Programador Web – Nivel 1

Unidad 4: Maquetación Web

Contacto: consultas@elearning-total.com
Web: www.elearning-total.com

Indice

Unidad 4: Maquetación web

Objetivos

Que el alumno logre:

- Aplicar los atributos de estilos CSS a la estructura del sitio a través de la utilización de elementos semánticos.

Posicionamiento y visualización

Cuando los navegadores descargan el contenido **HTML** y **CSS** de las páginas web, aplican un procesamiento muy complejo antes de mostrar las páginas en la pantalla del usuario.

Para cumplir con el modelo de cajas presentado, los navegadores crean una caja para representar a cada elemento de la página HTML. Los factores que se tienen en cuenta para generar cada caja son:

.: Las propiedades **width** y **height** de la caja (si están establecidas).
.: El **tipo** de cada elemento HTML (elemento de bloque o elemento en línea).
.: Posicionamiento de la caja (normal, relativo, absoluto o fijo).
.: Las **relaciones** entre elementos (dónde se encuentra cada elemento, elementos descendientes, etc.)
.: Otro tipo de información, como por ejemplo el **tamaño de las imágenes** y el **tamaño de la ventana del navegador**.

Veremos los **cuatro tipos de posicionamientos** definidos para las cajas y otras propiedades que afectan a la forma en la que se visualizan las cajas.

TIPOS DE ELEMENTOS

El estándar HTML clasifica a todos sus elementos en dos grandes grupos: **elementos en línea** y **elementos de bloque**.

Los **elementos de bloque** (“block elements” en inglés) siempre empiezan en una nueva línea y ocupan todo el espacio disponible hasta el final de la línea. Por su parte, los **elementos en línea** (“inline elements” en inglés) no empiezan necesariamente en nueva línea y sólo ocupan el espacio necesario para mostrar sus contenidos.

Debido a este comportamiento, el tipo de un elemento influye de forma decisiva en la caja que el navegador crea para mostrarlo.

Los elementos en línea definidos por HTML son: **a, abbr, acronym, b, basefont, bdo, big, br, cite, code, dfn, em, font, i, img, input, kbd, label, q, s, samp, select, small, span, strike, strong, sub, sup, textarea, tt, u, var**.

Los elementos de bloque definidos por HTML son: **address, blockquote, center, dir, div, dl, fieldset, form, h1, h2, h3, h4, h5, h6, hr, isindex, menu, noframes, noscript, ol, p, pre, table, ul**.

Los siguientes elementos también se considera que son de bloque: **dd, dt, frameset, li, tbody, td, tfoot, th, thead, tr.**

Los siguientes elementos pueden ser en línea y de bloque según las circunstancias: **button, del, iframe, ins, map, object, script.**

POSICIONAMIENTO

Los navegadores crean y posicionan de forma automática todas las cajas que forman cada página HTML. No obstante, CSS permite al diseñador modificar la posición en la que se muestra cada caja.

Utilizando las propiedades que proporciona CSS para alterar la posición de las cajas es posible realizar efectos muy avanzados y diseñar estructuras de páginas que de otra forma no serían posibles.

El estándar de CSS define **cuatro modelos** diferentes para posicionar una caja:

.: Posicionamiento normal o estático: se trata del posicionamiento que utilizan los navegadores si no se indica lo contrario.

.: Posicionamiento relativo: variante del posicionamiento normal que consiste en posicionar una caja según el posicionamiento normal y después desplazarla respecto de su posición original.

.: Posicionamiento absoluto: la posición de una caja se establece de forma absoluta respecto de su elemento contenedor y el resto de elementos de la página ignoran la nueva posición del elemento.

.: Posicionamiento fijo: variante del posicionamiento absoluto que convierte una caja en un elemento inamovible, de forma que su posición en la pantalla siempre es la misma independientemente del resto de elementos e independientemente de si el usuario sube o baja la página en la ventana del navegador.

El posicionamiento de una caja se establece mediante la propiedad **position**.

El significado de cada uno de los posibles valores de la propiedad position es el siguiente:

.: static: corresponde al posicionamiento normal o estático. Si se utiliza este valor, se ignoran los valores de las propiedades top, right, bottom y left que veremos a continuación.

.: relative: corresponde al posicionamiento relativo. El desplazamiento de la caja se controla con las propiedades top, right, bottom y left.

.: absolute: corresponde al posicionamiento absoluto. El desplazamiento de la caja también se controla con las propiedades top, right, bottom y left, pero su interpretación es mucho más compleja, ya que el origen de coordenadas del desplazamiento depende del posicionamiento de su elemento contenedor.

.: fixed: corresponde al posicionamiento fijo. El desplazamiento se establece de la misma forma que en el posicionamiento absoluto, pero en este caso el elemento permanece inamovible en la pantalla.

La propiedad **position** sólo indica cómo se posiciona una caja, pero no la desplaza.

Normalmente, cuando se posiciona una caja también es necesario desplazarla respecto de su posición original o respecto de otro origen de coordenadas. CSS define cuatro propiedades llamadas **top, right, bottom y left** para controlar el desplazamiento de las cajas posicionadas:

.: top: Desplazamiento superior

.: right: Desplazamiento derecho

.: bottom: Desplazamiento inferior

.: left: Desplazamiento izquiero

En el caso del **posicionamiento relativo**, cada una de estas propiedades indica el desplazamiento del elemento desde la posición original de su borde superior/derecho/inferior/izquierdo. Si el **posicionamiento es absoluto**, las propiedades indican el desplazamiento del elemento respecto del borde superior/derecho/inferior/izquierdo de su primer elemento padre posicionado.

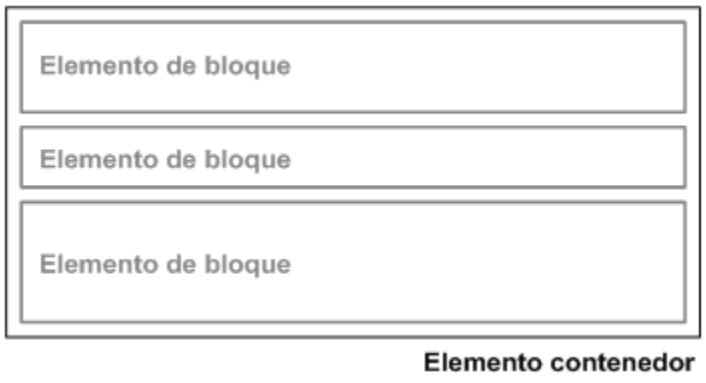
En cualquiera de los dos casos, si el desplazamiento se indica en forma de porcentaje, se refiere al porcentaje sobre la anchura (propiedades right y left) o altura (propiedades top y bottom) del elemento.

Posicionamiento estático

El **posicionamiento normal o estático** es el modelo que utilizan por defecto los navegadores para mostrar los elementos de las páginas. En este modelo, ninguna caja se desplaza respecto de su posición original, por lo que sólo se tiene en cuenta si el elemento es de bloque o en línea.

Los elementos de bloque forman lo que CSS denomina "**contextos de formato de bloque**".

En este tipo de contextos, las cajas se muestran una debajo de otra comenzando desde el principio del elemento contenedor. La distancia entre las cajas se controla mediante los márgenes verticales.

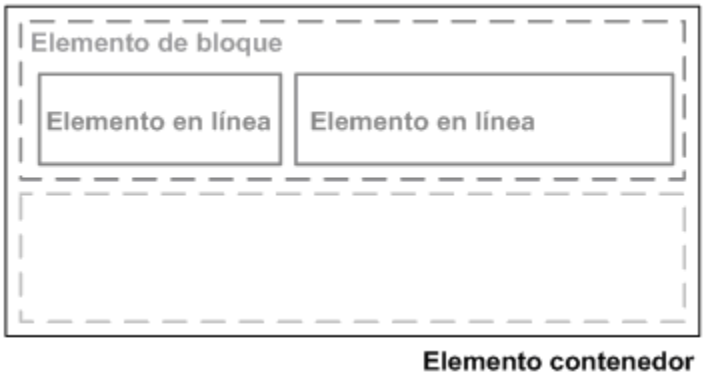


Si un elemento se encuentra dentro de otro, el elemento padre se llama "**elemento contenedor**" y determina tanto la posición como el tamaño de todas sus cajas interiores.

Si un elemento no se encuentra dentro de un elemento contenedor, entonces su elemento contenedor es el elemento **<body>** de la página.

Normalmente, la anchura de los elementos de bloque está limitada a la anchura de su elemento contenedor, aunque en algunos casos sus contenidos pueden desbordar el espacio disponible.

Los elementos en línea forman los "**contextos de formato en línea**". En este tipo de contextos, las cajas se muestran una detrás de otra de forma horizontal comenzando desde la posición más a la izquierda de su elemento contenedor. La distancia entre las cajas se controla mediante los márgenes laterales.



Si las cajas en línea ocupan más espacio del disponible en su propia línea, el resto de cajas se muestran en las líneas inferiores. Si las cajas en línea ocupan un espacio menor que su propia línea, se puede controlar la distribución de las cajas mediante la propiedad text-align para centrarlas, alinearlas a la derecha o justificarlas.

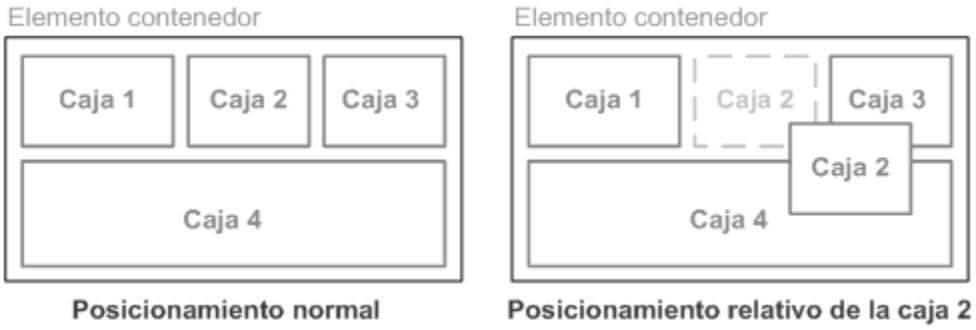
Posicionamiento relativo

El estándar CSS considera que el **posicionamiento relativo** es un caso particular del posicionamiento normal, aunque en realidad presenta muchas diferencias.

El posicionamiento relativo permite **desplazar una caja respecto de su posición original establecida mediante el posicionamiento normal**.

El desplazamiento de la caja se controla con las propiedades **top, right, bottom y left**.

El desplazamiento de una caja **no afecta al resto de cajas adyacentes**, que se muestran en la misma posición que si la caja desplazada no se hubiera movido de su posición original.



En la imagen anterior, la caja 2 se ha desplazado lateralmente hacia la derecha y verticalmente de forma descendente. Como el resto de cajas de la página **no modifican su posición**, se producen solapamientos entre los contenidos de las cajas.

La **propiedad left** desplaza la caja hacia su derecha, la **propiedad right** la desplaza hacia su izquierda, la **posición top** desplaza la caja de forma descendente y la **propiedad bottom** desplaza la caja de forma ascendente. Si se utilizan valores negativos en estas propiedades, su efecto es justamente el inverso.

Las cajas desplazadas de forma relativa no modifican su tamaño, por lo que los valores de las propiedades left y right siempre cumplen que left = -right.

Si tanto left como right tienen un valor de **auto** (que es su valor por defecto) la caja no se mueve de su posición original. Si sólo el valor de left es auto, su valor real es -right. Igualmente, si sólo el valor de right es auto, su valor real es -left.

Si tanto left como right tienen valores distintos de auto, uno de los dos valores se tiene que ignorar porque son mutuamente excluyentes. Para determinar la propiedad que se tiene en cuenta, se considera el valor de la propiedad direction.

La propiedad **direction** permite establecer la dirección del texto de un contenido. Si el valor de direction es **ltr**, el texto se muestra de izquierda a derecha, que es el método de escritura habitual en la mayoría de países. Si el valor de direction es **rtl**, el método de escritura es de derecha a izquierda, como el utilizado por los idiomas árabe y hebreo.

Si el valor de direction es ltr, y las propiedades left y right tienen valores distintos de auto, se ignora la propiedad right y sólo se tiene en cuenta el valor de la propiedad left.

De la misma forma, si el valor de direction es rtl, se ignora el valor de left y sólo se tiene en cuenta el valor de right.

Posicionamiento absoluto

El **posicionamiento absoluto** se emplea para establecer de forma precisa la posición en la que se muestra la caja de un elemento.

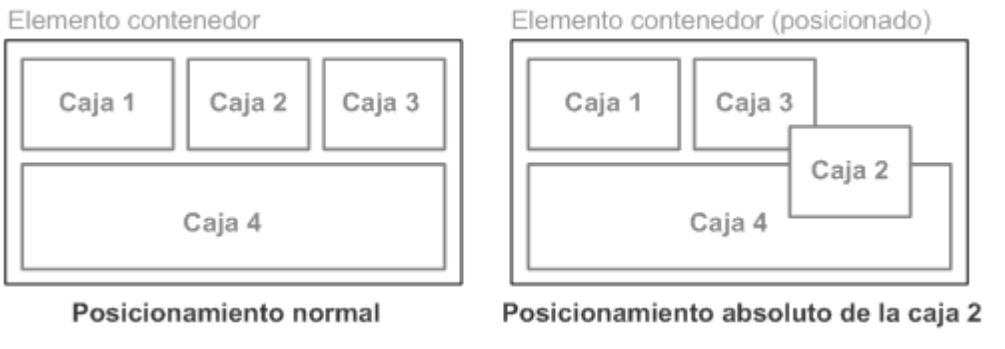
La nueva posición de la caja se indica mediante las propiedades **top, right, bottom y left.**

La interpretación de los valores de estas propiedades es mucho más compleja que en el posicionamiento relativo, ya que en este caso **dependen del posicionamiento del elemento contenedor**.

Cuando una caja se posiciona de forma absoluta, el resto de elementos de la página la ignoran y ocupan el lugar original ocupado por la caja posicionada.

Al igual que en el posicionamiento relativo, cuando se posiciona de forma absoluta una caja es probable que se produzcan solapamientos con otras cajas.

En el siguiente ejemplo, se posiciona de forma absoluta la caja 2:



La caja 2 está posicionada de forma absoluta, lo que implica que el resto de elementos ignoran que esa caja exista. Por este motivo, la caja 3 deja su lugar original y pasa a ocupar el hueco dejado por la caja 2.

En el estándar de CSS, esta característica de las cajas posicionadas de forma absoluta se explica como que **la caja sale por completo del flujo normal del documento**. De hecho, las cajas posicionadas de forma absoluta parece que estan en un nivel diferente al resto de elementos de la página.

Por otra parte, el desplazamiento de una caja posicionada de forma absoluta se indica mediante las propiedades **top, right, bottom y left**. A diferencia de posicionamiento relativo, en este caso la referencia de los valores de esas propiedades es el origen de coordenadas de su primer elemento contenedor posicionado.

Determinar el origen de coordenadas a partir del cual se desplaza una caja posicionada de forma absoluta es un proceso complejo que se compone de los siguientes pasos:

.: Se buscan todos los elementos contenedores de la caja hasta llegar al elemento **<body>** de la página.
.: Se recorren todos los elementos contenedores empezando por el más cercano a la caja y llegando hasta el **<body>**
.: De todos ellos, el navegador se queda con el primer elemento contenedor que esté posicionado de cualquier forma diferente a position: **static**
.: La esquina superior izquierda de ese elemento contenedor posicionado es el origen de coordenadas.

Una vez obtenido el origen de coordenadas, se interpretan los valores de las propiedades **top, right, bottom y left** respecto a ese origen y se desplaza la caja hasta su nueva posición.

Posicionamiento fijo

El estándar CSS considera que el **posicionamiento fijo** es un **caso particular del posicionamiento absoluto**, ya que sólo se diferencian en el comportamiento de las cajas posicionadas.

Cuando una caja se posiciona de forma fija, la forma de obtener el origen de coordenadas para interpretar su desplazamiento es idéntica al posicionamiento absoluto. De hecho, si el usuario no mueve la página HTML en la ventana del navegador, no existe ninguna diferencia entre estos dos modelos de posicionamiento.

La principal característica de una caja posicionada de forma fija es que **su posición es inamovible dentro de la ventana del navegador**. El posicionamiento fijo hace que las cajas no modifiquen su posición ni aunque el usuario suba o baje la página en la ventana de su navegador.

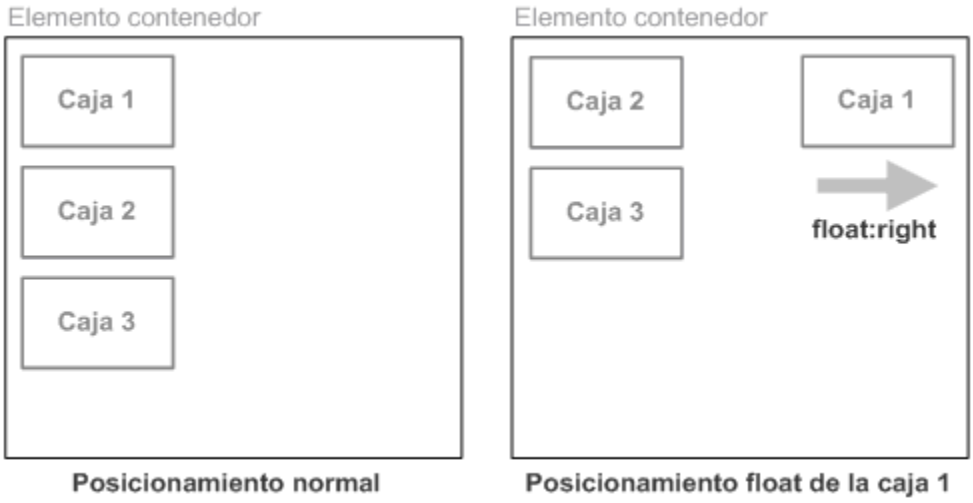
Si la página se visualiza en un medio paginado (por ejemplo en una impresora) las cajas posicionadas de forma fija se repiten en todas las páginas. Esta característica puede ser útil para crear encabezados o pies de página en páginas HTML preparadas para imprimir.

El **posicionamiento fijo** apenas se ha utilizado en el diseño de páginas web hasta hace poco tiempo porque el navegador **Internet Explorer 6** y **las versiones anteriores no lo soportan**.

ATRIBUTO FLOAT

Cuando una caja se posiciona utilizando el atributo float, automáticamente se convierte en una caja flotante, lo que significa que se desplaza hasta la zona más a la izquierda o más a la derecha de la posición en la que originalmente se encontraba.

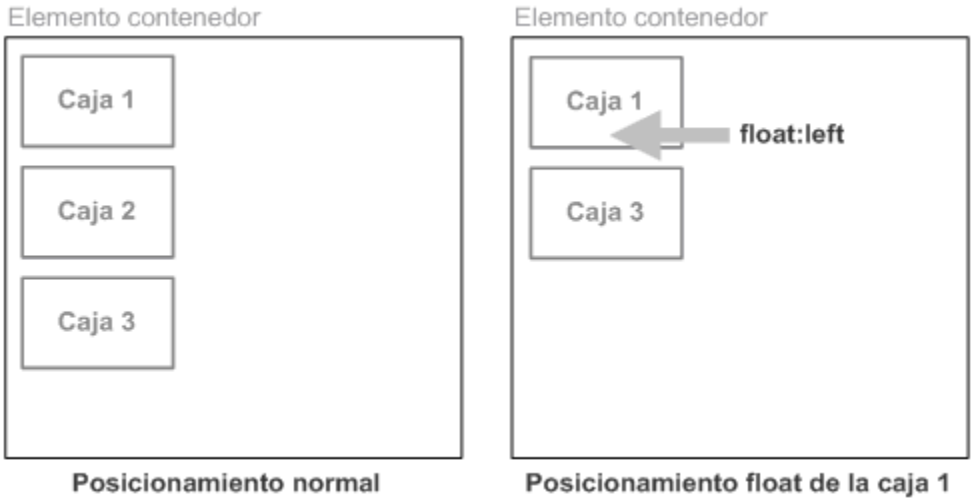
La siguiente imagen muestra el resultado de posicionar de forma flotante hacia la derecha la caja 1:



Cuando se posiciona una **caja de forma flotante**:

.: La caja **deja de pertenecer al flujo normal de la página**, lo que significa que el resto de cajas ocupan el lugar dejado por la caja flotante.
.: La caja flotante **se posiciona lo más a la izquierda o lo más a la derecha posible de la posición en la que se encontraba originalmente**.

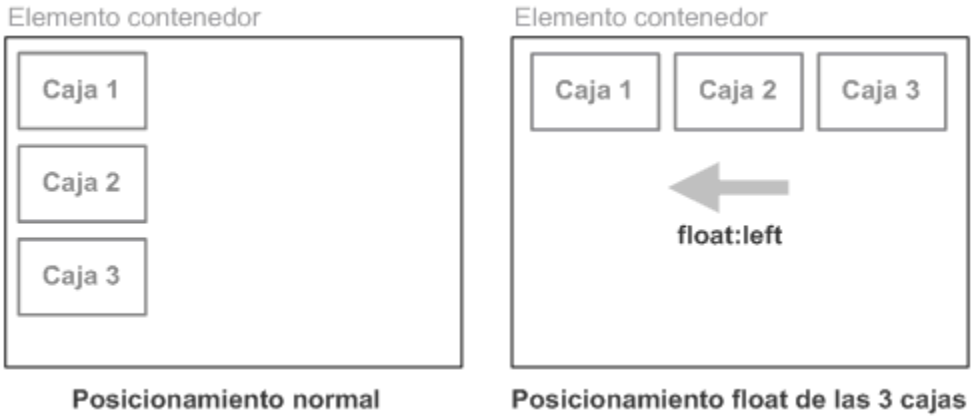
Si en el anterior ejemplo la caja 1 se posiciona de forma flotante hacia la izquierda, el resultado es el que muestra la siguiente imagen:



La caja 1 es de tipo flotante, por lo que desaparece del flujo normal de la página y el resto de cajas ocupan su lugar. El resultado es que la caja 2 ahora se muestra dónde estaba la caja 1 (en la imagen queda por debajo de la caja 1) y la caja 3 se muestra dónde estaba la caja 2.

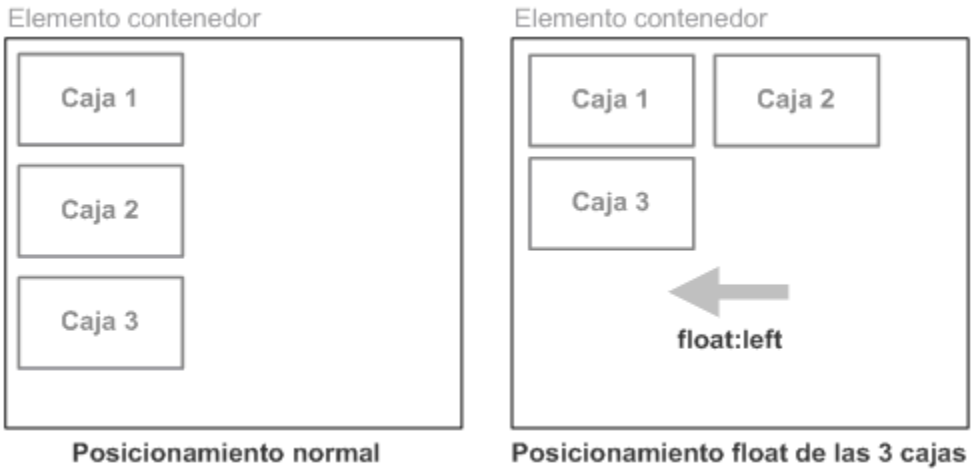
Al mismo tiempo, la caja 1 se desplaza todo lo posible hacia la izquierda de la posición en la que se encontraba. El resultado es que la caja 1 se muestra encima de la nueva posición de la caja 2 y tapa todos sus contenidos.

Si existen otras cajas flotantes, al posicionar de forma flotante otra caja, se tiene en cuenta el sitio disponible. En el siguiente ejemplo se posicionan de forma flotante hacia la izquierda las tres cajas:



En el ejemplo anterior, las cajas no se superponen entre sí porque las cajas flotantes tienen en cuenta las otras cajas flotantes existentes. Como la caja 1 ya estaba posicionada lo más a la izquierda posible, la caja 2 sólo puede colocarse al lado del borde derecho de la caja 1, que es el sitio más a la izquierda posible respecto de la zona en la que se encontraba.

Si no existe sitio en la línea actual, la caja flotante baja a la línea inferior hasta que encuentra el sitio necesario para mostrarse lo más a la izquierda o lo más a la derecha posible en esa nueva línea:



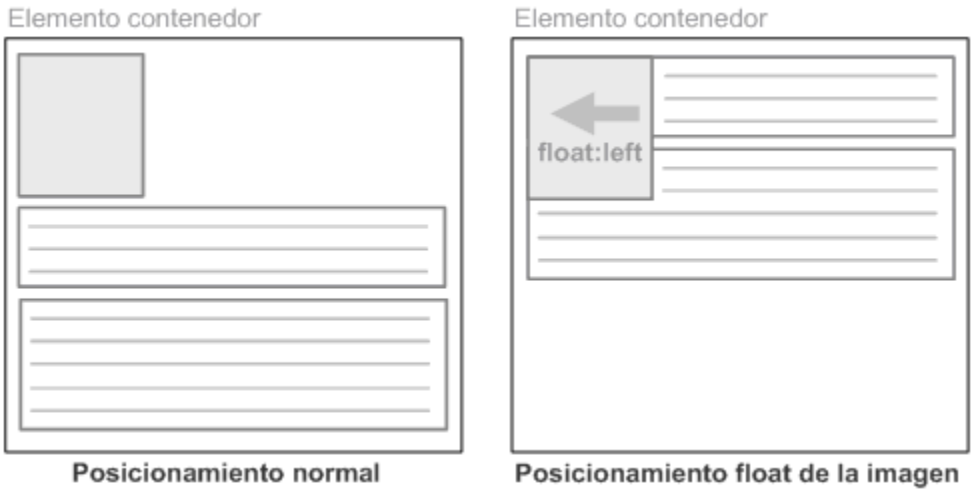
Las **cajas flotantes influyen en la disposición de todas las demás cajas**. Los **elementos en línea** hacen sitio a las cajas flotantes adaptando su anchura al espacio libre dejado por la caja desplazada. Los **elementos de bloque** no les hacen sitio, pero sí que adaptan sus contenidos para que no se solapen con las cajas flotantes.

La propiedad CSS que permite posicionar de forma flotante una caja se denomina **float**.

Si se indica un valor left, la caja se desplaza hasta el punto más a la izquierda posible en esa misma línea (si no existe sitio en esa línea, la caja baja una línea y se muestra lo más a la izquierda posible en esa nueva línea). El resto de elementos adyacentes se adaptan y fluyen alrededor de la caja flotante.

El valor right tiene un funcionamiento idéntico, salvo que en este caso, la caja se desplaza hacia la derecha. El valor none permite anular el posicionamiento flotante de forma que el elemento se muestre en su posición original.

Los elementos que se encuentran alrededor de una caja flotante adaptan sus contenidos para que fluyan alrededor del elemento posicionado:

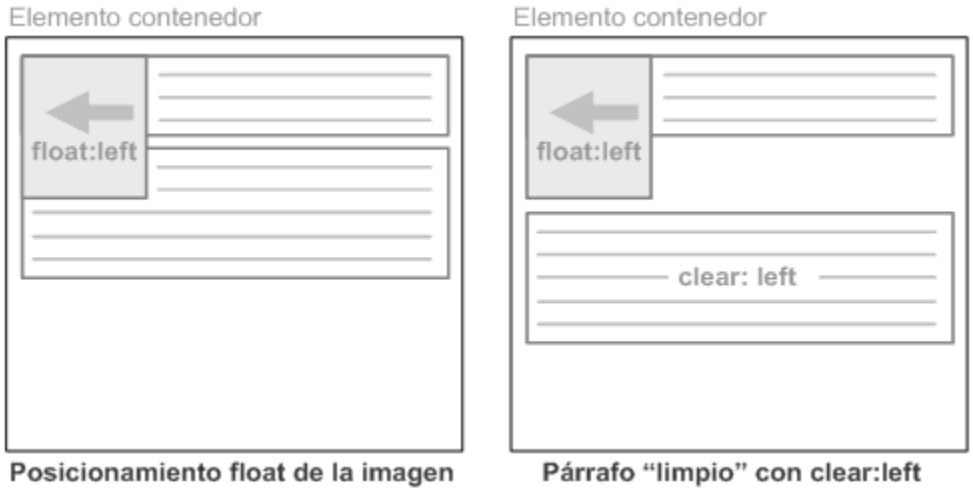


La regla CSS que se aplica en la imagen del ejemplo anterior es:

```
img {
  float: left;
}
```

Uno de los principales motivos para la creación del posicionamiento float fue precisamente la posibilidad de colocar imágenes alrededor de las cuales fluye el texto.

CSS permite controlar la forma en la que los contenidos fluyen alrededor de los contenidos posicionados mediante float. De hecho, en muchas ocasiones es admisible que algunos contenidos fluyan alrededor de una imagen, pero el resto de contenidos deben mostrarse en su totalidad sin fluir alrededor de la imagen:



Posicionamiento float de la imagen — Párrafo "limpio" con clear:left

La propiedad **clear** permite modificar el comportamiento por defecto del posicionamiento flotante para **forzar a un elemento a mostrarse debajo de cualquier caja flotante**. La regla CSS que se aplica al segundo párrafo del ejemplo anterior es la siguiente:

```
<p style="clear: left;">...</p>
```

La propiedad clear indica **el lado del elemento HTML que no debe ser adyacente a ninguna caja posicionada de forma flotante**. Si se indica el valor left, el elemento se desplaza de forma

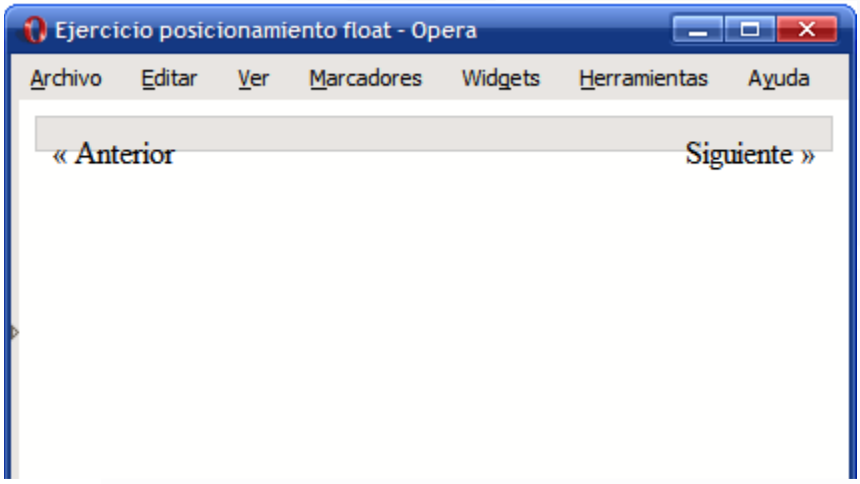
descendente hasta que pueda colocarse en una línea en la que no haya ninguna caja flotante en el lado izquierdo.

La especificación oficial de CSS explica este comportamiento como "**un desplazamiento descendente hasta que el borde superior del elemento esté por debajo del borde inferior de cualquier elemento flotante hacia la izquierda**".

Si se indica el valor right, el comportamiento es análogo, salvo que en este caso se tienen en cuenta los elementos desplazados hacia la derecha.

El valor both despeja los lados izquierdo y derecho del elemento, ya que desplaza el elemento de forma descendente hasta que el borde superior se encuentre por debajo del borde inferior de cualquier elemento flotante hacia la izquierda o hacia la derecha.

En el ejemplo, se utiliza la propiedad float para posicionar de forma flotante los dos elementos:



Como los dos elementos creados dentro del elemento se han posicionado mediante float, los dos han salido del flujo normal del documento. Así, el elemento no tiene contenidos y por eso no llega a cubrir el texto de los dos elementos :

**Los elementos posicionados vacían de contenidos al **

El añadido "limpia" el float y fuerza la altura del original

La solución consiste en añadir un elemento adicional invisible que limpie el float forzando a que el original cubra completamente los dos elementos . El código HTML y CSS final se muestra a continuación:

```
 <html>
<head><title>Ejercicio posicionamiento float</title>
<style>
#paginacion {
  border: 1px solid #CCC;
  background-color: #E0E0E0;
  padding: .5em;
}

.derecha   { float: right; }
.izquierda { float: left;  }

.borrar  { clear: both; }
</style>
</head>

<body>
```

```
<div id=”paginacion”>
  <span class=”izquierda”>&laquo; Anterior</span>
  <span class=”derecha”>Siguiente &raquo;</span>
  <div class=”borrar”> </div></div>
</body></html>
```

Al añadir un con la **propiedad clear: both**, se tiene la seguridad de que el añadido se va a mostrar debajo de cualquier elemento posicionado con float y por tanto, se asegura que el original tenga la altura necesaria como para encerrar a todos sus contenidos posicionados con float.

Además de un elemento invisible, también se puede utilizar un <p> invisible o un <hr/> invisible.

VISUALIZACIÓN

Además de las propiedades que controlan el posicionamiento de los elementos, CSS define otras cuatro propiedades para controlar su visualización: **display, visibility, overflow y z-index**.

Utilizando algunas de estas propiedades es posible **ocultar y/o hacer invisibles las cajas de los elementos**, por lo que son imprescindibles para realizar efectos avanzados y animaciones.

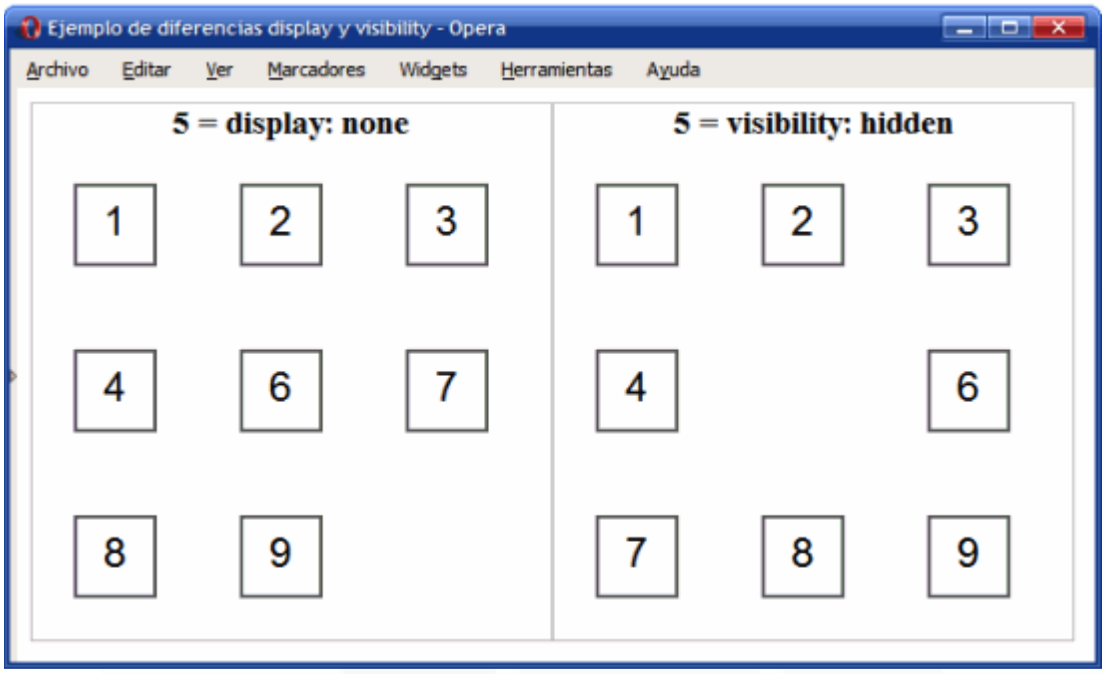
Propiedades display y visibility

Las propiedades display y visibility **controlan la visualización de los elementos**. Las dos propiedades **permiten ocultar cualquier elemento de la página**. Habitualmente se utilizan junto con **JavaScript** para crear efectos dinámicos como mostrar y ocultar determinados textos o imágenes cuando el usuario pincha sobre ellos.

La propiedad display permite **ocultar completamente un elemento haciendo que desaparezca de la página**. Como el elemento oculto no se muestra, el resto de elementos de la página se mueven para ocupar su lugar.

Por otra parte, la propiedad visibility permite **hacer invisible un elemento**, lo que significa que el navegador crea la caja del elemento pero no la muestra. En este caso, el resto de elementos de la página no modifican su posición, ya que aunque la caja no se ve, sigue ocupando sitio.

La siguiente imagen muestra la diferencia entre ocultar la caja número 5 mediante la propiedad display o hacerla invisible mediante la propiedad visibility:



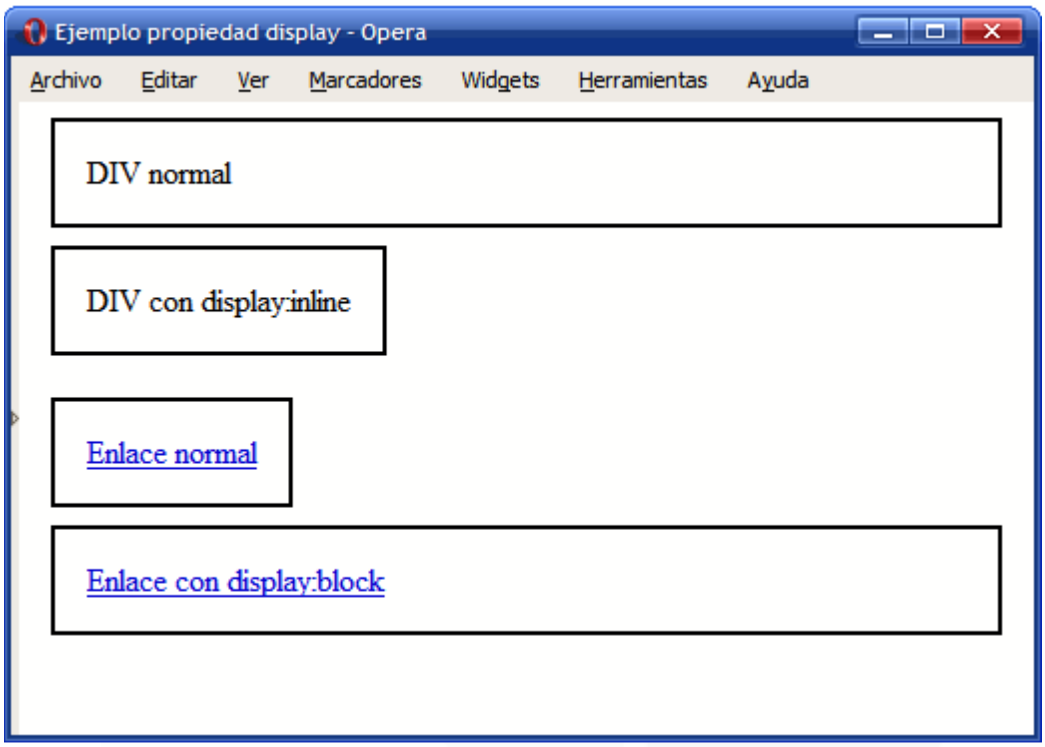
En general, cuando se oculta un elemento no es deseable que siga ocupando sitio en la página, por lo que la propiedad display se utiliza mucho más que la propiedad visibility.

Las posibilidades de la propiedad display son mucho más avanzadas que simplemente ocultar elementos. En realidad, **la propiedad display modifica la forma en la que se visualiza un elemento**.

Los valores más utilizados son **inline, block y none**. El valor **block** muestra un elemento como si fuera un elemento de bloque, independientemente del tipo de elemento que se trate. El valor **inline** visualiza un elemento en forma de elemento en línea, independientemente del tipo de elemento que se trate.

El valor **none** oculta un elemento y hace que desaparezca de la página. El resto de elementos de la página se visualizan como si no existiera el elemento oculto, es decir, pueden ocupar el espacio en el que se debería visualizar el elemento.

El siguiente ejemplo muestra el uso de la propiedad **display** para mostrar un elemento de bloque como si fuera un elemento en línea y para mostrar un elemento en línea como si fuera un elemento de bloque:



Las reglas CSS del ejemplo anterior son las siguientes:

```
<div> DIV normal</div>
<div style="display:inline;">DIV con display:inline</div>

<a href="#">Enlace normal</a>
<a href="#" style="display:block;">Enlace con display:block</a>
```

La propiedad **display: inline** se puede utilizar en las listas (<ul>, <ol>) que se quieren mostrar horizontalmente y la propiedad display: block se emplea frecuentemente para los enlaces que forman el menú de navegación.

Las posibilidades de la propiedad visibility son mucho más limitadas que las de la propiedad display, ya que sólo permite hacer visibles o invisibles a los elementos de la página.

Inicialmente todas las cajas que componen la página son visibles. Empleando el valor **hidden** es posible convertir una caja en invisible para que no muestre sus contenidos. El resto de elementos de la página se muestran como si la caja todavía fuera visible, por lo que en el lugar donde originalmente se mostraba la caja invisible, ahora se muestra un hueco vacío.

Por último, el valor **collapse** de la propiedad visibility sólo se puede utilizar en las filas, grupos de filas, columnas y grupos de columnas de una tabla. Su efecto es similar al de la propiedad display, ya que oculta completamente la fila y/o columna y se pueden mostrar otros contenidos en ese lugar. Si se utiliza el valor collapse sobre cualquier otro tipo de elemento, su efecto es idéntico al valor hidden.

Relación entre display, float y position

Cuando se establecen las propiedades **display, float y position** sobre una misma caja, su interpretación es la siguiente:

1. Si display vale none, se ignoran las propiedades float y position y la caja no se muestra en la página.
2. Si position vale absolute o fixed, la caja se posiciona de forma absoluta, se considera que float vale none y la propiedad display vale block tanto para los elementos en línea como para los elementos de bloque. La posición de la caja se determina mediante el valor de las propiedades top, right, bottom y left.
3. En cualquier otro caso, si float tiene un valor distinto de none, la caja se posiciona de forma flotante y la propiedad display vale block tanto para los elementos en línea como para los elementos de bloque.

PROPIEDAD OVERFLOW

Normalmente, los contenidos de un elemento se pueden mostrar en el espacio reservado para ese elemento. Sin embargo, en algunas ocasiones el contenido de un elemento no cabe en el espacio reservado para ese elemento y se desborda.

La situación más habitual en la que el contenido sobresale de su espacio reservado es cuando se **establece la anchura y/o altura de un elemento mediante la propiedad width y/o height.** Otra situación habitual es la de las líneas muy largas contenidas dentro de un elemento **<pre>**, que hacen que la página entera sea demasiado ancha.

CSS define la propiedad **overflow** para controlar la forma en la que se visualizan los contenidos que sobresalen de sus elementos.

Los valores de la propiedad overflow tienen el siguiente significado:

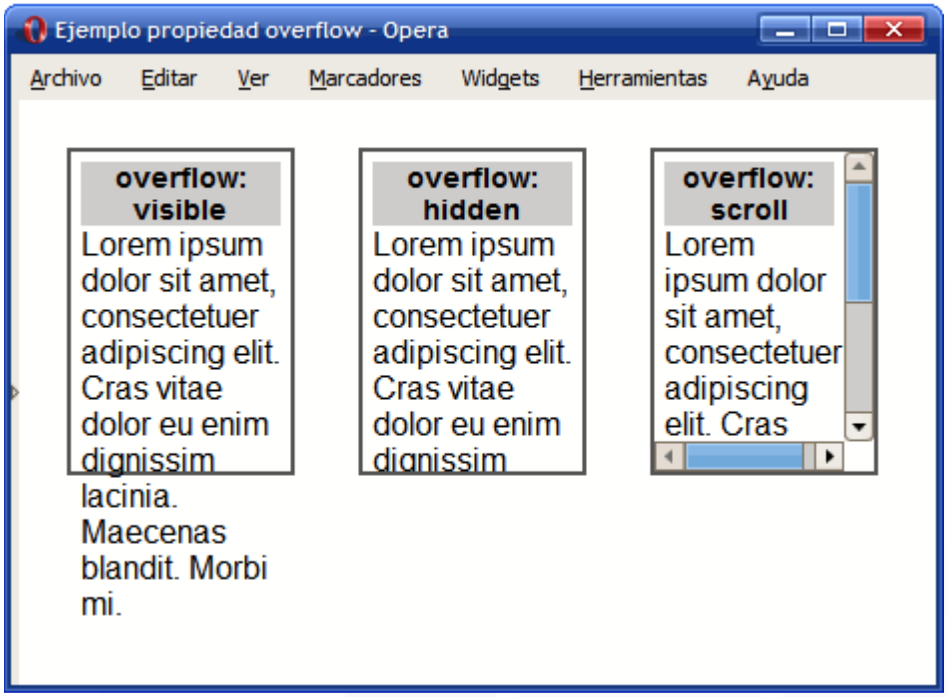
.: visible: el contenido no se corta y se muestra sobresaliendo la zona reservada para visualizar el elemento. Este es el comportamiento por defecto.

.: hidden: el contenido sobrante se oculta y sólo se visualiza la parte del contenido que cabe dentro de la zona reservada para el elemento.

.: scroll: solamente se visualiza el contenido que cabe dentro de la zona reservada para el elemento, pero también se muestran barras de scroll que permiten visualizar el resto del contenido.

.: auto: el comportamiento depende del navegador, aunque normalmente es el mismo que la propiedad scroll.

La siguiente imagen muestra un ejemplo de los **tres valores típicos** de la propiedad overflow:


Ejemplo propiedad overflow - Opera
Archivo Editar Ver Marcadores Widgets Herramientas Ayuda
overflow: visible
Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Cras vitae dolor eu enim dignissim lacinia. Maecenas blandit. Morbi mi.
overflow: hidden
Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Cras vitae dolor eu enim dignissim
overflow: scroll
Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Cras

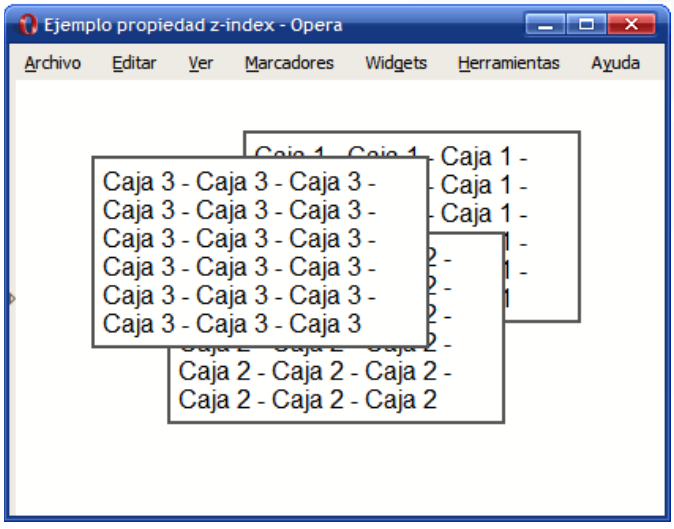
PROPIEDAD Z-INDEX

Además de posicionar una caja de forma horizontal y vertical, CSS permite controlar la **posición tridimensional** de las cajas posicionadas. De esta forma, es posible indicar las cajas que se muestran delante o detrás de otras cajas cuando se producen solapamientos.

La posición tridimensional de un elemento se establece sobre un **tercer eje llamado Z** y se controla mediante la **propiedad z-index.** Utilizando esta propiedad es posible crear páginas complejas con varios niveles o capas.

El valor más común de la propiedad z-index es un **número entero.** Aunque la especificación oficial permite los números negativos, en general se considera el número 0 como el nivel más bajo.

Cuanto más alto sea el valor numérico, más cerca del usuario se muestra la caja. Un elemento con z-index: 10 se muestra por encima de los elementos con z-index: 8 o z-index: 9, pero por debajo de elementos con z-index: 20 o z-index: 50.
La siguiente imagen muestra un ejemplo de uso de la propiedad z-index:



La **propiedad z-index** sólo tiene efecto en los elementos posicionados, por lo que es obligatorio que la propiedad z-index vaya acompañada de la propiedad **position**. Si debes posicionar un elemento pero

no quieres moverlo de su posición original ni afectar al resto de elementos de la página, puedes utilizar el posicionamiento relativo (position: relative).

HTML5

La evolución de las tecnologías y los lenguajes vinculados con internet llevan un proceso de trabajo arduo, hasta convertirse en un estándar recomendado por la entidad oficial. W3C (*World Wide Web Consortium*) es la entidad que se encarga de las especificaciones y estándares de Internet.

HTML5 nace de una necesidad de renovación de los estándares vigentes y con la promesa de potenciar y abrir nuevos caminos en el desarrollo web.

La estandarización de un lenguaje requiere su documentación sea analizada por expertos en equipos de trabajo especializados y transite diferentes etapas en los que son revisados.

La primera etapa es el **Working Draft** (borrador de trabajo), en donde se publica el primer documento de trabajo y se realizan las primeras revisiones y debates sobre él.

La segunda etapa es **Candidate Recommendation** (candidata a recomendación), en la que se verifica que el documento esté apto para su implementación.

La siguiente etapa se denomina **Proposed Recommendation** (recomendación propuesta), en donde el documente es revisado por un comité asesor.

Cuando se cumplen las 3 etapas, un documento llega a ser **W3C Recommendation** (recomendación del W3C).

HTML5 se encuentra finalizando la etapa de **Working Draft**, según lo informado por W3C se espera que llegue a ser **W3C Recommendation** para el año **2014**.

Orígenes

En el momento de tener que decidir cuál sería la evolución del lenguaje de la web, se planteó la disyuntiva entre seguir avanzando con la estándar utilizado hasta el momento y desarrollar la versión

XHTML 2.0 o bien realizar una nueva versión del lenguaje HTML puro. Finalmente esta última opción fue la que prevaleció y en 2008 tuvo su primer borrador público la 5ta versión del lenguaje HTML.

Después de más de una década de la versión 4.01, HTML5 trae la evolución que el lenguaje necesitaba para cumplir con las exigencias del desarrollo web actual.

En HTML5 se destacan sus características semánticas, las posibilidades multimedia, las nuevas funciones para formularios y las características que se definen para poder integrarse con tecnologías que permiten realizar aplicaciones para web.

FUNDAMENTOS DE HTML 5

Con esta nueva versión, HTML no cambia en lo esencial. La mayoría de los elementos permanecen. El nuevo estándar se enriquece con medios para simplificar el trabajo con las nuevas herramientas de gestión de contenidos (blogs, páginas personales, etc), y facilitar la inclusión de elementos multimedia.

El principal criterio de diseño de HTML 5 ha sido el de resolver problemas practicos, lo que hace que se hayan adoptado soluciones orientadas a facilitar el trabajo en situaciones reales.

Los siguientes son los principios de diseño que ha aplicado el WHATWG en el diseño HTML 5:

- ***Compatibilidad:*** Se garantiza la interpretación de páginas antiguas o que tengan elementos fuera del estándar.
- ***Aceptar las prácticas habituales:*** no imponer elementos que no son aceptados por la comunidad de desarrolladores.
- ***Evolución***
- ***Incluir la seguridad como parte del diseño:*** la nueva versión se ha pensado tratando de facilitar la creación de webs más seguras.

- ***Comportamiento bien definido:*** Mientras que en otras versiones se especificaba sobre todo la sintaxis, dejando libertad de implementación en los navegadores, ahora se quiere dar seguridad a los autores de la manera en la que se interpretará la información que creen.
- ***Acceso universal:*** Se puede visualizar desde cualquier dispositivo con acceso a internet.

Los navegadores disponibles hasta el momento no soportan todos los nuevos elementos de HTML 5. No obstante, y dado que los promotores del WHATWG (Apple, Mozilla, Google y Opera), el soporte a HTML 5 está aumentando rápidamente.

EL DOM

Uno de los cambios más destacados en HTML 5 sobre las versiones anteriores es la inclusión del DOM (Document Object Model) como parte del estándar. El DOM describe la estructura de un documento de acuerdo con el paradigma de la orientación a objetos y es básico a la hora de incluir código ejecutable dentro de una página para dotarla de dinamismo.

La inclusión del DOM se hace a través de sus APIs, extendiéndolas y añadiendo nuevas funciones.

¿Qué aporta HTML 5?

HTML 5 ha sido definido pensando en las personas que construyen páginas web y desarrollan aplicaciones para ellas, por lo que su orientación es ante todo práctica, con el objetivo de resolver los muchos problemas de versiones anteriores.

HTML es ahora una herramienta más útil y cercana a las personas que van a utilizarla. Además de simplificar el desarrollo, lo ha dotado de herramientas más potentes, reduciendo complejidades innecesarias. Ha eliminado componentes superfluos y que aportaban poco, y ha incorporado mecanismos ampliamente utilizados por la comunidad de desarrollo del lenguaje.

La potencial extensibilidad de HTML 5 trata de evitar que el lenguaje vaya quedando atrás cada vez más lejos de la continua evolución de la web.

HTML 5 va a suponer un marco estable para el desarrollo de páginas web, respondiendo a necesidades reales, y aportando medios para acceder a funciones potentes y sofisticadas, especialmente en lo que se refiere a los contenidos multimedia.

HTML5: LA ESTRUCTURA BÁSICA DEL DOCUMENTO

Comenzando desde las primeras líneas de código, vamos a repasar los elementos que ya conocemos y cuales se han modificado en esta nueva versión de HTML.

doctype

Para empezar cualquier página nuestra primera línea de código es el **doctype**.

Históricamente este elemento nunca ha tenido un buen acompañamiento, ya que había que indicarle algunos parámetros y versiones que normalmente nunca se ajustaban a la realidad. Ahora, con el HTML 5 esto queda reducido a una única opción muy simple:

<!DOCTYPE HTML>

Es un DOCTYPE mucho más simplificado que XHTML (cuyas reglas siguen siendo usadas) y te permite usar todas las habilidades de HTML5 sin que nada de lo que ya tienes programado deje de funcionar.

Este elemento no hay que cerrarlo ni nada, simplemente será la primera línea de cada una de nuestras páginas.

html

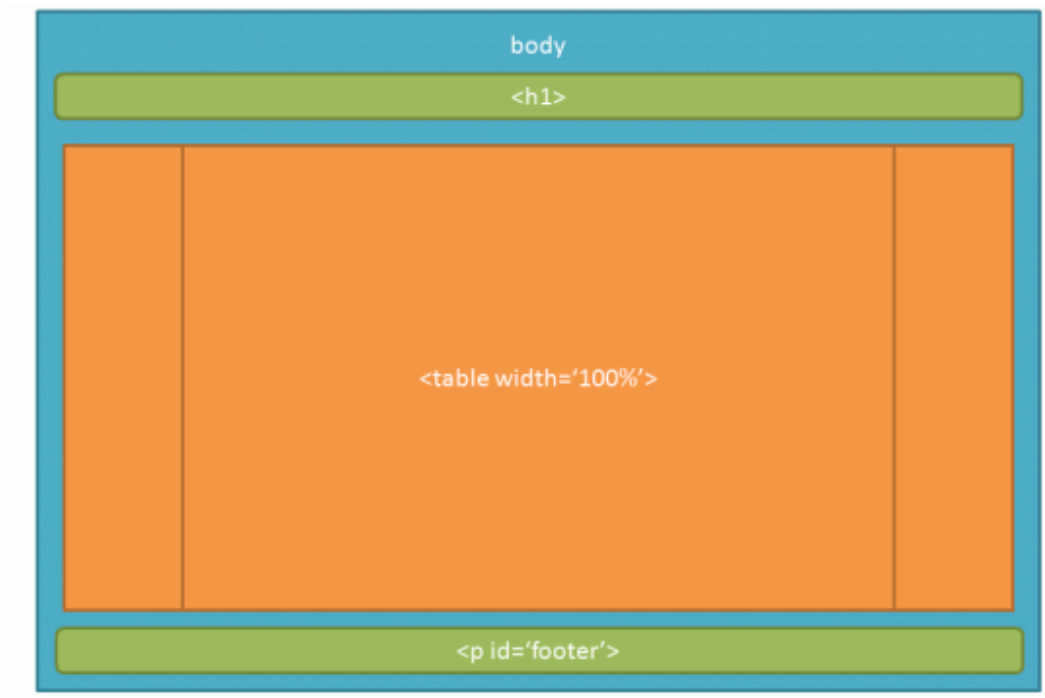
Tanto la etiqueta <html> … </html> como la etiqueta <head> … </head>se mantienen igual que en las versiones anteriores (en las próximas unidades veremos algunos atributos que podemos agregar)

Dentro de la etiqueta <body> </body>, la estructura de maquetación fue variando de acuerdo a cada versión de HTML.

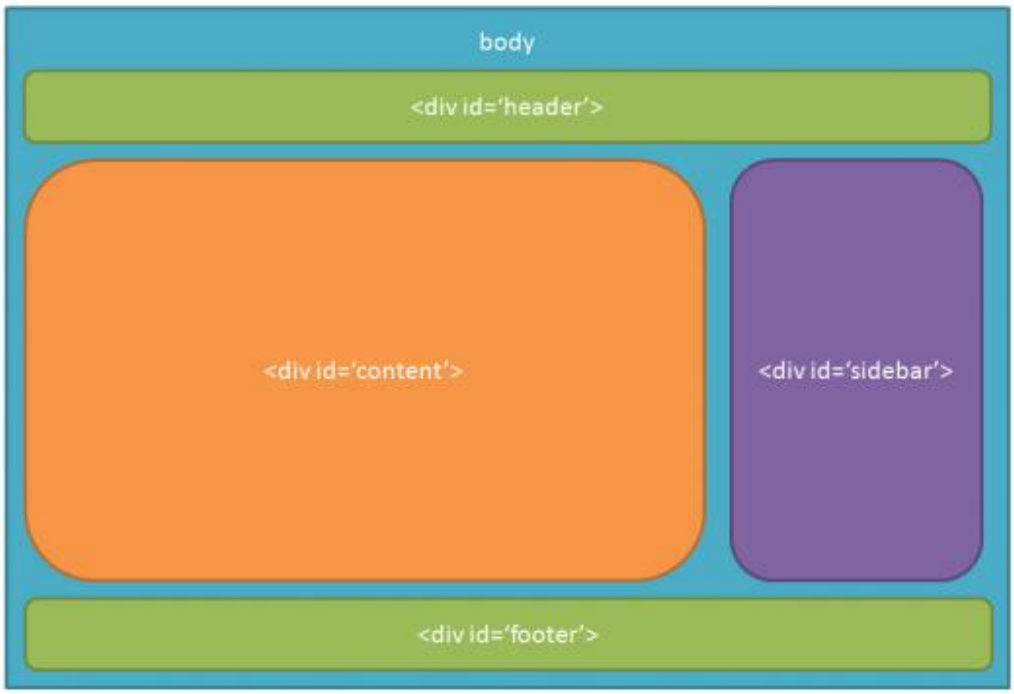
Hace unos 10 años, cuando se quería hacer una página, te podías plantear el uso de los *frames* que eran la forma más sencilla de no tener que repetir y repetir infinidad de veces el mismo código. Con la entrada de los lenguajes de programación esto comenzó a variar y se usaban *tablas* para dar formato al sitio… La aparición de los CSS hizo que los *div* se convirtieran en la mejor opción para formatear los sitios, y ahora llega el HTML 5 que incluye unas nuevas etiquetas que solucionan todos estos procesos genéricos.

Volviendo sobre las versiones anteriores de HTML podemos definir la estructura del sitio de la siguiente forma:

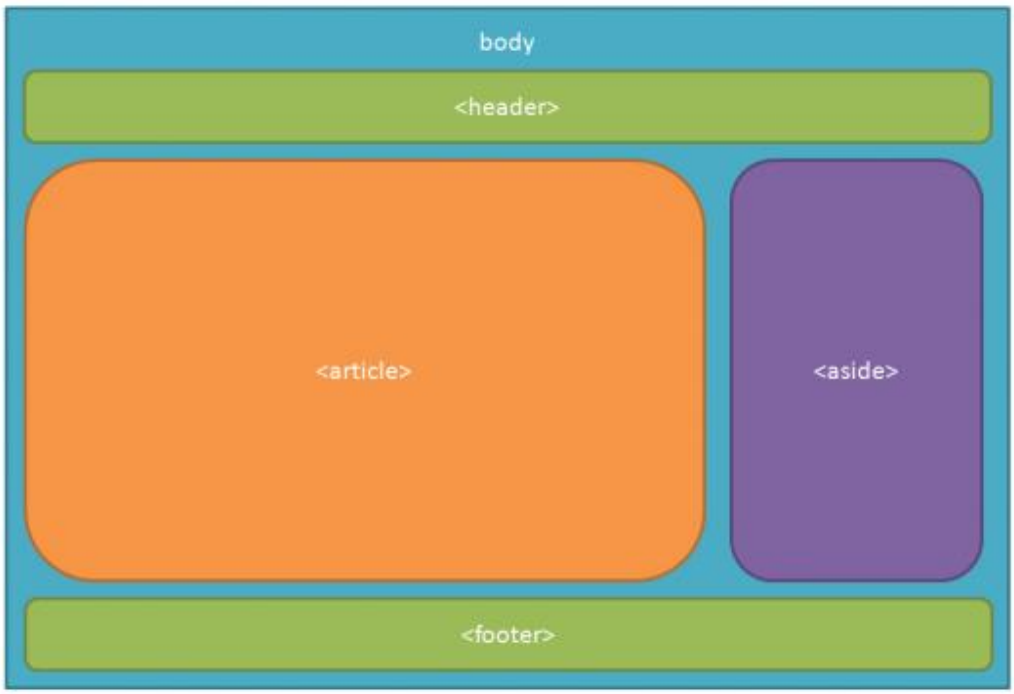
Estructura básica en HTML3:



Estructura básica en HTML4 (o en XHTML1):



Estructura básica en HTML5:



header

En principio se presenta como un contenedor que agrupa elementos introductorios o un conjunto de información referente a lo que a continuación se pondrá.

Podemos darle dos grandes usos de este elemento:

- **Cabecera de la página web**, donde se incluirían el logo, el menú de navegación, etc.
- **Encabezado de algunos bloques de información**. Por ejemplo, si tuvieramos un contenido que es la ficha de un libro, en este *header* incorporaríamos como bloque el título del libro y una pequeña ficha técnica, como encabezado del resto de información.

<header> está diseñada para reemplazar la necesidad de crear divs sin significado semántico.

footer

El elemento **footer** va a ser muy parecido al ***header*** anterior.

Básicamente se ha incluido como agrupador de elementos al final de la estructura del sitio.

Por norma general los sitios web tienen un pie de página en el que se indican datos legales y de contacto, y esta podría ser la principal función, aunque no está pensado para que sea la única.

De la misma forma que el header, hay dos funciones principales para el elemento:

- Pie de página del sitio, donde se encuentran los datos legales, etc.
- Pie de bloque de contenido, donde incorporar información como la fecha, enlaces relacionados, nube de tags y similares.

article

El nuevo elemento **article** está destinado a revolucionar los contenedores y el SEO.

Este elemento viene a ser el "contenedor" de lo que es importante en la página. Puede haber varios y es el agrupador de lo que vendría a ser "lo indexable" por un buscador, el contenido principal de la página.

Para hacernos una idea básica, una página tendría este formato:

```
<body>
  <header>...</header>
<section>
  <article>...</article>
  <article>...</article>
  <article>...</article>
</section>
  <aside>...</aside>
```

```
 <footer>...</footer>
</body>
```

Esto daría a entender que hay 3 bloques de información principales en el sitio, que son los contenidos que el usuario ha de contemplar, además de una cabecera, un menú lateral y un pie de página.

aside

Otro de los nuevos elementos es el **aside. V**iene a ser "la barra lateral" que creábamos con divs.

Cualquier contenido que no esté relacionado con el objetivo primario de la página va en un aside.

Es el contenido que, página a página dentro del sitio web se repite junto a la cabecera y el pie, y a su vez, los buscadores le darán menos peso a este bloque que al contenido dentro de **article**.

section

El nuevo elemento **section** viene a ser el ** ** que llevamos usando hasta ahora.

Define un área de contenido única dentro del sitio.

```
<section>
 <header>...</header>
 <article class="noticia">...</article>
 <article class="relacionados">...</article>
 <article class="multimedia">...</article>
</section>
```

nav

Este elemento es ***el que agrupa los enlaces***. Es un ***section*** especial para los enlaces, tanto absolutos como relativos.

Dentro de este elemento es donde incluiremos nuestra botonera.

Se puede utilizar cualquier etiqueta para organizar los enlaces, lo recomendado es utilizar listas no ordenadas (<ul>).

h1 – h6

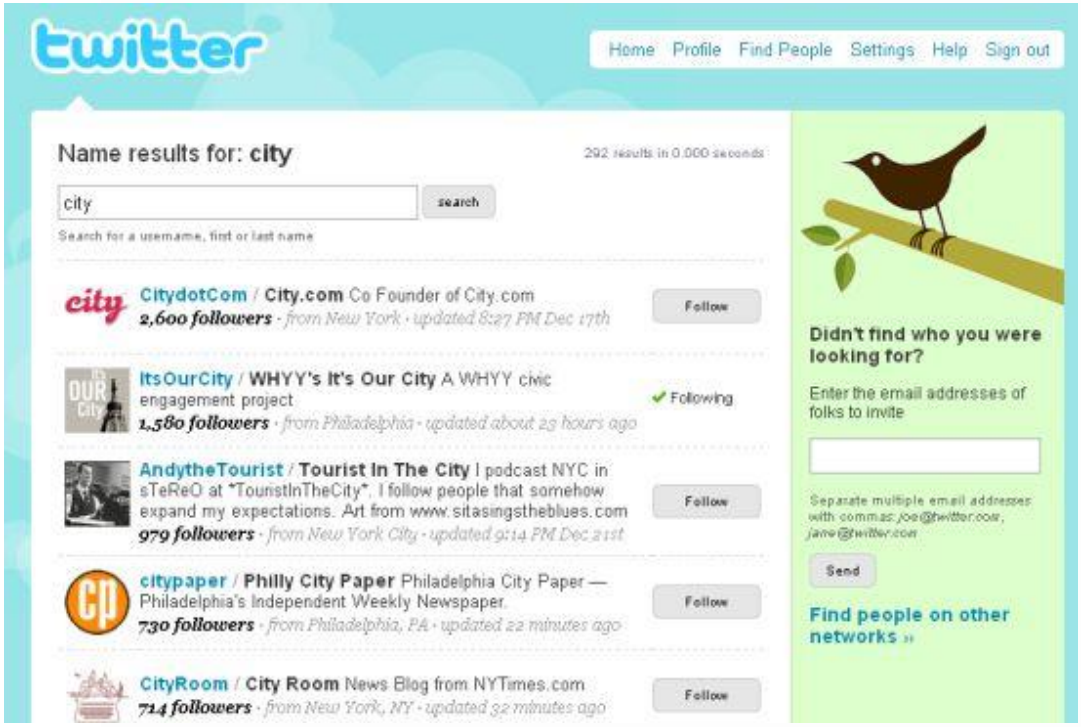
Estos elementos son los títulos de las diferentes ***sections*** y cada **h** tiene un peso (ranking específico en base al número que tiene, del 1 (el más alto) al 6 (el más bajo)).

HTML5 HOY.

Sitios de uso general que ya están utilizando HTML5.

Twitter y m.twitter.com (versión para dispositivos móviles)

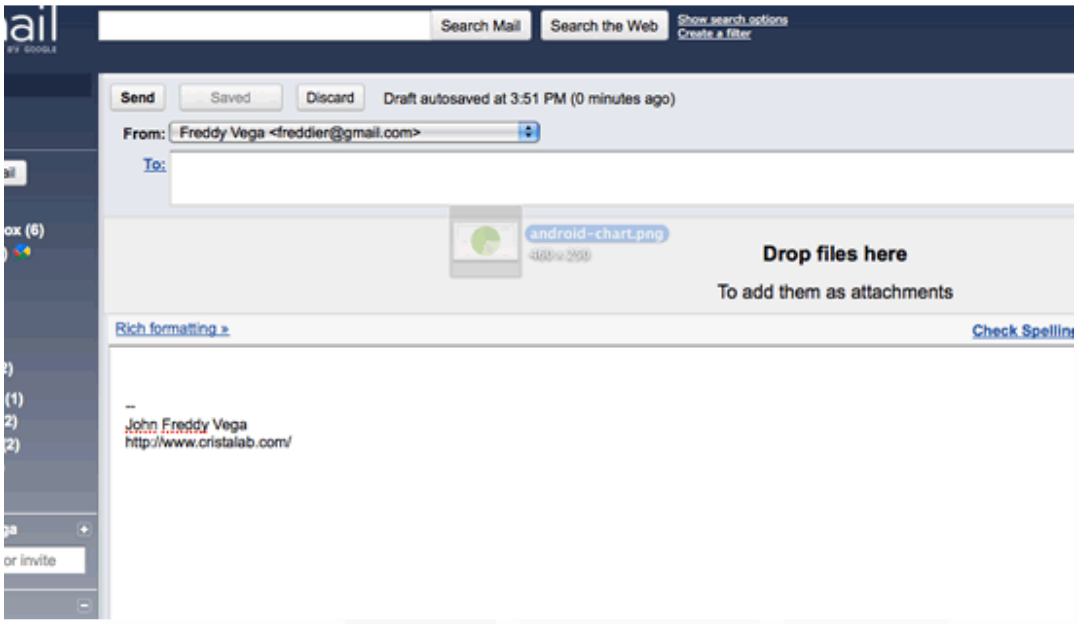
Casi todo el diseño de las versiones desktop y móvil de Twitter usan intensivamente CSS3. En especial por los bordes redondeados. En la versión móvil de Twitter se usa geolocalización sumado a Google Geolocation Services para geolocalizar los tweets.



Vimeo/m/ y m.youtube.com

Las versiones móviles de Vimeo y Youtube para teléfonos, así como sus versiones para tablets (iPad, Samsung Galaxy Tab, Playbook, etc) están hechas sólo con HTML5. CSS3 para los diseños y obviamente la etiqueta <video> para servir los videos, sin necesidad de Flash.

Gmail.com



Gmail usa Web Storage para guardar en el disco del usuario los más recientes correos. Así puedes acceder a ellos temporalmente si se cae la conexión. También usa Drag and Drop para arrastrar y soltar archivos adjuntos, entre otras habilidades de CSS3 para diseño.

Facebook

Las versiones móviles y desktop de Facebook hacen un uso intensivo de CSS3 para diseño y animaciones, así como de Web Sockets para las notificaciones de actividad y el chat.

Resumen

En esta Unidad…

En la presente unidad desarrollamos los conceptos necesarios para incorporar los atributos gráficos a nuestras estructuras de HTML utilizando el lenguaje CSS

Con las propiedades propuestas podemos comenzar a plantear la estructura gráfica de una página web.

En la próxima Unidad…

En la próxima unidad vamos a comenzar a trabajar con los nuevos elementos incorporados en la versión HTML5 para maquetar sitios web.