden overflow

En un lugar de la Mancha, de cuyo nombre no quiero acordarme, no ha mucho tiempo que vivía un hidalgo de los de lanza en astillero, adarga antigua, rocín flaco y galgo corredor. Una olla de algo más vaca que carnero, salpicón las más noches, duelos y quebrantos los sábados, lentejas los viernes,

En un lugar de la Mancha, de cuyo nombre no quiero acordarme, no ha mucho tiempo que vivía un hidalgo de los de lanza en astillero, adarga antigua, rocín flaco y galgo corredor. Una olla de algo más vaca que carnero, salpicón las más noches, duelos y quebrantos los sábados, lentejas los viernes, algún palomino de añadidura los domingos, consuman las tres partes de su hacienda.

En un lugar de la Mancha, de cuyo nombre no quiero acordarme, no ha mucho tiempo que vivía un hidalgo de los de lanza en astillero, adarga antigua, rocín flaco y galgo corredor. Una olla de algo más vaca que carnero, salpicón las más noches, duelos y quebrantos los sábados, lentejas los viernes, algún palomino de añadidura los domingos, consuman las tres partes de su hacienda.



## 4. Tutorial Capítulo 5: Entendiendo CSS Básico: Layouts

### 4.1. Usando una Flexible Box para Establecer Contenido de Alineación, Dirección y Orientación

#### 4.1.1. Trabajando con Flexbox y Flexbox Items

##### α. APLICACIÓN DE ESCALA PROPORCIONAL DENTRO DE UNA CAJA FLEXIBLE

### CREAR UNA FLEXBOX CON ELEMENTOS DE FLEXBOX:

Para aprender a crear una flexbox con elementos de flexbox que tienen una altura fija pero un ancho flexible, se realizan los siguientes pasos:

1. En una herramienta de edición o una app de desarrollo, crear un fichero *HTML* que incluya el siguiente código CSS, donde se define una flexbox con las propiedades que se ven a continuación:

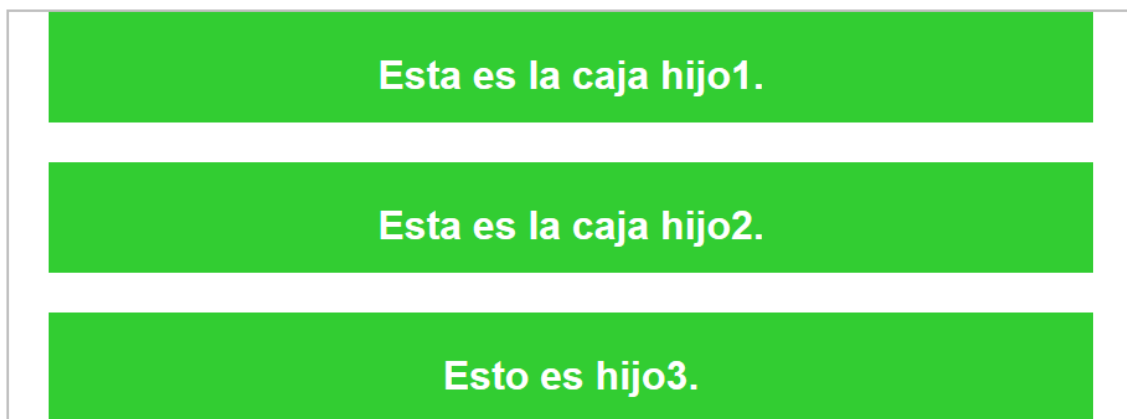
```
<!doctype html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Ejemplo de Child Box flexible</title>
    <style>
      div { display: flexbox;
        outline: 2px solid silver }
      p { flex: 1 auto; margin: 1em;
        font-family: sans-serif;
        color: white;
        background: limegreen;
        height: 25px;
        padding: 1em;
        font-weight: bold;
        font-size: xx-large;
        text-align: center;
      }
    </style>
  </head>
  <body>
    <div>
      <p>Esta es la caja hijo1.</p>
      <p> Esta es la caja hijo2.</p>
      <p>Esto es hijo3.</p>
    </div>
  </body>
</html>
```





La propiedad CSS **display: flexbox** crea la caja padre, la flexbox. A El contorno plateado se crea para la flexbox, lo que simplemente le ayuda a ver la flexbox en la ventana del navegador para este ejercicio. Los estilos del párrafo (p) se aplican a los artículos de la caja flexible (las cajas hijas). La propiedad **flex** aplica flexibilidad a cada una de las cajas hijas. Los elementos tienen un ancho de 75 píxeles. Si sobra espacio en la **flexbox**, cuando el tamaño de la pantalla aumenta, las posiciones de la caja flexible se expanden horizontalmente para llenar el espacio.

2. Se guarda el archivo como *L5-flexbox-exercise.html* y se abre en el navegador web. La vista debería mostrarse como la siguiente imagen:



3. Se cambia el tamaño de la ventana del navegador, haciéndola más estrecha y ancha, arrastrando el borde derecho de la ventana hacia el centro de la pantalla y luego hacia la derecha. Se observa cómo los artículos de la **flexbox** se expanden y encogen junto con la **flexbox**.
4. Se cierra el archivo, pero se deja la herramienta de edición y el explorador web abiertos si se va a continuar con los siguientes ejercicios.

## CREAR ELEMENTOS EN UN FLEXBOX CON LA FUNCIÓN FLEX:

1. En una herramienta de edición o una app de desarrollo, crear un fichero *HTML* que incluya el siguiente código *CSS*, donde se especifica la propiedad **flex-wrap**:

```
<!doctype html>
<html>
<head>
  <meta charset="utf-8" />
  <title>Ejemplo de función Flex</title>
  <style>
    div {
      display: flex;
      display: -ms- flex;
      display: -moz- flex;
      display: -o- flex;
      display: -webkit- flex;
      flex-wrap: wrap;
      -ms-flex-wrap: wrap;
      -moz-flex-wrap: wrap;
      -o-flex-wrap: wrap;
      -webkit-flex-wrap: wrap;
```

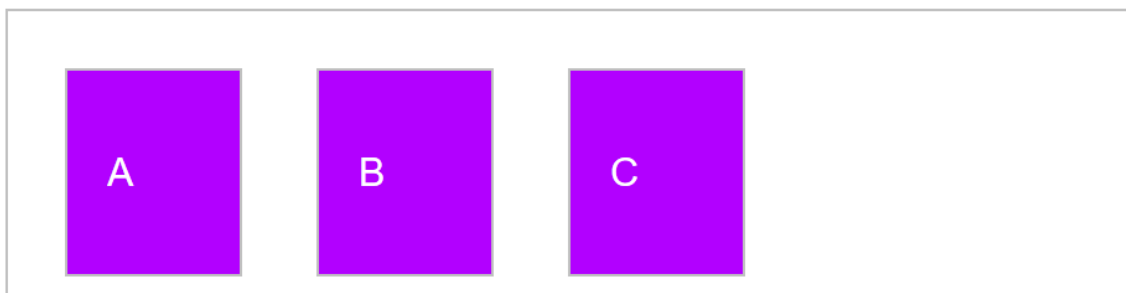
```

        height: 200px;
        padding: 1em;
        color: white;
        outline: 2px solid silver;
    }
    div>div {
        width: 75px;
        width: -ms-flex(1 75px);
        width: -moz-flex(1 75px);
        width: -o-flex(1 75px);
        width: -webkit-flex(1 75px);
        margin: 1em;
        height: 100px;
        background-color: #b200ff;
        font-family: sans-serif;
        text-align: center;
        line-height: 100px;
        font-size: xx-large;
    }
    </style>
</head>
<body>
    <div>
        <div>A</div>
        <div>B</div>
        <div>C</div>
    </div>
</body>
</html>

```

Como en el último ejercicio, la propiedad **display: flex** crea la caja principal. El segundo conjunto de estilos `div>div`, que es simplemente una forma abreviada de aplicar estilos a un de elementos *HTML* sin asignar clases) se aplican a los elementos de la flexbox: la directiva **width** junto con la función **flex** controlan el ancho de los artículos de la flexbox, que tienen un ancho de 75 píxeles pero que llenan cualquier espacio disponible cuando el aumenta el tamaño de la pantalla. La propiedad **flex-wrap** fuerza a los artículos de la caja flexible a estar dentro de la caja flexible.

2. Se guarda el archivo como *L5-flexfunction-exercise.html* y se abre en el navegador web. La vista debería mostrarse como la siguiente imagen:



3. Se reduce ligeramente el ancho de la ventana del navegador arrastrando el borde derecho de la ventana hacia el centro de la pantalla. Se observa que a medida que la flexbox (indicada por el contorno plateado) se encoge, los artículos de la caja flexible se encogen uniformemente en tamaño.
4. Se disminuye aún más el tamaño de la ventana hasta que los artículos de la flexbox se envuelvan.

5. Se abre el archivo en cada uno de los otros navegadores Web principales para ver si el archivo se muestra correctamente.
6. Se cierra el archivo, pero dejar la herramienta de edición y el explorador web abiertos si se completa el siguiente ejercicio en esta sesión.

#### b. CAMBIANDO LA DIRECCIÓN DE LOS ARTÍCULOS INFERIORES EN UNA FLEXBOX

### INVERTIR EL ORDEN DE LOS ARTÍCULOS DE LA FLEXBOX:

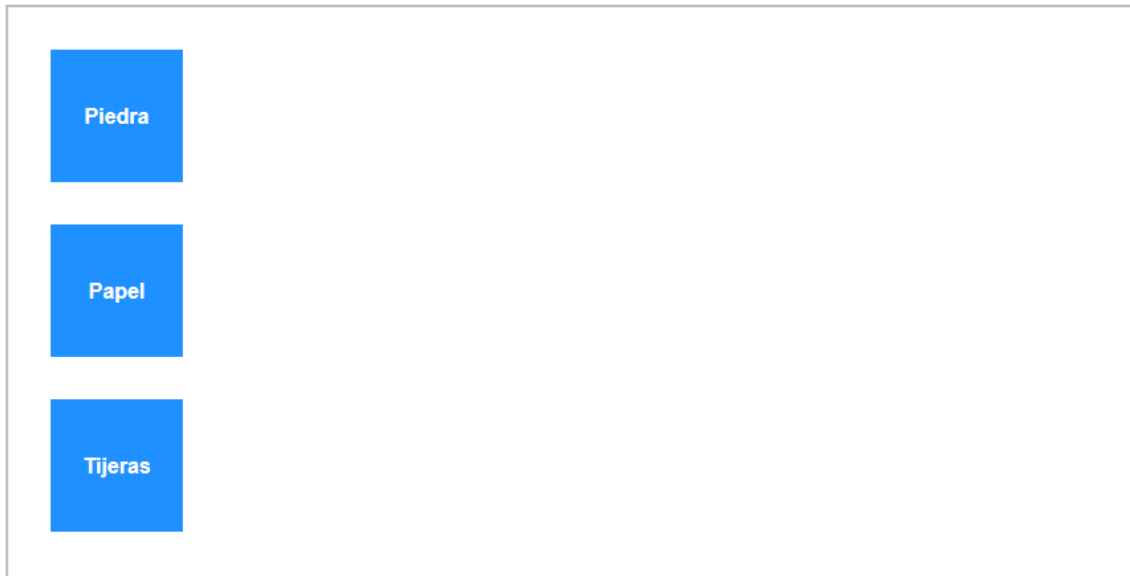
Para crear una flexbox que invierta el orden de las posiciones de la flexbox, se pueden realizar los siguientes pasos:

1. En una herramienta de edición o de desarrollo de aplicaciones, se crear un documento *HTML* que incluya la propiedad `flex-flow: column`:

```
<!doctype html>
<html>
<head>
  <meta charset="utf-8">
  <title>Ejemplo Flex-flow</title>
  <style>
    div {
      display: flex;
      display: -ms-flex;
      display: -moz-flex;
      display: -o-flex;
      display: -webkit-flex;
      flex-flow: column;
      -ms-flex-flow: column;
      -moz-flex-flow: column;
      -o-flex-flow: column;
      -webkit-flex-flow: column;
      height: 400px;
      padding: 1em;
      outline: 2px solid silver;
      color: white;
      font-family: sans-serif;
      font-weight: bold;
    }
    p {
      width: 100px;
      margin: 1em;
      height: 100px;
      background-color: dodgerblue;
      text-align: center;
      line-height: 100px;
    }
  </style>
</head>
<body>
  <div>
    <p>Piedra</p>
    <p>Papel</p>
    <p>Tijeras</p>
  </div>
</body>
</html>
```



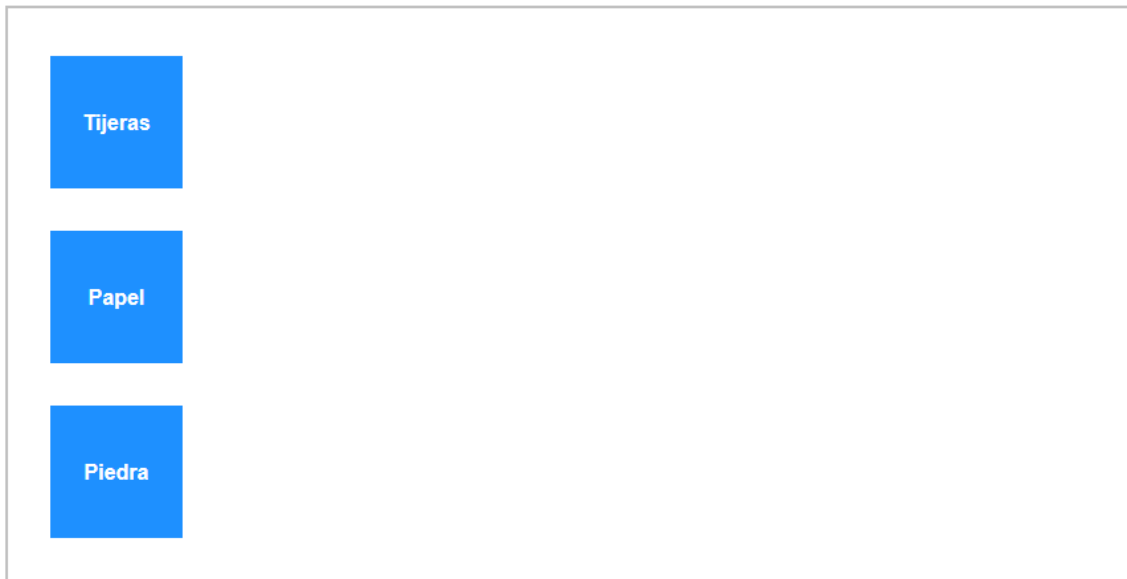
2. Se guarda el fichero como *L5-reverse-exercise.html* y se abre en el navegador Web. Ajustar el tamaño de la ventana del explorador web para que la pantalla se vea similar a la siguiente imagen:



3. Se abre el archivo en cada uno de los otros navegadores Web principales para ver si el archivo se muestra correctamente.
4. En el fichero HTML, se invierte el orden de las columnas utilizando el valor **flex-flow: column-reverse**, como se indica a continuación:

```
flex-flow: column-reverse;  
-ms-flex-flow: column-reverse;  
-moz-flex-flow: column-reverse;  
-o-flex-flow: column-reverse;  
-webkit-flex-flow: column-reverse;
```

5. Se guarda el archivo y se abre en el navegador web, la vista debería aparecer de la siguiente forma:



6. Se abre el archivo en cada uno de los otros navegadores Web principales para ver si el archivo se muestra de forma adecuada.
7. Se cierra el archivo, pero se deja la herramienta de edición y el explorador web abiertos si completa el siguiente ejercicio en esta sesión.

## 5. Tutorial Capítulo 6: Manejando Texto Flotante usando CSS

### 5.1. Entendiendo y Usando Regions para Contenido de Texto entre Múltiples Secciones

#### 5.1.1. Usando Columnas y Unión con Guión para Optimizar la Lectura del Texto

##### a. Creando Columnas

### CREAR UN LAYOUT MULTICOLUMNA:

Para crear una estructura de varias columnas, se debe realizar los pasos siguientes:

1. En una herramienta de edición o de desarrollo de aplicaciones, se crea un documento *HTML* adecuado que incluya el siguiente código, donde se aplica la propiedad `column-count` para especificar el número de columnas en las que se debe dividir el texto:

```
<!doctype html>
<html>
<head>
  <meta charset="utf-8" />
  <title>Tres columnas</title>
  <style>
    .tricolumn {
      -ms-column-count: 3;
      -moz-column-count: 3;
      -o-column-count: 3;
      -webkit-column-count: 3;
    }
  </style>
</head>
<body>
  <h2>Mis tres columnas</h2>
  <div class="tricolumn">
    En un lugar de la Mancha, de cuyo nombre no quiero acordarme, no ha mucho
    tiempo que viv&iacutea un hidalgo de los de lanza en astillero, adarga antigua,
    roc&iacuten flaco y galgo corredor. Una olla de algo m&aacutes vaca que carnero,
    salpic&oacuten las m&aacutes noches, duelos y quebrantos los s&aacuteacutebados,
    lentejas los viernes, alg&uacuten palomino de a&ntildeadidura los domingos,
    consum&iacutean las tres partes de su hacienda. El resto della conclu&iacutean sayo
    de velarte, calzas de velludo para las fiestas con sus pantuflos de lo mismo, los
    d&iacuteas de entre semana se honraba con su vellori de lo m&aacutes fino.
  </div>
</body>
</html>
```

2. Se guarda el archivo como **L6-columns-exercise.html**.
3. Se especifica un espaciado de columna de 2em y una regla para el ancho de columna que es 2px de ancho y verde. La sintaxis para las propiedades `column-gap` y `column-rule` son las siguientes:

```
.tricolumn {
  -ms-column-count: 3;
  -ms-column-gap: 2em;
```



```

-ms-column-rule: 2px solid green;
-moz-column-count: 3;
-moz-column-gap: 2em;
-moz-column-rule: 2px solid green;
-o-column-count: 3;
-o-column-gap: 2em;
-o-column-rule: 2px solid green;
-webkit-column-count: 3;
-webkit-column-gap: 2em;
-webkit-column-rule: 2px solid green;
}

```

## Mis tres columnas

En un lugar de la Mancha, de cuyo nombre no quiero acordarme, no ha mucho tiempo que vivía un hidalgo de los de lanza en astillero, adarga antigua, rocín flaco y galgo corredor. Una olla de

algo más vaca que carnero, salpicón las más noches, duelos y quebrantos los sábados, lentejas los viernes, algún palomino de añadidura los domingos, consumían las tres partes de su hacienda. El

resto della concluían sayo de velarte, calzas de velludo para las fiestas con sus pantuflos de lo mismo, los días de entre semana se honraba con su vellori de lo más fino.

4. Se guarda el fichero y se abre en el navegador, el resultado debería aparecer como en la siguiente imagen:
5. Se debe redimensionar la ventana del navegador para observar los efectos en las columnas.
6. (Opcional) Se cambia el título por otro más largo.
7. Se cierra el archivo, pero se deja abierta la herramienta de edición y el navegador web.

b. Usando Separación por Guiones (Hyphenation)

## PERMITIR LA SEPARACION POR GUIONES AUTOMÁTICA:

1. En una herramienta de edición o de desarrollo de aplicaciones, se debe crear un documento *HTML* que incluya el siguiente código, donde se usa la separación por guiones automática con la propiedad `hyphens: auto`, adicionalmente, se ha dejado la opción de texto justificado y con un tamaño de fuente de 14pt. Se debe reemplazar el texto del párrafo que comienza con "La separación por guiones es" con cualquier otro texto:

```

<!doctype html>
<html lang="en-us">
<head>
  <meta charset="utf-8" />
  <title>Separación por Guiones</title>
</head>
<body>
  <div style="width: 200px;
    border: 2px solid orange;">
    <p style="-moz-hyphens: auto;
      text-align: justify;
      font-size: 14pt;">
      La ciudad con el nombre más largo es
      Llanfairpwllgwyngyllgogerychwyrndrobwlllantysiliogogoch.
    </p>
  </div>
</body>

```



```
</html>
```

2. Se guarda el archivo como *L6-hyphen-exercise.html* y se debe observar el resultado en el navegador, en este caso, mozilla firefox.
3. Se cierra el archivo, pero se deja abierta la herramienta de edición y el navegador web.





## 6. Tutorial Capítulo 7: Entendiendo JavaScript y Fundamentos de Programación

### 6.1. Gestión y Mantenimiento JavaScript

#### CREAR UN PROGRAMA SIMPLE DE JAVASCRIPT:

En este tutorial se va a crear un programa simple en el que se utilicen las alertas. Éstas son muy útiles cuando se quieren realizar pruebas del funcionamiento del código realizado. En este caso, la alerta se mostrará en la pantalla del navegador cuando se pulse un botón. Para crear alertas se utiliza `alert()`.

Para crear un programa simple de *JavaScript*, se deben seguir los siguientes pasos:

1. Usando una herramienta de desarrollo, se crea un archivo con el siguiente contenido:

```
<!doctype html>
<html>
<head>
  <title>
    Mi programa de JavaScript
  </title>
</head>
<body>
  <h1>
    Mi primer programa de JavaScript
  </h1>
  <p>
    Esto es texto.
    <button type = 'button' onclick = "alert('Has hecho clic');">
      Hazme clic
    </button>
  </p>
</body>
</html>
```

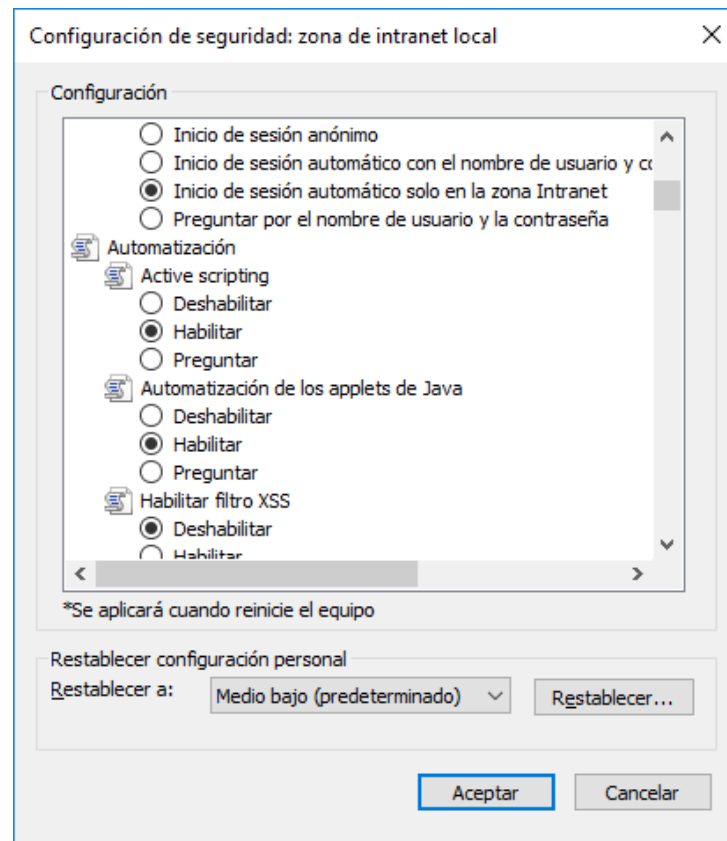
2. Se guarda el archivo como *L8-js1.html*.
3. Para ejecutar el programa, se abre *L8-js1.html* en un navegador. El resultado es parecido a la siguiente imagen:

## Mi primer programa de JavaScript

Esto es texto.



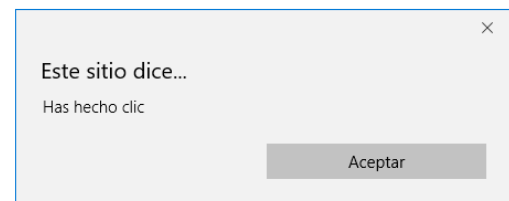
Si el programa de *JavaScript* no se muestra, se necesitaría activar *JavaScript* en la configuración del navegador. En *Internet Explorer 9*, por ejemplo, se hace seleccionando Herramientas -> Opciones de Internet. En la ventana de diálogo, se hace clic en *Seguridad*. Se hace clic en Nivel Personalizado. En la ventana de diálogo, se busca la sección *Scripting* (como se muestra en la imagen). Se hace clic en el *RadioButton* para activarlo. Y después se pincha en *Aceptar* también, para cerrar las ventanas de diálogo. Ahora se puede intentar abrir el archivo *L8-js1.html* otra vez en el navegador para ejecutar el programa *JavaScript*.



4. Se hace clic en el botón creado en *JavaScript*, que se muestra en la pantalla. Una ventana de alerta aparecerá, como se muestra en la imagen. Esto indica que el programa de *JavaScript* está funcionando correctamente.

## Mi primer programa de JavaScript

Esto es texto.



5. Se hace clic en *Aceptar* para cerrar la ventana de alerta.
6. Se cierra el archivo *HTML*, pero se deja la herramienta de edición y el explorador web abiertos si se piensa completar el siguiente ejercicio de esta sesión.

Este es el primer programa de *JavaScript*. No sólo tiene una apariencia concreta en la pantalla, sino que ésta cambia. Responder a las acciones del usuario es frecuentemente realizado en los programas.

Se puede comprobar en el código anterior cuál es la parte de *JavaScript*. Es el fragmento que hay a continuación:

```
alert('Has hecho clic');
```

El `alert()` es muy intrusivo y casi nunca se usa en la documentación de referencia. Pero es el camino más fácil para empezar con *JavaScript* y puede ser extremadamente útil durante el desarrollo o la depuración.

## CREAR UN PROGRAMA CON MÁS DE UNA SENTENCIA JAVASCRIPT:

En el anterior tutorial sólo se ha utilizado una sentencia de *JavaScript*, pero eso en la práctica casi nunca será así; normalmente se querrá realizar más de una acción. Para mostrar esto, el presente tutorial generará dos alertas, una después de otra. Para ello, al igual que antes, se utilizará el comando `alert()`

Para crear un programa con más de una sentencia *JavaScript*, se pueden seguir los siguientes pasos:

1. En una aplicación de desarrollo, se actualiza *L8-js1.html*, reemplazando el `alert()` por lo siguiente:

```
alert('Primera alerta');  
alert('Segunda alerta');
```

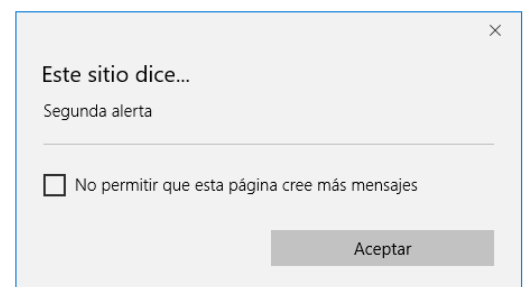
2. Se guarda el archivo.
3. Se ejecuta el programa de *JavaScript* abriendo el archivo *HTML* en el navegador web.



4. Cuando la primera ventana de alerta se muestre, se cierra haciendo clic en *Aceptar*. El programa *JavaScript* procede a mostrar la segunda ventana de alerta, como se muestra en la imagen:

## Mi primer programa de JavaScript

Esto es texto.



5. Se cierra la segunda alerta haciendo clic en *Aceptar*.
6. Se cierra el archivo *HTML*, pero se deja la herramienta de edición y el explorador web abiertos si se piensa completar el siguiente ejercicio de esta sesión.

### 6.1.1. Creación y Uso de Funciones

## CREACIÓN Y USO DE FUNCIONES EN JAVASCRIPT:

En el siguiente tutorial se va a realizar un programa similar a los mostrados anteriormente, pero utilizando funciones. Las funciones son muy útiles si se pretende realizar su contenido de forma repetida, además facilitan la depuración. Esto se hace así: `function nombreFuncion(){...}`

Para aprender a usar una función de *JavaScript*, se pueden seguir los siguientes pasos:

1. En una herramienta de desarrollo, se crea un archivo *L8-js2.html* con el siguiente contenido:

```
<!doctype html>
<html>
<head>
  <title>
    Mi programa de JavaScript
  </title>
  <script type = "text/JavaScript">
    function ejemplo1() {
      alert('Primera alerta');
      alert('Segunda alerta');
    }
  </script>
</head>
```



```
<body>
  <h1>
    Mi primer programa de JavaScript
  </h1>
  <p>
    Esto es texto.
    <button type = 'button' onclick = "ejemplo1();">
      Hazme clic
    </button>
  </p>
</body>
</html>
```

2. Se abre *L8-js2.html* en el navegador. El programa se muestra, como se puede ver, en la siguiente imagen. Hay texto y un botón. De momento, no hay evidencia de *JavaScript*.

## Mi primer programa de JavaScript

Esto es texto.

3. Se hace clic en el botón. Se puede comparar la acción de esta página con la del archivo *L8-js1.html* que contenía dos alertas. Se comprueba que la ventana de alerta se ve de la misma forma cuando se llama desde una función y cuando no.
4. Se cierra el archivo *HTML*, pero se deja la herramienta de edición y el explorador web abiertos si se piensa completar el siguiente ejercicio de esta sesión.

Este ejemplo introduce al menos otros dos conceptos nuevos, además del uso de una función. En primer lugar, muestra *JavaScript* incrustado en etiquetas `<script>` dentro del *HTML*. Hay varias maneras de "conectar" un programa *JavaScript* al *HTML*. Se suele utilizar la etiqueta `<script>` dentro de la etiqueta `<head>`, especialmente para proyectos *JavaScript* de tamaño mediano.

Además, el nombre "*ejemplo1*" de la función merece atención. Este nombre está bajo nuestro control.

Cuando se escribe `href` como parte de un hipervínculo, o `alert()` para que aparezca una alerta, se debe basarse en palabras clave definidas en los estándares para *HTML* y *JavaScript*, respectivamente. El nombre de la función "*ejemplo1*", sin embargo, no está en tales estándares, es un nombre a nuestra elección. Simplemente se debe ser coherentes; si escribimos "*mi\_ejemplo*" en lugar de "*example1*" en la definición de la función, entonces también se debe usar "*mi\_ejemplo*".

### USO DE VARIABLES EN JAVASCRIPT:

Para almacenar la información se utilizan las variables. En este tutorial se puede ver fácilmente cuál es la asignación de variable; `var nombre_version` almacena una cadena de texto que se puede utilizar donde sea necesario. En este caso se utilizará para agregar contenido a las alertas.

Para usar una variable en un programa *JavaScript*, se pueden seguir los siguientes pasos:

1. En una herramienta de desarrollo, se crea *L8-js3.html* con el siguiente contenido:

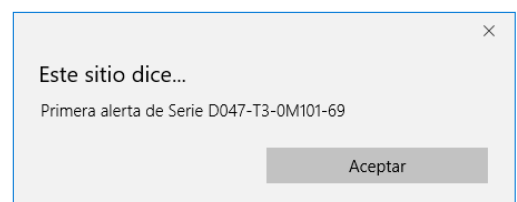


```
<!doctype html>
<html>
<head>
  <title>
    Mi programa de JavaScript
  </title>
  <script type = "text/JavaScript">
    function ejemplo1() {
      var nombre_version = "Serie D047-T3-0M101-69"
      alert("Primera alerta de " + nombre_version);
      alert("Segunda alerta de " + nombre_version);
    }
  </script>
</head>
<body>
  <h1>
    Primer uso de variables
  </h1>
  <p>
    Esto es texto.
    <button type = 'button' onclick = "ejemplo1();">
      Hazme clic
    </button>
  </p>
</body>
</html>
```

2. Se abre *L8-js3.html* en el navegador y se hace clic en el botón. La primera ventana de alerta se muestra en la pantalla.

## Primer uso de variables

Esto es texto.



3. Se hace clic en *Aceptar* para cerrar la primera alerta y pasar a la segunda. Se puede comprobar la utilidad de esto. Un número de serie u otra cantidad importante pueden aparecer en varios lugares diferentes en un programa. No es necesario copiar el valor en cada ubicación; en su lugar, *JavaScript* permite utilizar el nombre de la variable que contiene el valor.
4. Se hace clic en *Aceptar* para cerrar el segundo cuadro de alerta.



5. Se cierra el archivo *HTML*, pero se deja la herramienta de edición y el explorador web abiertos si se piensa completar el siguiente ejercicio de esta sesión.

### 6.1.2. Usando jQuery y Otras Librerías

#### UTILIZACIÓN DE JQUERY:

*jQuery* es de gran ayuda, permite agilizar el trabajo cuando se utiliza JavaScript. Utilizando `$("p").click(...)` se pueden realizar acciones ante el evento *click*, en este caso un párrafo, aunque se podría utilizar los mismos identificadores que se utilizan en CSS para acceder a ellos (Por ejemplo `.class` o `#id`).

Se puede comprobar que también se realiza ocultación de contenido; esto se hace utilizando `$(this).hide();`, aunque también existe su opuesto para mostrar contenido: `$(this).show();`.

En el siguiente tutorial se utilizará *jQuery* para hacer que desaparezcan unos párrafos al hacer clic sobre ellos. Para ello seguir los siguientes pasos:

1. En una herramienta de desarrollo se crea un archivo llamado *L8-js4.html* con el siguiente contenido:

```
<!doctype html>
<html>
<head>
  <title>
    Primer uso de jQuery
  </title>
  <script type = "text/JavaScript" src = "http://ajax.aspnetcdn.com/ajax/jquery/jquery-
1.5.js">
  </script>
  <script type = "text/JavaScript">
    // Una vez que el HTML se carga, se ejecuta
    // la función ready().
    $(document).ready(function() {
      // Cada párrafo tiene una acción clic
      // que oculta la frase en particular
      $("p").click(function() {
        $(this).hide();
      });
    });
  </script>
</head>
<body>
  <p>Primera frase, hazme clic para hacerme desaparecer.
  <p>Esto es la segunda frase.
  <p>Esto es la tercera frase.
</body>
</html>
```

2. Se abre el archivo en el navegador. Éste mostrará una ventana como la siguiente. Se comprueba que se muestran tres frases.



Primera frase, hazme clic para hacerme desaparecer.

Esto es la segunda frase.

Esto es la tercera frase.

3. Se hace clic en algún lugar de la primera frase y ésta desaparecerá.

Esto es la segunda frase.

Esto es la tercera frase.

4. Se hace clic en algún lugar de la segunda frase para hacerla desaparecer.
5. Se cierra el archivo *HTML*, pero se deja la herramienta de edición y el explorador web abiertos si se piensa completar el siguiente ejercicio de esta sesión.

Se puede ver la utilidad que esto puede tener en una página Web, aplicaciones móviles, u otros. Es posible que se desee que la información se muestre a sus usuarios finales de forma distinta dependiendo de las circunstancias. Aunque es posible escribir *JavaScript* "puro", sin **jQuery**, se requiere una cantidad de código considerablemente mayor. **jQuery** hace que muchas operaciones comunes sean más cortas, fáciles de entender y fáciles de expresar.

## 6.2. Actualizando la UI Usando JavaScript

### ACTUALIZANDO LA INTERFAZ DE USUARIO CON JAVASCRIPT:

Para mostrar un ejemplo de lo que *JavaScript* hace posible, se puede construir una pequeña calculadora dentro del navegador, para ello se pueden seguir los siguientes pasos:

1. En una herramienta de edición, se crea el archivo *L8-js5.html* con el siguiente contenido:

```
<!doctype html>
<html>
<head>
  <title>
    Calculadora de navegador
  </title>
</head>
<body>
  <h1>
    Calculadora de navegador
  </h1>
  <form name="calculadora">
    <table border=4>
      <tr>
        <td>
```





```

        <input type="text" name="Entrada" Size="20">
        <br>
    </td>
</tr>
<tr>
    <td>
        <input type="button" name="uno" value=" 1 " OnClick="calculadora.
Entrada.value += '1'">
        <input type="button" name="dos" value=" 2 " OnClick="calculadora.
Entrada.value += '2'">
        <input type="button" name="tres" value=" 3 " OnClick="calculadora.
Entrada.value += '3'">
        <input type="button" name="suma" value=" + " OnClick="calculadora.
Entrada.value += ' + '">
        <br>
        <input type="button" name="cuatro" value=" 4 " OnClick="calculadora.
Entrada.value += '4'">
        <input type="button" name="cinco" value=" 5 " OnClick="calculadora.
Entrada.value += '5'">
        <input type="button" name="seis" value=" 6 " OnClick="calculadora.
Entrada.value += '6'">
        <input type="button" name="menos" value=" - " OnClick="calculadora.
Entrada.value += ' - '">
        <br>
        <input type="button" name="siete" value=" 7 " OnClick="calculadora.
Entrada.value += '7'">
        <input type="button" name="ocho" value=" 8 " OnClick="calculadora.
Entrada.value += '8'">
        <input type="button" name="nueve" value=" 9 " OnClick="calculadora.
Entrada.value += '9'">
        <input type="button" name="multiplicar" value=" x " OnClick=" calculadora.
Entrada.value += ' * '">
        <br>
        <input type="button" name="clear" value=" c " OnClick=" calculadora.
Entrada.value = ''">
        <input type="button" name="cero" value=" 0 " OnClick="calculadora.
Entrada.value += '0'">
        <input type="button" name="Hacer" value=" = " OnClick="calculadora.
Entrada.value = eval(calculadora.Entrada.value)">
        <input type="button" name="dividir" value=" / " OnClick="calculadora.
Entrada.value += ' / '">
    </td>
</tr>
</table>
</form>
</body>
</html>

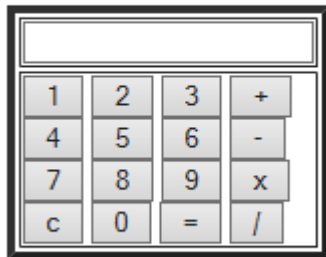
```

Se puede comprobar el código consta de un formulario con *name* "calculadora", y que la pantalla de la calculadora tiene el *name* "Entrada". En cada tecla hay un evento **onClick** que le asigna un valor a la pantalla mediante el siguiente código **calculadora.Entrada.value**. Se puede comprobar que, además, en la tecla '=' aparece la función **eval()**, que permite realizar el cálculo deseado.



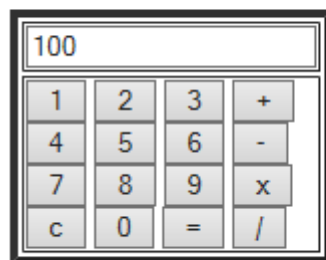
2. Se abre el archivo *L8-js5.html* en el navegador. La calculadora se muestra como en la siguiente imagen:

## Calculadora de navegador



3. Se intenta hacer una cuenta haciendo click en los botones  $7 \times 14 + 2 =$  y se observa el resultado. Se comprueba que funciona correctamente y muestra un 100 como resultado.

## Calculadora de navegador



4. Se cierra el archivo *HTML*, pero se deja la herramienta de edición y el explorador web abiertos si se piensa completar el siguiente ejercicio de esta sesión.

Este pequeño ejemplo ilustra que un programa de *JavaScript* puede hacer dentro del navegador todo lo que hace cualquier otra aplicación, y a veces en sólo unas pocas líneas de código. Las capacidades de *JavaScript* incluyen la entrada de datos, la respuesta a las pulsaciones del teclado y el ratón, movimientos, visualización de resultados, cálculos complejos y más, como los siguientes ejercicios ilustran. Las aplicaciones móviles construidas sobre la base de *HTML5*, por supuesto, tienen las mismas capacidades.

### 6.2.1. Localizando y Accediendo a Elementos

#### ACCESO A ELEMENTOS EN JAVASCRIPT:

Algo básico en JavaScript es el acceso a los elementos, en este tutorial se puede ver cómo acceder a elementos desde su id. El siguiente ejercicio consta de una entrada al usuario y un botón, la función del programa es comprobar si lo que se ha introducido es una cifra.

Para aprender a utilizar `getElementById()` se pueden seguir los siguientes pasos:

1. En una herramienta de desarrollo, se crea un archivo llamado *L8-js6.html* con el siguiente contenido:

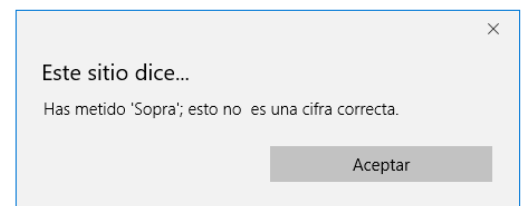
```
<html>
<head>
  <title>
    Validar datos de usuario
  </title>
  <script type = "text/JavaScript">
    function validar() {
      var valor = document.getElementById("entrada1").value;
      if (isNaN(valor)) {
        modificador = "no ";
      } else {
        modificador = "";
      }
      var reporte = "Has metido '" + valor + "'; esto "+ modificador + " es una cifra
correcta.";
      alert(reporte);
    }
  </script>
</head>
<body>
  <h1>
    Validar datos de usuario
  </h1>
  <form name="calculadora">
    <input type = "text" id = "entrada1"></input>
    <button type = "button" onclick = "validar();">Hazme clic para ver lo que opino
sobre tu entrada.
  </button>
  </form>
</body>
</html>
```

Se puede comprobar que la entrada de datos del usuario tiene el id *entrada1*, además, se puede comprobar que existe un botón con la función **onClick**. Este evento invocará la función **validar()**, que mediante el uso de **getElementById**, almacena en la variable **valor** el contenido del **input**. Se puede ver también la existencia de una estructura condicional donde se utiliza la función **isNaN()**, ésta devuelve un **true** en el caso de que el valor no sea un número, o **false** en caso contrario.

2. Se abre el archivo en el navegador.
3. Se escribe una palabra corta en el área de entrada y luego se hace clic en el botón. Los resultados se muestran en la imagen. Se hace clic en Aceptar para agregar los cambios y se cierra el cuadro de diálogo.



## Validar datos de usuario

 Hazme clic para ver lo que opino sobre tu entrada.

4. Se escribe un número en el área de entrada y se hace clic en el botón. Se hace clic en *Aceptar* para agregar los cambios y se cierra el cuadro de diálogo.
5. Se cierra el archivo *HTML*, pero se deja la herramienta de edición y el explorador web abiertos si se piensa completar el siguiente ejercicio de esta sesión.

Este pequeño ejemplo ilustra que un programa de *JavaScript* puede hacer dentro del navegador todo lo que hace cualquier otra aplicación, y a veces en sólo unas pocas líneas de código.

### 6.2.2. Escuchando y Respondiendo a Eventos

## TRABAJANDO CON EVENTOS EN JAVASCRIPT:

La parte interesante del siguiente tutorial es la utilización del evento **onload**. Se puede comprobar que el evento está en el **body** y llama a la función **init()**, ésta simplemente mostrará una alerta cuando el **body** se haya cargado completamente.

Para usar el evento **onload**, se pueden seguir los siguientes pasos:

1. En una herramienta de desarrollo, se crea un archivo *L8-js7.html* con el siguiente código:

```
<!doctype html>
<html>
<head>
  <title>
    El evento onload
  </title>
  <script type = "text/JavaScript">
    function validar() {
      var valor = document.getElementById("entrada1").value;
      if (isNaN(valor)) {
        modificador = "no ";
      } else {
```



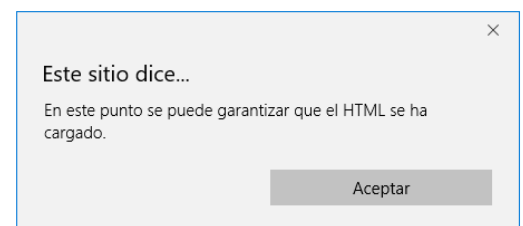
```

        modificador = "";
    }
    var reporte = "Has metido '" + valor + "'; esto "+ modificador + " es una cifra
correcta.";
    alert(reporte);
}
function init() {
    alert("En este punto se puede garantizar que el HTML se ha cargado.");
}
</script>
</head>
<body onload = "init();">
    <h1>
        El evento onload
    </h1>
    <form name="calculadora">
        <input type = "text" id = "entrada1"></input>
        <button type = "button" onclick = "validar();">
            Hazme clic para ver lo que opino de tu entrada.
        </button>
    </form>
</body>
</html>

```

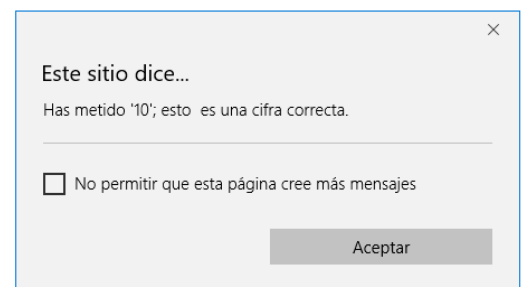
2. Se abre *L8-js7* en un navegador. Se muestra un formulario como se muestra en la imagen. La ventana de alerta es visible cuando el `<body>` ha terminado de cargar; ese es el significado de `onload="init();"`.

## El evento onload

 Hazme clic para ver lo que opino de tu entrada.


3. Se hace clic en *Aceptar* para cerrar la ventana de diálogo.
4. Se introduce un valor en el *input*, se puede ver como ejemplo la siguiente imagen:

## El evento onload

5. Se puede experimentar con otros valores.
6. Se cierra el archivo *HTML*, pero se deja la herramienta de edición y el explorador web abiertos si se piensa completar el siguiente ejercicio de esta sesión.

Un síntoma común de los programas *JavaScript* defectuosos es que son erráticos, dan diferentes resultados en diferentes momentos. En algunos casos, esto se debe a que el programa está escrito en un archivo de manera que depende de la existencia de un elemento de pantalla en particular, pero no asegura su existencia. Iniciar el programa en diferentes momentos puede tener resultados distintos. Una táctica para tales problemas es empezar cálculos sólo después de que **onload** haya "disparado", como en *L8-js7.html*.

### 6.2.3. Mostrando Elementos Ocultos

#### OCULTANDO Y MOSTRANDO ELEMENTOS CON JAVASCRIPT:

El siguiente ejercicio contiene una entrada de datos que introducirá el usuario y, en caso de que éste introduzca una cantidad que esté dentro de los límites marcados aparecerá un mensaje de advertencia. Si se introduce un valor mayor o igual de 80 aparecerá dicha alerta. Se puede comprobar que la entrada de datos del usuario tiene id "precio1", el párrafo de advertencia en este caso no tiene id pero se accede a él utilizando la siguiente función de JavaScript `getElementsByName("p");`. También es interesante prestar atención a la sentencia `primer_parrafo.style.display = mostrar;`, mediante la cual accede a los estilos del párrafo y se decide, según el contenido de la variable "mostrar", si se mostrará o no. Si se le asigna "**block**" se mostrará el mensaje, si se le asigna "**none**", no se mostrará.

Para crear una aplicación que muestre y oculte párrafos basados en los valores que introduce un usuario se pueden seguir los siguientes pasos:

1. En el programa de desarrollo se crea un archivo *L8-js8.html* con el siguiente contenido:

```
<!doctype html>
<html>
```



```

<head>
  <title>
    Mostrar/ocultar elementos
  </title>
  <script type = "text/JavaScript">
    function comprobar_rango(){
      var valor = document.getElementById("precio1").value;
      var lista_parrafos = document.getElementsByTagName("p");
      var primer_parrafo = lista_parrafos[0];
      if (valor >= 80) {
        mostrar = "block";
      } else {
        mostrar = "none";
      }
      primer_parrafo.style.display = mostrar;
    }
  </script>
</head>
<body>
  <h1>
    Mostrar/Ocultar elemento
  </h1>
  <form>
    Introduce un precio:
    <input type = "number" id = "precio1" min = "1" max = "100" oninput = "
comprobar_rango();">
    </input>
  </form>
  <p style = "display:none;">Advertencia: Cifra dentro del 20% de las limitaciones.</p>
</body>
</html>

```

2. Se abre el archivo *L8-js8.html* en el navegador. El resultado es el que se muestra en la siguiente imagen.

## Mostrar/Ocultar elemento

Introduce un precio:

3. Con el teclado, se introduce cada uno de estos "precios" pulsando la tecla *Intro* después de cada uno de ellos. Precios: 1, 50, 79, 80, 90 y 60. Aparece un mensaje de advertencia después de introducir 80 y 90, como se muestra a continuación:



## Mostrar/Ocultar elemento

Introduce un precio:

Advertencia: Cifra dentro del 20% de las limitaciones.

4. Se cierra el archivo *HTML*, pero se deja la herramienta de edición y el explorador web abiertos si piensa completar el siguiente ejercicio de esta sesión.

Se debe recordar que cuando se prueba la funcionalidad de mostrar/ocultar, *JavaScript* hace que el mensaje se oculte de nuevo si se está por debajo de los límites marcados, a través de asignación del atributo *display*; esto se hace una vez superada la estructura condicional que decide si se mostrará o no. Si no estuviese la estructura condicional, una vez que se muestra, no desaparecería nunca más.

### 6.2.4. Actualizando el Contenido de Elementos

El siguiente tutorial es una extensión del anterior, aunque en este caso se actualizará la pantalla mostrando el precio total tras sumarle los impuestos del 21%.

Algo básico en *JavaScript* es la actualización del contenido de los elementos, esto se puede realizar utilizando *innerHTML* contra un elemento obtenido anteriormente mediante *getElementById*. Se puede comprobar que se utiliza *oninput*, en la línea de *input*, esto llamará a las funciones creadas cada vez que el usuario introduzca un valor.

Para crear una aplicación que actualiza el contenido visible en la pantalla, se pueden seguir los siguientes pasos:

1. Se crea un archivo llamado *L8-js9.html* con el siguiente contenido:

```
<!doctype html>
<html>
<head>
  <title>
    Actualizar elementos
  </title>
  <script type = "text/JavaScript">
    // comprueba_rango asigna el estilo del display
    // al primer párrafo como una función
    // que muestra si el precio es 80 o
    // mayor, o 79 y menor.
    function comprobar_rango () {
      var valor = document.getElementById("precio1").value;
      var lista_parrafos = document.getElementsByTagName("p");
      var primer_parrafo = lista_parrafos[0];
      if (valor >= 80)
      {
        mostrar = "block";
      }
      else
```





```

        {
            mostrar = "none";
        }
        primer_parrafo.style.display = mostrar;
    }
    // computo_total() tiene la responsabilidad
    // de actualizar el display con el total
    // del precio con el impuesto
    function computo_total()
    {
        var valor = document.getElementById("precio1").value;
        if (isNaN(valor))
        {
            total = "INDETERMINADO";
        }
        else
        {
            // Se asume que los impuestos son un 21%.
            total = 1.21 * valor;
            total = total.toFixed(2);
        }
        var cantidad_total = document.getElementById("total");
        cantidad_total.innerHTML = total;
    }
</script>
</head>
<body>
    <h1>
        Actualizar elementos
    </h1>
    <form>
        Introduce el precio
        <input type = "number" id = "precio1" min = "1" max = "100" oninput =
"comprobar_rango(); computo_total();">
        </input>
        El precio total, incluyendo el impuesto, es
        <span id = "total">INDETERMINADO</span>.
    </form>
    <p style = "display:none;">Advertencia: Cifra dentro del 20% de las limitaciones
    </p>
</body>
</html>

```

2. Se abre *L8-js9.html* en un navegador Web.
3. Utilizando el teclado, se escribe el número 1 en el cuadro de texto. Aparece un mensaje como el que se muestra en la siguiente imagen, mostrando el precio más el 21 por ciento de impuestos en un artículo de \$99. Esto es rápido y no tiene problemas de actualizar la pantalla.

## Actualizar elementos

Introduce el precio  El precio total, incluyendo el impuesto, es 119.79.

Advertencia: Cifra dentro del 20% de las limitaciones

4. Se pulsa *Intro*.



5. Se escribe 50 en el cuadro de texto y se observa cómo cambia el mensaje mostrado.
6. Se repiten los pasos 3 y 4 utilizando los valores 79, 80, 90 y 60 cada vez.
7. Se cierra el archivo *HTML*, pero deja la herramienta de edición y el explorador web abiertos si se piensa completar el siguiente ejercicio de esta sesión.

### 6.2.5. Añadiendo Elementos

## AÑADIENDO ELEMENTOS CON JAVASCRIPT:

En el siguiente tutorial se puede comprobar cómo añadir párrafos de texto al contenido de la página. Se utilizará `createElement("p");` para crear un nuevo elemento `<p>` (párrafo). Utilizando `Date();` se obtiene la fecha y hora actual. Utilizando `createTextNode()` se introduce el texto que previamente se ha decidido, que será añadido mediante el uso de `appendChild` e insertado sobre el párrafo que ya existía mediante `insertBefore`.

Para crear una aplicación que añada elementos en la pantalla se pueden seguir los siguientes pasos:

1. En una aplicación de desarrollo de código, se crea el archivo *L8-js10.html* con el siguiente contenido:

```
<!doctype html>
<html>
<head>
  <title>
    Crea un nuevo elemento
  </title>
  <script type = "text/JavaScript">
    function add_parrafo() {
      var original = document.getElementById("original");
      var nuevo_parrafo = document.createElement("p");
      var tiempo_actual = new Date();
      var este_texto = "Esta frase aparece en" +
        tiempo_actual + ".";
      // Incluso después de que el navegador haya
      // cargado todo el HTML, es posible añadir
      // nuevos elementos HTML. Con createTextNode()-
      // appendChild()-insertBefore() son unas
      // de las formas típicas de añadirlo
      var nuevo_contenido = document.createTextNode(este_texto);
      nuevo_parrafo.appendChild(nuevo_contenido);
      document.body.insertBefore(nuevo_parrafo,original);
    }
  </script>
</head>
<body>
  <h1>
    Crear un nuevo elemento
  </h1>
  <p id = "original">Esto es texto que aparece cuando el primer display se carga.
  </p>
  <button type = "button" onclick = "add_parrafo();">
    Clic para agregar contenido
  </button>
</body>
</html>
```



2. Se abre *L8-js10.html* en un navegador web. El programa muestra lo siguiente:

## Crear un nuevo elemento

Esto es texto que aparece cuando el primer display se carga.

Clic para agregar contenido

3. Se hace clic en el botón. La ventana cambia como se muestra en la figura:

## Crear un nuevo elemento

Esta frase aparece en Tue Dec 04 2018 10:45:28 GMT+0100 (Hora estándar romance).

Esto es texto que aparece cuando el primer display se carga.

Clic para agregar contenido

4. Se hace clic en el botón unas cuantas veces más para ver los resultados.
5. Se cierra el archivo *HTML*, pero se deja la herramienta de edición y el explorador web abiertos si se piensa completar el siguiente ejercicio de esta sesión.



## 7. Tutorial Capítulo 8: Trabajando con Gráficos y Accediendo a Datos

### 7.1. Trabajando con Imágenes, Formas y otros Gráficos

#### REALIZACIÓN DE UN RELOJ ANALÓGICO CON CANVAS EN JAVASCRIPT:

En el siguiente tutorial se va a realizar un reloj analógico mediante el uso de canvas y JavaScript. Se puede comprobar que se utiliza `getSeconds()`, `getMinutes()` y `getHours()`, para obtener los segundos, minutos y hora actual. Mediante `fillStyle` se fija el color, con `moveTo` se fija el centro y con `lineTo` se dibujan las manecillas del reloj. Mediante el uso de `setTimeout` se fija que se refresque cada 1000 ms, es decir, cada segundo.

Para utilizar canvas de *HTML5*, se pueden seguir los siguientes pasos:

1. Se crea un archivo *L9-js3.html* con el siguiente contenido:

```
<!doctype html>
<html>
  <head>
    <title>
      Reloj hecho en JavaScript con canvas
    </title>
    <script type = "text/javascript">
      function dibujar_manecilla(fraccion) {
        dctx.lineTo(centro_x + longitud * Math.sin(2 * Math.PI * fraccion), centro_y
- longitud * Math.cos(2 * Math.PI * fraccion));
      }
      function init() {
        var canvas = document.getElementById("esferareloj");
        // Las siguientes variables son creadas como
        // globales, deben ser convenientemente
        // accedidas por otras funciones.
        dctx = canvas.getContext('2d');
        dctx.fillStyle = "black";
        centro_x = 100;
        centro_y = 100;
        longitud = 100;
        mostrar_manecillas();
      }
      // Cada manecilla se dibuja como un triángulo isósceles
      // desde el centro hasta el borde de la esfera del reloj.
      function mostrar_manecilla(fraccion, ancho)
      {
        dctx.beginPath();
        dctx.moveTo(centro_x, centro_y);
        dibujar_manecilla(fraccion - ancho);
        dibujar_manecilla(fraccion + ancho);
        dctx.fill();
      }
      function mostrar_manecillas()
      {
        // Elimina cualquier cosa presente en el area
        // que representa la esfera del reloj.
        dctx.clearRect(0, 0, 200, 200);
        // ¿Qué hora es?
```



```

        var ahora = new Date();
        segundos = ahora.getSeconds();
        minutos = ahora.getMinutes() + segundos / 60;
        horas = ahora.getHours() + minutos / 60;
        // La manecilla de segundos es la más fina de todas.
        mostrar_manecilla(segundos / 60, 0.002);
        mostrar_manecilla(minutos / 60, 0.005);
        // La manecilla de la hora es tan extensa como
        // la de los minutos.
        mostrar_manecilla(horas / 12, 0.01);
        var rate = 1000;
        setTimeout(mostrar_manecillas, rate);
    }
</script>
</head>
<body onload = "init();">
    <h1>
        Reloj hecho en JavaScript con canvas
    </h1>
    <canvas id = "esferareloj" width = "200" height = "200">
    </canvas>
</body>
</html>

```

2. Se abre el archivo creado en un navegador. El navegado muestra lo que se puede ver en la siguiente imagen:

## Reloj hecho en Javascript con canvas



3. Se cierra el archivo *HTML*, pero se deja la herramienta de edición y el explorador web abiertos si se piensa completar el siguiente ejercicio de esta sesión.

### Creando Formas Animadas con Canvas en JavaScript

El siguiente ejercicio crea un dibujo abstracto colocando bloques en un canvas *HTML5*. El ejercicio ilustra que se necesitan relativamente pocas líneas de *JavaScript* para producir efectos complejos. Las funciones que se han utilizado son similares a las del tutorial anterior.

Para crear formas animadas usando canvas, se pueden seguir los siguientes pasos:

1. Crea un archivo *HTML* llamado *L9-js4.html* con el siguiente contenido:

```
<!doctype html>
<html>
<head>
  <title>
    Bloque
  </title>
  <script type = "text/javascript">
    // Esta página pone el color
    // en la pantalla con una cierta aleatoriedad
    // para lograr efectos interesantes
    // La recursividad se utiliza de dos formas distintas
    // por debajo: colocar_bloques() llama a dibujar_espiral(), y
    // dibujar_espiral() llama también a colocar_bloques() o
    // dibujar_espiral(), dependiendo de cómo
    // se ha dibujado recientemente.
    function init()
    {
      var canvas = document.getElementById("dibujar_area");
      dctx = canvas.getContext('2d');
      colocar_bloques();
    }
    function dibujar_espiral()
    {
      // Una vez que el bloque se mueve fuera
      // del area, para la espiral actual, y crea
      // una nueva.
      if (x > 500 || y > 500 || x < 0 || y < 0)
      {
        colocar_bloques();
      }
      ratio = 1.6;
      nuevax = x;
      nuevay = y;
      dx = size;
      dy = size;
      // Cada bloque se gira 90º
      // desde el anterior.
      switch (direccion)
      {
        case "arriba":
          dy = -size;
          nuevay += dy;
          direccion = "izquierda";

          break;
        case "izquierda":
          dx = -size;
          dy = -size;
          nuevax += dx;
          direccion = "abajo";
          break;
        case "abajo":
          dx = -size;
          nuevay += dy;
          direccion = "derecha";
          break;
        case "derecha":
          nuevax += dx;
          direccion = "arriba";
          break;
      }
    }
  }
</script>
</head>
<body>
  <div id = "dibujar_area">
  </div>
</body>
</html>
```



```

        dctx.fillRect(x, y, dx, dy);
        // Cada bloque sucesivo es más grande
        // que el anterior.
        size *= ratio;
        x = nuevax;
        y = nuevay;
        setTimeout(dibujar_espiral, delay);
    }
    function colocar_bloques()
    {
        dctx.fillStyle = '#' + Math.floor(Math.random()*16777215).toString(16);
        x = 100 + 300 * Math.random();
        y = 100 + 300 * Math.random();
        delay = 100 + 2000 * Math.random();
        size = 3 + 7 * Math.random();
        direccion = "arriba";
        dibujar_espiral();
    }
</script>
</head>
<body onload = "init();">
    <h1>Bloques</h1>
    <canvas id = "dibujar_area" width = "500" height = "500"></canvas>
</body>
</html>

```

2. Se abre el navegador web. La pantalla muestra bloques de color como se puede ver en la siguiente imagen.

## Bloques



3. Se cierra el archivo *HTML*, pero se deja la herramienta de edición y el explorador web abiertos si se piensa completar el siguiente ejercicio de esta sesión.

## 7.2. Enviando y Recibiendo Datos

### ENVÍO Y RECEPCIÓN DE DATOS CON JAVASCRIPT:

En el siguiente tutorial se puede comprobar cómo acceder al título de la página web, esto se hace fácilmente mediante el uso de `document.title`.

Para acceder a datos usando *JavaScript* se puede seguir el siguiente ejemplo:

1. Se crea el archivo *HTML L9-js5.html* con el siguiente contenido:

```
<!doctype html>
<html>
<head>
  <title>
    JavaScript accediendo a datos
  </title>
  <script type = "text/javascript">
    function init()
    {
      var objeto_parrafo =
        document.getElementById('parrafo');
      mensaje = "El titulo de esta web es '" + document.title + "'.";
      objeto_parrafo.innerHTML = mensaje;
    }
  </script>
</head>
<body onload = "init();">
  <h1>
    JavaScript accede a datos
  </h1>
  <p id = "parrafo"></p>
</body>
</html>
```

2. Se abre el archivo *L9-js5.html* en el navegador web. La pantalla se mostrará como en la siguiente imagen:

## JavaScript accede a datos

El titulo de esta web es 'JavaScript accediendo a datos'.

3. Se comprueba que *JavaScript* puede acceder al contenido *HTML*.





4. Se cierra el archivo *HTML*, pero se deja la herramienta de edición y el explorador web abiertos si se piensa completar el siguiente ejercicio de esta sesión.

El sentido de este tipo de ejemplo *JavaScript* está destinado a mostrar la posibilidad de la existencia de una librería de todo el contenido. Podría haber un estilo, o política, a aplicar en todas las páginas. En ese sentido, esta forma de actuar es mucho más conveniente y no es propensa a problemas; de esta forma se puede encapsular la política en una librería *JavaScript* colocada en el `title`, donde se sabe cómo recuperar la información, en lugar de tener que copiar y pegar el título de todas las páginas para todo el sitio. Cada página incluye una copia idéntica de la librería *JavaScript*.

"Política" es una palabra que los programadores usan para referirse, en grandes rasgos, a "estilo de acción". Un estilo visual podría ser algo que los humanos expresamos como "cada página debe aparecer con imagen de fondo en particular"; un ejemplo de una política es "cada página debe hacer su texto visible casi instantáneamente, y debería ser legible incluso si se redimensiona horizontalmente." Las funciones de ayuda de *JavaScript* y los estándares de codificación de todo el sitio pueden ayudar a realizar políticas de este tipo.

### 7.2.1. Transmitiendo objetos complejos y Parsing

#### HACIENDO PARSING EN JAVASCRIPT:

En este tutorial se muestra cómo separar una cadena cuando se encuentra un caracter concreto, en este caso será el punto y coma. Esto se hace mediante la utilización de la función `split(';')`, que crea un array de elementos. Posteriormente, mediante el uso de `trim()`, se pueden eliminar los espacios a ambos lados de una cadena de texto. En este tutorial se procede de la siguiente forma:

Se toma una variable llamada `dato_ejemplo` que contiene texto. En esta variable hay diferentes miembros que están separados por `;`, donde cada uno contiene un lugar y un precio (estos se diferencian porque el lugar está a la izquierda de `:` y su precio en dólares a la derecha).

El orden de actuación consiste en dividir cada miembro utilizando `Split` para `;`, posteriormente se utilizará `Split` de nuevo, en este caso para `:` , para poder trabajar con los lugares y precios de forma individual.

Para realizar un ejemplo sobre parsing a datos complejos se pueden seguir los siguientes pasos:

1. Se crea un archivo *HTML* llamado *L9-js6.html* con el siguiente contenido:

```
<!doctype html>
<html>
<head>
  <title>
    Parsing datos complejos
  </title>
  <script type = "text/javascript">
    // La siguiente declaración tiene
    // que aparecer en una línea simple.
    dato_ejemplo = "Mobil-17: 3.49; Kroger-03: 3.36; Exxon-01: 3.59; Kroger-04:
3.49;Valero-A: 3.41; Chevron-01: 3.52";
    de_interes = "Kroger-04";
    // dato_ejemplo es un ejemplo de típica
    // estructura de datos: una string con partes separadas
    // por semi-columnas, donde cada parte tiene dos
    // subpartes separadas por una columna. Otros formatos
    // son posibles. Para hacer parse a este formato particular,
    // init() separa en dos miembros distintos.
```



```

function init() {
    var objeto_parrafo = document.getElementById("parrafo");
    var lista_datos = dato_ejemplo.split(';');
    for (j = 0; j < lista_datos.length; j++) {
        partes = lista_datos[j].split(':');
        var lugar = partes[0].trim()
        if (lugar == de_interes) {
            var mensaje = "Datos los datos de ejemplo '" +
                dato_ejemplo + "', el programa ha analizado el precio $" +
                partes[1].trim() + " para el " + lugar;
            objeto_parrafo.innerHTML = mensaje;
        }
    }
}
</script>
</head>
<body onload = "init();">
    <h1>
        Parsing datos complejos
    </h1>
    <p id = "parrafo">
        Bienvenido.
    </p>
</body>
</html>

```

2. Se abre *L9-js6.html* en el navegador web. Se mostrará una pantalla como la de la siguiente imagen:

## Parsing datos complejos

Dados los datos de ejemplo 'Mobil-17: 3.49; Kroger-03: 3.36; Exxon-01: 3.59; Kroger-04: 3.49; Valero-A: 3.41; Chevron-01: 3.52', el programa ha analizado el precio \$3.49 para el Kroger-04

3. Se cierra el archivo *HTML*, pero se deja la herramienta de edición y el explorador web abiertos si se piensa completar el siguiente ejercicio de esta sesión.

## 7.3. Cargando y Guardando Archivos

### CARGANDO Y GUARDANDO ARCHIVOS CON JAVASCRIPT:

En el siguiente tutorial se muestra cómo introducir archivos mediante *JavaScript*. Se puede comprobar que hay un **input** con el siguiente código `onchange = 'handle_file_selection(this);'`, que invoca a la función cada vez que dicha entrada cambia. Utilizando `.files` se accede al archivo seleccionado, con `.name` se puede acceder al nombre del archivo. Utilizando `lastIndexOf('.')` se puede obtener la extensión del archivo seleccionado. Para comprobar si es un formato válido se utiliza `toLowerCase()`, ya que debemos estandarizar el formato antes de realizar las comprobaciones, ya que de lo contrario podría no reconocerse *JPG* u otro formato como imagen, ya que se realizan las comprobaciones con éste en minúscula.

Para realizar un ejemplo sobre acceder a archivos locales se pueden seguir los siguientes pasos:

1. Se crea un archivo *HTML* llamado *L9-js7.html* con el siguiente contenido:

```

<!doctype html>
<html>
<head>
    <title>

```



```

    JavaScript accede a archivos locales
</title>
<script type = "text/javascript">
function reconocer(file_handle)
{
    var tam = file_handle.size;
    var fnombre = file_handle.name;
    var mensaje = "Has elegido el archivo '" +
    fnombre + "'. Reconocible imagen, conteniendo " + tam + " bytes.";
    alert(mensaje);
}
function alertar(fnombre)
{
    var mensaje = "El nombre del archivo '" + fnombre + "'no es de extension
aceptable.";
    alert(mensaje);
}
function handle_file_selection(item)
{
    var f = item.files[0];
    var fnombre = f.name;
    var ultimo_indice = fnombre.lastIndexOf('.');
    if (ultimo_indice == -1)
    {
        alertar(fnombre);
        return;
    }
    var ext = fnombre.substr(ultimo_indice + 1);
    if (ext.toLowerCase() in {'gif': '', 'jpg': '', 'png': '', 'tif': ''})
    {
        reconocer(f);
    }
    else
    {
        alertar(fnombre);
    }
}
</script>
</head>
<body>
    <h1>
        JavaScript accede a archivos locales
    </h1>
    <input type = 'file' onchange = 'handle_file_selection(this);' />
</body>
</html>

```

2. Se abre el archivo anteriormente creado en un navegador web, donde se puede ver una pantalla como la que se muestra en la imagen siguiente.

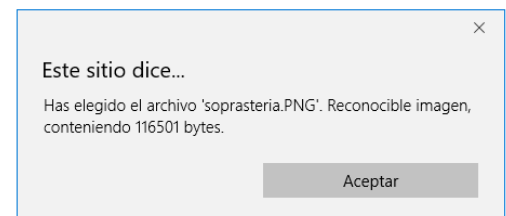
## JavaScript accede a archivos locales



3. Se hace clic en *Examinar* y, a continuación, se navega y selecciona un archivo en el equipo local que no sea una imagen. Se hace clic en *Abrir*. El mensaje le informa de que ha seleccionado un tipo de archivo incorrecto. Se hace clic en *Aceptar* para cerrar la ventana de diálogo cuadro de mensajes.
4. Se repite el paso 3 y esta vez se abre un archivo que es una imagen. El mensaje resultante debe tener un aspecto similar al de la siguiente imagen, informando de que se ha seleccionado una opción aceptable tipo de archivo. Haga clic en *Aceptar* para cerrar el cuadro de mensaje.

## JavaScript accede a archivos locales

d:\Profiles\jhecagutiern\ Examinar...



5. Se cierra el archivo *HTML*, pero se deja la herramienta de edición y el explorador web abiertos si se piensa completar el siguiente ejercicio de esta sesión.

### 7.3.1. Usando AppCache

## UTILIZACIÓN DE APPCACHE EN JAVASCRIPT:

En el siguiente tutorial se va a mostrar cómo poder ver una web cuando se está sin conexión, una vez se ha abierto la página con anterioridad.

Para ver el funcionamiento de AppCache se pueden seguir los siguientes pasos:

1. Se obtienen privilegios para usar un servidor Web. El servidor debe tener la directiva tipo *MIME* configurado para *text/cache-manifest*. *AppCache* sólo es efectivo para redes; no actúa cuando un recurso Web es local.
2. Se crea *L9-appcache.html* con el siguiente contenido:

```
<!doctype html>
<html manifest = "test.appcache">
<head>
  <title>Minimal AppCache example</title>
</head>
<body>
  <h1>
    Ejemplo de AppCache
  </h1>
```

```
<p> Esta web debe recargar tras desconectarse de internet y refrescarla.
</body>
</html>
```

3. Se cierra el archivo.
4. Se abre el archivo *L9-appcache.html*.
5. Se observa que se muestra una pantalla como la de la siguiente imagen.

## Ejemplo de AppCache

Esta web debe recargar tras desconectarse de internet y refrescarla.

6. Se desconecta el ordenador de internet.
7. Se confirma que no se puede acceder a páginas web.
8. Se refresca la imagen de *L9-appcache.html* en el navegador.
9. Se comprueba que se muestra rápidamente, incluso aunque *L9-appcache.html* ahora no está disponible.
10. Se cierra el archivo *HTML*, pero se deja la herramienta de edición y el explorador web abiertos si se piensa completar el siguiente ejercicio de esta sesión.

### 7.4. Usando JavaScript para Validar Campos de Formularios Introducidos por Usuarios

#### VALIDACIÓN DE CAMPOS DE FORMULARIOS EN JAVASCRIPT:

El siguiente tutorial valida la entrada del usuario, haciendo que ésta sea del formato deseado, en este caso XXX-XXX-XX-X, donde cada X representa un valor numérico. Utilizando `.length` se puede obtener el tamaño de la cadena de caracteres que se está introduciendo. Mediante `substring(longitud_actual - 1)` se obtiene el ultimo caracter introducido en el `input`.

Se puede comprobar que, al igual que en otros lenguajes de programación se pueden utilizar estructuras condicionales `switch`, que nos permite seleccionar el código a realizar en caso de que una variable tome unos valores u otros. También se puede ver que para comprobar si el carácter es numérico se utiliza `!/^\d/.test()`.

Para realizar esto por JavaScript de forma instantánea se pueden seguir los pasos del siguiente tutorial:

1. Se crea un archivo llamado *L9-js8.html* con el siguiente contenido:

```
<!doctype html>
<html>
<head>
<meta charset="UTF-8">
  <title>
    Formulario
  </title>
  <script type = "text/javascript">
    // La función correcto()
```



```

// hace un test de lo que introduce el usuario
// con la máscara 'XXX-XXX-XX-X'. Si
// el usuario introduce valores distintos
// de la máscara se borra el último caracter. Esto
// da impresión al usuario de que solo debe
// introducir caracteres válidos.
function correcto()
{
    var objeto_entrada = document.getElementById("serie");
    var valor = objeto_entrada.value;
    var longitud_actual = valor.length;
    if (longitud_actual)
    {
        var ultimo_caracter = valor.substring(longitud_actual - 1);
        switch (longitud_actual)
        {
            case 4:
            case 8:
            case 11:
                if (ultimo_caracter != '-')
                {
                    valor = valor.substring(0, longitud_actual - 1);
                }
                break;
            default:
                if (!/\d/.test(ultimo_caracter))
                {
                    valor = valor.substring(0, longitud_actual - 1);
                }
        }
        if (longitud_actual > 12)
        {
            valor = valor.substring(0, longitud_actual - 1);
        }
        longitud_actual = valor.length;
        switch (longitud_actual)
        {
            case 3:
            case 7:
            case 10:
                valor += "-";
        }
        objeto_entrada.value = valor;
    }
}
</script>
</head>
<body>
    <h1>
        Formulario
    </h1>
    <form>
        Introduce el número de serie con la máscara XXX-XXX-XX-X,
        Donde cada X es una cifra: <input id = "serie" type = 'text' size = '12' onkeyup
        ="correcto();">.
    </form>
</body>
</html>

```



2. Se abre el archivo creado en un navegador web. La pantalla se mostrará como en la imagen que se ve a continuación.

## Formulario

Introduce el número de serie con la máscara XXX-XXX-XX-X, Donde cada X es una cifra: .

3. Se comienza a introducir caracteres. Se comprueba que el campo de entrada acepta solamente caracteres que tengan la forma XXX-XXX-XX-X, ignorando otros casos.
4. Se cierra el archivo *HTML*, pero se deja la herramienta de edición y el explorador web abiertos si se piensa completar el siguiente ejercicio de esta sesión.

Una aplicación del mundo real requeriría más validaciones: no basta con que una entrada se parezca a un número de serie, la aplicación también debe confirmar que se trata de un número de serie. Si se obliga a rectificar mientras que un usuario introduce un valor de forma incorrecta, la validación *JavaScript* de las entradas de formulario ayudan a asegurar el éxito.

### 7.5. Entendiendo y Usando Cookies

#### USO DE COOKIES EN JAVASCRIPT:

En el siguiente ejercicio se utilizarán las cookies para recordar contenido introducido por el usuario. Se puede utilizar `document.cookies` para crear, leer y borrar cookies. Se puede introducir el nombre del usuario y la fecha de expiración de ésta.

Para comprobar el uso de cookies en *JavaScript* se pueden seguir los siguientes pasos:

1. Se crea un archivo *HTML* llamado *L9-js9.html* con el siguiente contenido:

```
<!doctype html>
<html>
<head>
  <title>
    Uso de cookies
  </title>
  <script type = "text/javascript">
    function obtenerCookie(c_nombre)
    {
      var i,x,y,ARRcookies = document.cookie.split(";");
      for (i=0; i < ARRcookies.length;i++)
      {
        x = ARRcookies[i].substr(0,ARRcookies[i].indexOf("="));
        y = ARRcookies[i].substr(ARRcookies[i].indexOf("=")+1);
        x = x.replace(/^\s+|\s+$/g,"");
        if (x == c_nombre)
        {
          return unescape(y);
        }
      }
    }
    function init()
    {
```



```

        var mensaje;
        objeto_nivel = document.getElementById("nivel");
        var bienvenido = document.getElementById("bienvenido");
        var nivel = obtenerCookie("nivel");
        if (nivel == null || nivel == '')
        {
            mensaje = "Es tu primera vez, se empieza por el nivel 1";
            nivel = 1;
        }
        else
        {
            mensaje = "La pasada vez llegaste al nivel " + nivel + ". Vas a empezar
ahora.";
        }
        bienvenido.innerHTML = mensaje;
        objeto_nivel.value = nivel;
    }
    function guarda_nivel()
    {
        ponerCookie("nivel", objeto_nivel.value, 10);
    }
    function ponerCookie(c_nombre,valor,exdias)
    {
        var exfecha = new Date();
        exfecha.setDate(exfecha.getDate() + exdias);
        var c_valor = escape(valor) + ((exdias == null) ? "": " ");
        expires="+exfecha.toUTCString());
        document.cookie = c_nombre + "=" + c_valor;
    }
</script>
</head>
<body onload = "init();">
    <h1>Uso de cookies</h1>
    <p id = "bienvenido">Bienvenido.</p>
    <form>
        Puedes actualizar tu nivel. Ahora mismo es
        <input id = "nivel" type = "number" min = "1" max = "100" oninput =
"guarda_nivel();" />.
    </form>
</body>
</html>

```

2. Se obtienen privilegios para usar un servidor web. Las cookies sólo pueden ser habilitadas para páginas web a las que se accede a través de la red, no localmente.
3. Se abre *L9-js9.html*.
4. Se Observa que como es la primera vez que se entra, se comenzaría por el nivel 1.
5. Se supone que se ha jugado durante un tiempo y se ha llegado a un nivel diferente. Se introduce ese nivel en el área de entrada.

## Uso de cookies

La pasada vez llegaste al nivel 87. Vas a empezar ahora.

Puedes actualizar tu nivel. Ahora mismo es .





6. Se cierra la ventana de la aplicación.
7. Se abre *L9-js9.html* de nuevo. Se comprueba que la aplicación recuerda la posición de una sesión a otra.
8. Se utiliza la configuración de preferencia del navegador, se eliminan las cookies creadas en este ejercicio o, si es más conveniente, se eliminan todas las cookies.
9. Se abre el archivo de nuevo. Se comprueba que el valor ha vuelto al valor predeterminado.
11. Se cierra el archivo *HTML*, pero se deja la herramienta de edición y el explorador web abiertos si se piensa completar el siguiente ejercicio de esta sesión.

## 7.6. Entendiendo y Usando Almacenamiento Local

### USO DE ALMACENAMIENTO LOCAL EN JAVASCRIPT:

**localStorage** guarda datos sin fecha de expiración. Los datos no serán eliminados del navegador cuando éste se cierre, y estarán disponibles el día siguiente, semana o incluso año. En este tutorial se usará el almacenamiento local para almacenar el nivel introducido.

Para realizar un ejemplo donde se guarde información sin utilizar cookies se puede realizar el siguiente tutorial siguiendo los siguientes pasos:

1. Se crea un archivo *L9-js10.html* con el siguiente contenido:

```
<!doctype html>
<html>
<head>
  <title>
    Uso de almacenamiento local
  </title>
  <script type="text/javascript">
    // Esta página muestra un modelo para
    // guardar niveles de jugadores en almacenamiento local
    function init()
    {
      var mensaje;
      objeto_nivel = document.getElementById("nivel");
      var bienvenido = document.getElementById("bienvenido");
      // "localStorage" es una keyword de HTML5
      // "localStorage.nivel" es una variable elegida
      // para este lugar el particular

      var nivel = localStorage.nivel;
      if (nivel == null || nivel == '')
      {
        mensaje = "Aparece cuando se juega por primera vez. Se empieza en el
nivel 1";
        nivel = 1;
      }
      else
      {
        mensaje = "La pasada vez que se ha jugado se ha llegado al nivel " +
nivel + ". Se va a empezar en este nivel.";
      }
      bienvenido.innerHTML = mensaje;
      objeto_nivel.value = nivel;
    }
    function guarda_nivel()
```



```

        {
            localStorage.nivel = objeto_nivel.value;
        }
    </script>
</head>
<body onload="init();">
    <h1>
        Uso de almacenamiento local
    </h1>
    <p id="bienvenido">Bienvenido.</p>
    <form>
        Puedes actualizar tu nivel. Ahora mismo es:
        <input id="nivel" type="number" min="1" max="100" oninput="guarda_nivel();" />.
    </form>
</body>
</html>

```

2. Se obtienen los privilegios para usar un servidor web. El almacenamiento local sólo puede ser habilitado para páginas web a las que se accede través de la red, no localmente.
3. Se carga el archivo anteriormente creado al servidor web
4. En un navegador Web, utilice la dirección Web para navegar y abrir *L9-js10.html*.
5. Observe el nivel al que comienza, como se muestra en la imagen siguiente.

## Uso de cookies

La pasada vez llegaste al nivel 3. Vas a empezar ahora.

Puedes actualizar tu nivel. Ahora mismo es .

6. Se finge que se ha jugado durante un tiempo y se ha llegado a un nivel diferente. Se introduce ese nivel en el área de entrada.
7. Se cierra la ventana de la aplicación.
8. Se abre *L9-js10.html* de nuevo. Se ve cómo la aplicación recuerda la posición de una sesión de navegador a otra.
9. Se cierra el archivo y cualquier programa o ventana abierta.

