

Tema 9. Edición inline

Contenido

Tema 9. Edición inline.....	1
1) Introducción.....	1
2) Opciones disponibles.....	1
3) Creando un editor de texto.....	6
4) Ejercicios	13
Ejercicio 1: Crear un sistema para publicar mensajes con un editor para los mensajes a introducir	13
Pasos a seguir	14

1) Introducción

Vamos a estudiar en esta unidad, otra de las novedades que nos ofrece HTML5 como es la edición inline. En este caso, se trata de la posibilidad de hacer editable cualquier elemento que tengamos en nuestra página, desde el propio navegador. Hasta ahora la única opción que disponíamos donde el usuario podía introducir contenido, eran los campos de texto, o campos de entrada de los formularios.

Pues bien, en este caso, se nos ofrece la posibilidad de hacer editable cualquier elemento de nuestra página. Podemos permitir al usuario introducir texto en un elemento, o bien, modificar el contenido que ya contiene. Tenemos la opción además, de ofrecerle una corrección ortográfica automática, como si de un editor de texto se tratara.

Como en la mayoría de las novedades que se ofrecen en HTML5, en este caso también, será necesario el uso de un lenguaje de script como JavaScript. Mediante dicho lenguaje, y utilizando las librerías disponibles para ello, vamos a ser capaces de crear nuestro propio editor de texto, para dar formato al contenido que el usuario pueda introducir, o para modificar el contenido que ya contenga cualquier otro elemento.

Por lo tanto, podremos ofrecerle a un usuario, la posibilidad de modificar el contenido de una página a su gusto, o la posibilidad de introducir contenido y darle el formato deseado, desde el propio navegador.

¡IMPORTANTE!

Podemos consultar todos los aspectos relacionados con [Edición Inline](#) en la recomendación W3C desde este enlace: [Edición Inline](#)

2) Opciones disponibles

En primer lugar, deberemos definir cuáles serán los elementos editables, y a partir de ahí ofrecer al usuario las diferentes opciones que disponemos. Estos tres atributos serán muy importantes para ello:

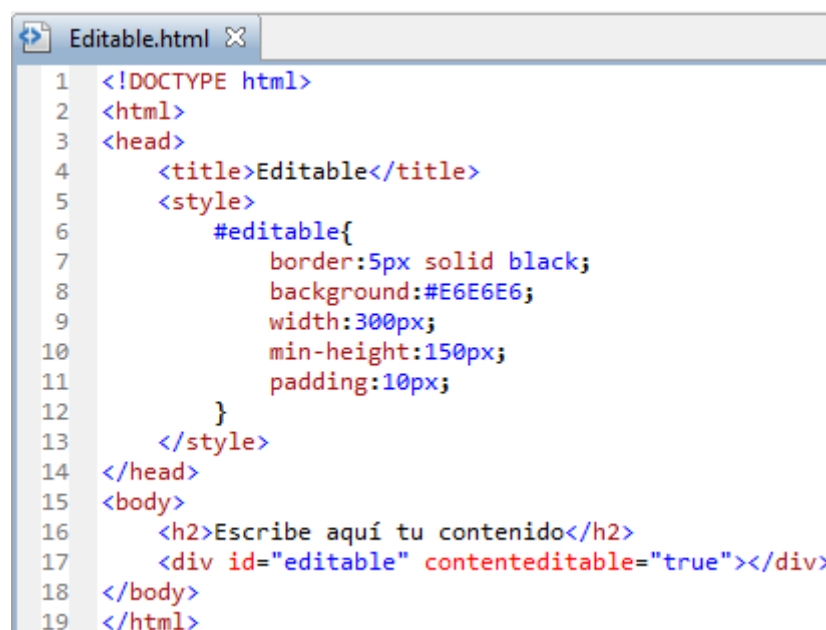
contenteditable: Atributo global de HTML5 que convierte el elemento en editable. Los valores permitidos son *true*, *false* o *inherit*. El primero hace al elemento editable, el segundo no permite la edición, y el tercero, lo permite si su elemento padre es editable.

spellcheck: Atributo global de HTML5 que activa la corrección ortográfica para dicho elemento editable. Los valores permitidos son *true* o *false*.

designMode: Atributo que hace editable todo el documento. Los valores permitidos son *on* u *off*.

Los dos primeros, al ser atributos globales, podemos aplicarlos a cualquier elemento de nuestra página. Y el tercero, basándonos en lo que hemos explicado, se lo aplicaremos al documento como tal, aunque cabe la posibilidad de aplicárselo al contenido de un *iFrame* como veremos más adelante.

Empezaremos estudiando el funcionamiento del primero de ellos, **contenteditable**. Vamos a crear una página con un elemento *DIV*, y a dicho elemento le aplicaremos este atributo, convirtiéndolo así en editable. Por lo tanto el código HTML quedaría de la siguiente manera:

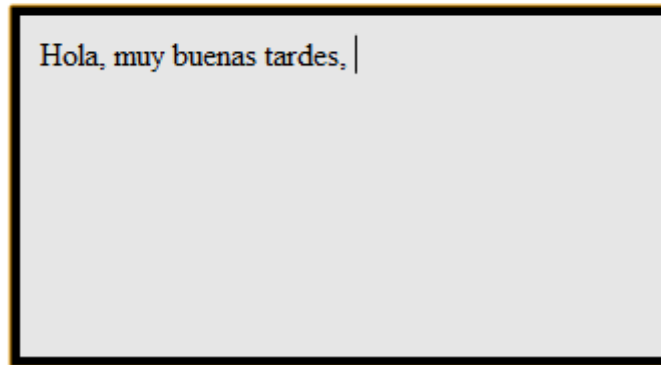
A screenshot of a code editor window titled 'Editable.html'. The code is as follows:

```
1 <!DOCTYPE html>
2 <html>
3 <head>
4   <title>Editable</title>
5   <style>
6     #editable{
7       border:5px solid black;
8       background:#E6E6E6;
9       width:300px;
10      min-height:150px;
11      padding:10px;
12    }
13  </style>
14 </head>
15 <body>
16   <h2>Escribe aquí tu contenido</h2>
17   <div id="editable" contenteditable="true"></div>
18 </body>
19 </html>
```

Como vemos, únicamente hemos definido nuestro elemento *DIV*, al que le hemos dado el formato deseado desde el cabecero. Le hemos aplicado el atributo *contenteditable*, con un valor *true*. De esta manera, al focalizar el elemento podremos escribir sobre él.

Vemos como quedaría el resultado en el navegador:

Escribe aquí tu contenido



En el momento que hacemos clic sobre el elemento, vemos que podemos escribir sobre él sin ningún problema, o bien editar el contenido si lo tuviera. Al igual que se lo hemos aplicado a un elemento *DIV*, podemos aplicárselo a cualquier otro elemento de nuestra página.

El segundo de los atributos, **spellcheck**, define si dotamos a un elemento editable de un corrector de ortografía o no. Por defecto, cuando hacemos un elemento editable, el corrector está activado, por lo que no sería necesario especificárselo mediante el atributo *spellcheck*. Pero quizá no queramos que dicho corrector este disponible, en ese caso deberemos especificarlo con dicho atributo y un valor *false*.

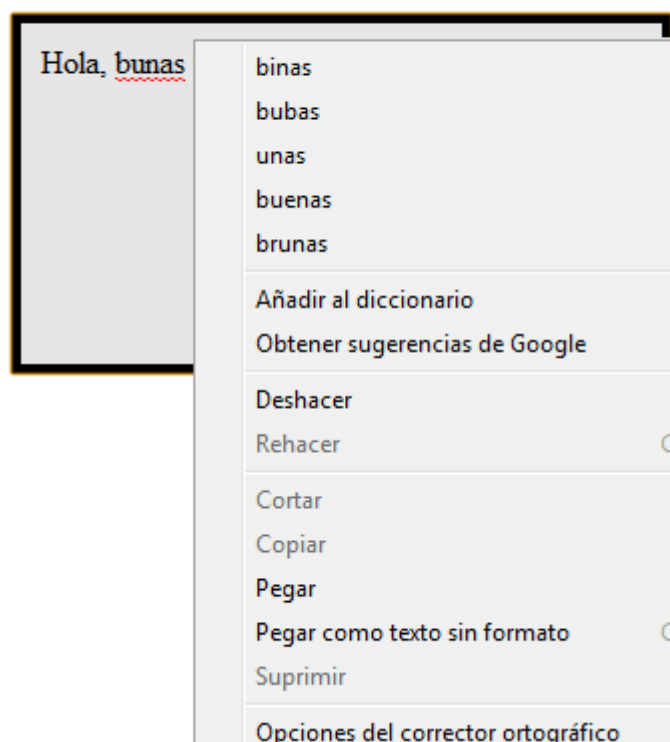
En el ejemplo anterior, el código del elemento editable quedaría de la siguiente manera:

```
<div id="editable" contenteditable="true" spellcheck="true"></div>
```

Vemos que no es más que un simple atributo con su valor *true*. Como hemos indicado, si quisiéramos que el corrector no apareciera, le daríamos el valor *false*.

Este sería el resultado en el navegador en el momento que escribimos algo mal:

Escribe aquí tu contenido



En el momento que escribimos una palabra mal, el navegador la marca en rojo, como hace cualquier editor de texto. Si hacemos clic en el botón derecho sobre la palabra errónea, se despliega el menú que vemos en la imagen, con diferentes sugerencias para dicha palabra, y otras opciones que podemos utilizar.

Y por último, el atributo **designMode**, que como hemos dicho convertirá en editable todo el documento. Por lo tanto, aplicaremos dicho atributo al documento desde JavaScript, como vemos a continuación:



```
1 <!DOCTYPE html>
2 <html>
3 <head>
4   <title>Editable</title>
5   <style>
6     #editable{
7       border:5px solid black;
8       background:#E6E6E6;
9       width:300px;
10      min-height:150px;
11      padding:10px;
12    }
13  </style>
14
15  <script language="JavaScript">
16    window.onload = function (){
17      document.designMode = "on";
18    }
19  </script>
20
21 </head>
22 <body>
23   <h2>Escribe aquí tu contenido</h2>
24   <div id="editable"></div>
25 </body>
26 </html>
```

De esta manera, hacemos editable el documento completo, por lo que serán editables también, todos los elementos que contiene, con las mismas características definidas anteriormente.

Anotación

Otra posibilidad es, crear un iFrame donde cargaremos nuestra página, y hacer editable mediante el atributo designMode el contenido del iFrame. De esta manera, también estaríamos haciendo editable todo el documento que contiene dicho iFrame.

```
iEditable.html x
1  <!DOCTYPE html>
2  <html>
3  <head>
4      <title>iEditable</title>
5      <style>
6          #pagina{
7              width:345px;
8              min-height:260px;
9          }
10     </style>
11
12     <script language="JavaScript">
13         window.onload = function (){
14             document.getElementById("pagina").contentWindow.document.designMode
15         }
16     </script>
17
18 </head>
19 <body>
20     <iframe id="pagina" src="Editable.html"></iframe>
21 </body>
22 </html>
```

Vemos como cargamos sobre el *iFrame* la página *Editable.html*, y activamos el atributo *designMode* desde JavaScript para el contenido del *iFrame*. De esta manera, todo el documento cargado sobre el *iFrame* es editable.

El uso de *designMode* utilizando *iFrames*, no funciona en **Chrome**, a no ser que lo ejecutemos en servidor. Y para que ejecutándolo en servidor funcione en cualquiera de los navegadores, a excepción de **Opera** que no funciona, la página que carguemos sobre dicho *iFrame* debe de estar en el mismo dominio que la página que contiene el propio *iFrame*.

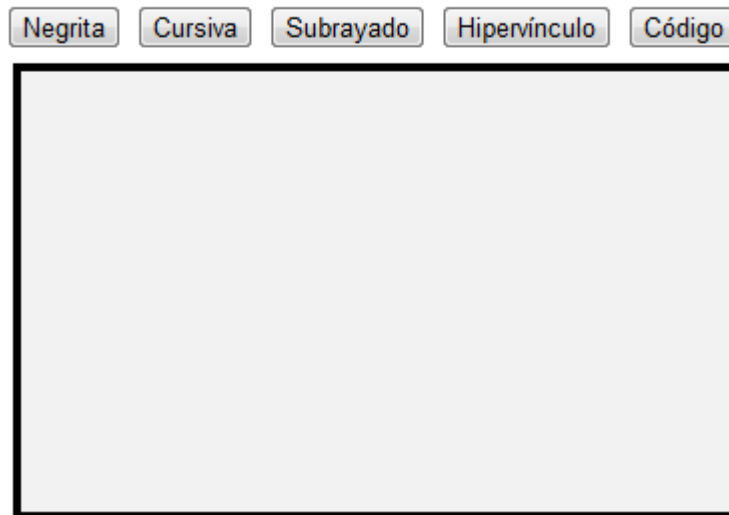
3) Creando un editor de texto

Hemos visto ya como hacer editable un elemento, o un documento entero, y la posibilidad de corregir dicho contenido como si de un editor de texto se tratara. Pues bien, ahora vamos a ver como dar formato a ese contenido editable. Vamos a crear un verdadero editor de texto para los elementos de nuestra página que queramos editar y dar formato. Iremos creando los diferentes botones de formato, y las funciones que lo ejecutarán. Estas funciones, serán creadas desde JavaScript, y para este trabajo, tendrá especial importancia el método **execCommand()**, disponible desde este lenguaje de script.

Este método recibe tres valores: **execCommand(comando, interfaz, valor)**. El primero de ellos, es el comando a aplicar, el segundo indica si necesitamos una interfaz gráfica de usuario, en caso contrario tendrá un valor *false*, y el tercero es un valor que dependiendo del comando será necesario, en caso contrario será *null*.

Comenzaremos creando el código HTML. Vamos a crear un elemento *DIV*, que será sobre el que vayamos introduciendo contenido, y crearemos también varios botones que aplicarán el formato básico a dicho contenido. Lo que queremos conseguir es lo siguiente:

Editor personalizado



Vemos que hemos creado un contenedor que será editable, y por lo tanto es ahí donde iremos introduciendo o modificando el contenido. En la parte superior, hemos colocado cinco botones, el primero de ellos pondrá nuestro texto en negrita, el segundo lo pondrá en cursiva y el tercero lo subrayará. El cuarto botón, convertirá el texto seleccionado en un enlace, o si no tenemos texto seleccionado creará un enlace. Y por último, el quinto botón, nos mostrará una alerta con el código HTML del contenido del *DIV*.

El código que hemos utilizado para conseguirlo es el siguiente:

```
EditorInline.html X
1 <!DOCTYPE html>
2 <html>
3 <head>
4   <title>Editor Inline</title>
5
6   <style>
7     #contenido{
8       border:4px solid black;
9       background:#F2F2F2;
10      width:335px;
11      min-height:200px;
12      padding:10px;
13      margin:5px;
14    }
15  </style>
16
17 </head>
18 <body>
19   <h2>Editor personalizado</h2>
20   <div>
21     <input type="button" value="Negrita" onclick="negrita()">
22     <input type="button" value="Cursiva" onclick="cursiva()">
23     <input type="button" value="Subrayado" onclick="subrayado()">
24     <input type="button" value="Hipervínculo" onclick="hipervinculo()">
25     <input type="button" value="Código" onclick="codigo()">
26   </div>
27
28   <div id="contenido" contenteditable="true"></div>
29 </body>
30 </html>
```

Por un lado tenemos un contenedor con todos los botones, y por otro el contenedor editable. A este último, le hemos aplicado formato desde el cabecero mediante la etiqueta *STYLE*. Vemos como los botones, tienen asignada una función que se ejecutará al hacer clic sobre ellos, y que iremos creando a continuación desde JavaScript.

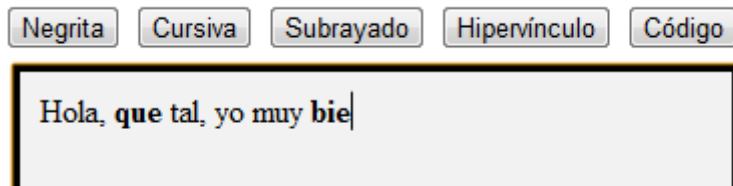
Por lo tanto, una vez creada la estructura y los elementos que queremos, vamos ir creando una por una las funciones que se ejecutarán al hacer clic sobre dichos botones.

Comenzaremos por la función *negrita()*, que evidentemente convertirá nuestro texto a negrita. Este sería el código:

```
19 function negrita ()
20 {
21   document.execCommand("bold", false, null);
22 }
```

Simplemente ejecutamos el método *execCommand()*, que recibe tres parámetros, y que como hemos dicho, el primero de ellos es el comando a aplicar sobre nuestro texto editable, en este caso *bold*. De esta manera, tanto si seleccionamos un texto y pulsamos el botón, como si lo pulsamos y después escribimos, el texto aparecerá en negrita.

Si lo vemos en funcionamiento:



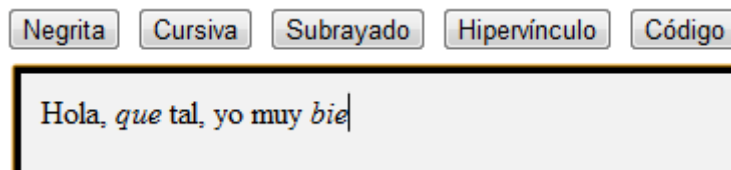
En primer lugar hemos seleccionado la palabra "que" y le hemos aplicado negrita pulsando el botón, y después, al pulsar el botón sin texto seleccionado, cuando escribimos, lo hace ya en negrita como vemos. El funcionamiento por lo tanto es idéntico al de cualquier editor de texto.

Vamos ahora con la función *cursiva()*, que evidentemente convertirá nuestro texto a cursiva. Este sería el código:

```
24     function cursiva ()
25     {
26         document.execCommand("italic", false, null);
27     }
```

Es exactamente igual que en el caso de la negrita, pero en este caso el comando es *italic*.

Si lo vemos en funcionamiento:



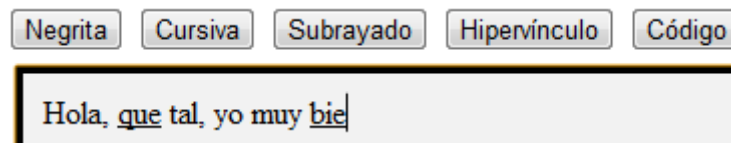
En primer lugar hemos seleccionado la palabra "que" y le hemos aplicado cursiva pulsando el botón, y después, al pulsar el botón sin texto seleccionado, cuando escribimos, lo hace ya en cursiva como vemos. El funcionamiento en este caso también, es idéntico al de cualquier editor de texto.

Vamos pues con la función *subrayado()*, que evidentemente subrayará nuestro texto. Este sería el código:

```
29     function subrayado ()
30     {
31         document.execCommand("underline", false, null);
32     }
```

Es exactamente igual que en el caso de la negrita o cursiva, pero en este caso el comando es *underline*.

Si lo vemos en funcionamiento:



En primer lugar hemos seleccionado la palabra "que" y la hemos subrayado pulsando el botón, y después, al pulsar el botón sin texto seleccionado, cuando escribimos, lo hace ya con el texto

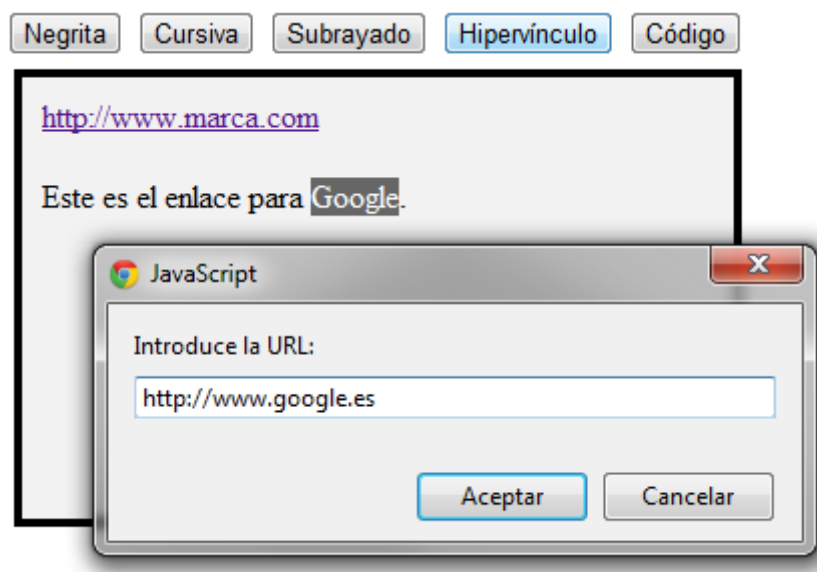
subrayado como vemos. El funcionamiento en este caso también, es idéntico al de cualquier editor de texto.

Hemos visto ya por lo tanto, las tres funciones que nunca faltan en los editores de texto, vamos ahora con la función en la que crearemos un enlace, y que es la función *hipervinculo()*. Este sería el código:

```
34     function hipervinculo()  
35     {  
36         var direccion = prompt("Introduce la URL:", "http://");  
37         document.execCommand("createlink", false, direccion);  
38     }
```

En este caso, cuando pulsamos el botón, lo primero que hacemos es solicitarle al usuario la dirección a la que quiere crear el enlace, lo hacemos mediante la función *prompt()*. Una vez introducida la dirección por parte del usuario, la almacenamos en una variable, y ejecutamos el método *execCommand()*, pero en este caso el comando será *createlink*. Vemos que para este comando, es necesario el tercer parámetro, donde como vemos, le pasaremos la dirección introducida por el usuario. De esta manera, si tenemos seleccionado un texto, será ese el texto que convierta en enlace, pero sino tenemos nada seleccionado, creará un enlace con la dirección que hemos introducido.

Si lo vemos en funcionamiento:



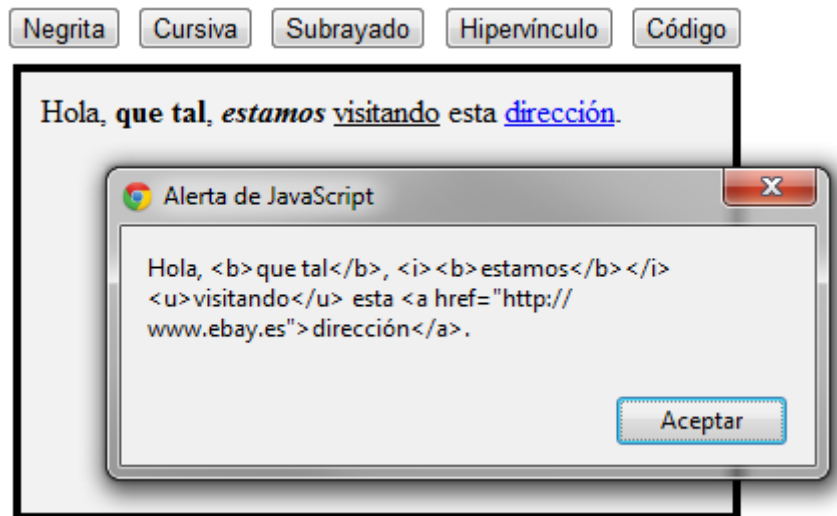
En primer lugar hemos creado un enlace a "http://www.marca.com" sin tener ningún texto seleccionado, y como vemos, se ha creado con la dirección que hemos introducido. Después sin embargo, hemos seleccionado la palabra "Google", y hemos pulsado el botón de hipervínculo. En ese momento, nos solicita que introduzcamos la dirección para el enlace, y convierte la palabra seleccionada en hipervínculo a esa dirección.

Por último, vamos a ver la función *codigo()*, que nos mostrará una alerta con el código HTML del contenido del *DIV*. Este sería el código:

```
40     function codigo ()  
41     {  
42         var conte = document.getElementById("contenido").innerHTML;  
43         alert(conte);  
44     }
```

En este caso, lo único que hacemos es recoger el contenido del *DIV* mediante la propiedad *innerHTML*, y mostrar una alerta con ello.

Si lo vemos en funcionamiento:



Vemos que tenemos un texto al que le hemos aplicado un formato, y al pulsar el botón para ver el código, nos salta una alerta que lo muestra. El código mostrado, es el código HTML del contenido del elemento *DIV* y como vemos tiene ya todas las etiquetas que aplican el formato seleccionado utilizando los botones que hemos ido creando. En caso de querer guardar el contenido introducido y editado, este sería el código que almacenaríamos.

Por tanto, ya tenemos un pequeño editor de texto en funcionamiento, para poder así, aplicar formato al contenido de cualquier elemento editable.

Anotación

Aparte de los comandos utilizados ya, el método *execCommand()* soporta muchos más. A continuación tenemos un listado con los más importantes:

backcolor: Comando que cambia el color de fondo del contenido seleccionado, o del que escribamos a partir de que lo activemos. Necesita como valor el nombre o el código del color que queramos.

bold: Comando que escribe en negrita el contenido seleccionado, o del que escribamos a partir de que lo activemos.

createlink: Comando que convierte en enlace el contenido seleccionado, o que inserta un enlace al contenido. Necesita como valor la URL que queramos para dicho enlace.

delete: Comando que elimina el contenido seleccionado.

fontname: Comando que cambia el tipo de la fuente del contenido seleccionado, o del que escribamos a partir de que lo activemos. Necesita como valor el nombre de la fuente que queramos.

fontsize: Comando que cambia el tamaño de la fuente del contenido seleccionado, o del que escribamos a partir de que lo activemos. Necesita como valor el tamaño que queramos.

forecolor: Comando que cambia el color de la fuente del contenido seleccionado, o del que escribamos a partir de que lo activemos. Necesita como valor el nombre o el código del color que queramos.

formatblock: Comando que formatea a un tipo de bloque predefinido el contenido seleccionado, o del que escribamos a partir de que lo activemos. Necesita como valor el nombre del bloque que queramos.

inserthorizontalrule: Comando que inserta la regla horizontal al elemento editable.

insertimage: Comando que inserta una imagen al contenido. Necesita como valor la ruta de la imagen que queramos insertar.

insertorderedlist: Comando que inserta una lista ordenada al contenido.

insertunorderedlist: Comando que inserta una lista no ordenada al contenido.

insertparagraph: Comando que inserta un párrafo al contenido.

italic: Comando que escribe en cursiva el contenido seleccionado, o del que escribamos a partir de que lo activemos.

justifycenter: Comando que justifica el contenido en el centro.

justifyfull: Comando que justifica el contenido de manera general.

justifyleft: Comando que justifica el contenido en la izquierda.

justifyright: Comando que justifica el contenido en la derecha.

strikethrough: Comando que escribe tachado el contenido seleccionado, o del que escribamos a partir de que lo activemos.

subscript: Comando que escribe en subíndice el contenido seleccionado, o del que escribamos a partir de que lo activemos.

superscript: Comando que escribe en superíndice el contenido seleccionado, o del que escribamos a partir de que lo activemos.

underline: Comando que escribe subrayado el contenido seleccionado, o del que escribamos a partir de que lo activemos.

unlink: Comando que deshace un enlace seleccionado.

¡IMPORTANTE!

Si en lugar de tener un elemento editable, tuviéramos el documento entero, la forma de aplicarle formato sería exactamente la misma.

Pero si lo hemos hecho mediante el uso de un iFrame como hemos explicado antes, tendríamos que aplicarle el formato al contenido del iFrame, y no directamente al documento. En la siguiente imagen se ve más claro:

```
document.getElementById("pagina").contentWindow.document.execCommand("bold", false, null);
```

En este caso, el iFrame tiene como identificador "pagina", y le estamos aplicando negrita al contenido de ese iFrame.

¡IMPORTANTE!

Esta nueva opción que nos ofrece HTML5 como es la Edición Inline, junto a Drag&Drop, nos permitirá hacer una página totalmente editable. Por un lado, podremos desplazar y

recolocar los elementos gracias a Drag&Drop, y por otro lado editar y dar formato a cualquiera de los elementos gracias a la Edición Inline.

Además, teniendo una buena estructura en servidor, podremos publicar los cambios realizados sobre nuestra página de manera directa. Tanto es así, que por ejemplo, el gestor de contenidos OpenCms, ofrece ya un gestor para configurar nuestras páginas, basado en estas dos tecnologías. Podemos leer algo más sobre el tema en este enlace.

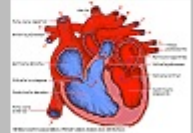
4) Ejercicios

Ejercicio 1: Crear un sistema para publicar mensajes con un editor para los mensajes a introducir

Vamos a crear un sistema donde vamos a ir publicando los mensajes que el usuario escribe. Para poder escribirlos con el formato deseado, facilitaremos al usuario un editor de texto que vamos a crear de la manera que hemos estudiado en la unidad. El contenido que el usuario escriba, será publicado en el apartado de mensajes, tal y como lo haya escrito, con el formato que le haya dado. Hay que tener en cuenta, que no estamos creando ningún sistema de almacenamiento, y que en el momento que recargamos la página, los mensajes se borrarán. Lo que queremos conseguir es lo que vemos en la siguiente imagen:

Mensajes publicados

La [App Store](#) es quizás la tienda de aplicaciones con mejores **clientes de Twitter**, grandes ejemplos como *Tweetbot* o *Twittelator Neue* **son** simplemente excepcionales.



La razón es que un corazón grande puede **bombear más oxígeno** a los músculos. Y estos músculos tienen mucha más **capacidad de resistencia** al usar ese oxígeno de manera inmediata.

Publica aquí tu mensaje



PUBLICAR

Pasos a seguir

1. En primer lugar, estructuraremos la página para conseguir el formato que vemos en la imagen.
2. Una vez estructurada, crearemos el espacio donde vamos a permitir escribir y formatear los mensajes.

3. Después, crearemos el editor con las opciones que vemos en la imagen. En este caso, en lugar de botones para ejecutar las funciones de formato, utilizaremos las imágenes que se proporcionan.
4. Una vez hecho esto, crearemos una por una las funciones que darán el formato deseado al contenido.
5. Para terminar, nos faltará crear la función que se ejecute cuando pulsemos el botón publicar. Esta, recogerá el contenido que el usuario haya escrito, y los publicará en la zona de mensajes.