CURSO CSS

1 CAPÍTULO I. INTRODUCCIÓN

En este capítulo vamos a dar una visión general de lo que son y significan las CSS. A partir del próximo capítulo entraremos en profundidad en cada uno de los conceptos que se relacionan con los estilos y las CSS.

1.1 ABSTRACT

Las CSS se han convertido hoy en día en un estándar de hecho. Permiten separar el contenido de nuestras páginas web de la presentación. Dicho de otro modo: permite que el contenido de la página web se escriba en un fichero y la presentación de la página en otro fichero diferente. Esto trae innumerables ventajas para el programador y para el diseñador. Poco a poco las iremos viendo.

1.2 INTRODUCCIÓN

Cualquier empresa de gran tamaño –un banco, por ejemplo- desea que su sitio Web utilice los colores, tipo de letra, logo, etc. de la empresa. Es normal que si la empresa tiene cierta envergadura, sea relativamente elevado el número de páginas del sitio. Imaginemos que tiene, por ejemplo, 500 páginas. Si en un determinado momento se toma la decisión de cambiar el logo, los colores, etc. de la empresa, es lógico que se modifiquen también en el sitio Web.

Realmente, es frustrante tener que cambiar cada una de las 500 páginas. Probablemente, tendríamos que ir a cada una de las 500 páginas, cambiando la letra, los colores, logo, etc. Es posible que después de este trabajo hubiésemos cometido errores y nos dejemos algunos cambios sin realizar.

1.3 CONCEPTO DE ESTILO

Las CSS (Cascading Style Sheets o CSS) u Hojas de estilo en cascada –en castellano- resuelven este problema, ya que permiten **separar el contenido o estilo** (la forma en que se ve una página web) **de la presentación** en una página o en un sitio Web. Las CSS permiten mayor control a los programadores, diseñadores y usuarios de la web sobre cómo se muestran las páginas web.

Podemos definir las CSS como un lenguaje formal usado para definir la presentación de un documento estructurado escrito en HTML o XML.

Habitualmente se llama estilo de un elemento de una página web al código que define su aspecto o presentación gráfica, como puede ser el tamaño, el tipo y el color de la letra de un determinado párrafo. A veces, al código que define el estilo de un elemento se llama regla.

Las hojas de estilo en cascada son los ficheros que definen los estilos de los elementos de un sitio web.

Con las CSS los diseñadores crean hojas de estilo que definen cómo se mostrarán diversos elementos, como encabezados, enlaces, texto, imágenes, etc. Estas hojas de estilo se pueden entonces aplicar a cualquier página o a todas las páginas en un sitio web particular, lo que hace la codificación mucho más sencilla.

El término Cascading (en cascada) deriva del hecho de que múltiples hojas de estilo se pueden aplicar a una misma página web.

Por ejemplo, el elemento de HTML `<h1>` indica que un bloque de texto es un encabezamiento y que es más importante que un bloque etiquetado como `<h2>`. Versiones más antiguas de HTML permitían atributos extra dentro de la etiqueta abierta para darle formato (como el color o el tamaño de fuente). No obstante, cada etiqueta `<h1>` debía disponer de la información si se deseaba un diseño consistente para una página, y además, una persona que lea esa página con un navegador pierde totalmente el control sobre la visualización del texto.

Cuando se utiliza CSS, la etiqueta `<h1>` no debería proporcionar información sobre cómo va a ser visualizado, solamente definir la estructura del documento. La información de la presentación separada en una hoja de estilo, especifica cómo se ha de mostrar `<h1>`: color, fuente, alineación del texto, tamaño, y otras características no visuales como definir el volumen de un sintetizador de voz, por ejemplo.

1.4 VENTAJAS DE LAS CSS

Las ventajas de utilizar CSS son:

- Control centralizado de la presentación de un sitio web completo con lo que se agiliza de forma considerable la actualización del mismo. Si en un determinado momento, se desea cambiar la presentación, basta con modificar un fichero de texto de extensión `css` y los cambios se aplican de manera instantánea a nuestro sitio Web.
- Los navegadores permiten a los usuarios especificar su propia hoja de estilo local que será aplicada a un sitio web, con lo que aumenta considerablemente la accesibilidad. Por ejemplo, personas con deficiencias visuales pueden configurar su propia hoja de estilo para aumentar el tamaño del texto o remarcar más los enlaces.
- Una página puede disponer de diferentes hojas de estilo según el dispositivo que la muestre o incluso a elección del usuario. Por ejemplo, para ser impresa, mostrada en un dispositivo móvil, o ser "leída" por un sintetizador de voz.
- El documento HTML en sí mismo es más claro de entender y se consigue reducir considerablemente su tamaño.

Hay varias versiones: CSS1 y CSS2, con CSS3 en desarrollo por el World Wide Web Consortium (W3C). Los navegadores modernos implementan CSS1 bastante bien,

aunque existen pequeñas diferencias de implementación según marcas y versiones de los navegadores. CSS2, sin embargo, está solo parcialmente implementado en los más recientes.

1.5 FORMAS DE APLICAR ESTILOS

Hay tres caminos definir los estilos:

a) **Fichero externo**: Es el método más eficaz sobre todo cuando la Web es extensa. Los estilos se definen en un fichero externo y diferente a la página web. Cada página html tiene un link o referencia a este fichero por medio de la etiqueta `<link>`. Cuando se desea hacer alguna modificación en la presentación de la página basta con modificar la hoja de estilos y los cambios se realizan de manera automática en la página web. Una misma hoja de estilos servirá para muchas páginas web o para todo el sitio, por lo que los cambios son tan sencillos de realizar cuando nuestro sitio consta de una sola página que cuando contiene 500. Es en este modo de trabajo en donde se ve más claramente que la presentación y el contenido están separados.

La etiqueta a la que se ha hecho referencia tiene más o menos este “aspecto”.

```
<link rel="stylesheet" type="text/css" href="hojadeestilos.css" />
```

b) **En el propio fichero html**: Es posible definir los estilos en la propia página web utilizando la etiqueta `<style>` en el interior de la cabecera de la página, es decir, dentro de la etiqueta `<head>`. El problema de esta forma de trabajo se presenta en grandes Webs, ya que cuando se desean hacer los cambios, hay que ir también página a página. En esta forma la presentación y los contenidos están separados, sin embargo están definidos en la propia página con lo que – a mi modo de ver- no hay plena separación.

En el siguiente bloque de código, se presenta un ejemplo de cómo sería un estilo muy sencillo definido de esta manera:

```
<style type="text/css">
   .menu {
      width: 150px;
      float: left;
   }
   #mainContenido {
      margin-left: 165px;
   }
   p{
      background-color:lime;
      list-style:upper-roman;
   }
</style>
```

c) **En línea**: Es posible definir el estilo en la propia etiqueta de un elemento. El aspecto del estilo de un elemento párrafo definido de esta forma podría ser el siguiente:

```
<p style="color: #FF0000; font-family: 'Lucida Sans', sans-serif;
background-color: #FFFF00">
```

Por supuesto, no hay ningún problemas si se utilizan una, dos o las tres formas de aplicación de los estilos en un mismo sitio o página web.

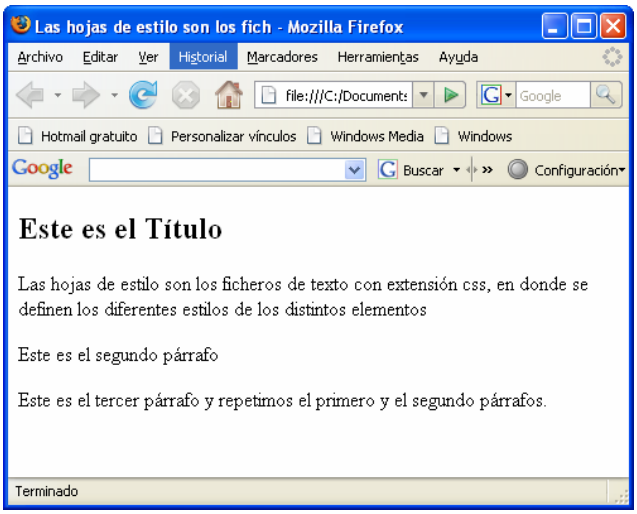
1.6 HOJAS DE ESTILO: UN POCO DE PRÁCTICA

Vamos seguir profundizando sobre cómo definir los estilos. Una hoja de estilos consiste en un fichero de texto donde se definen una serie de reglas o estilos que determinan el aspecto o presentación de cada uno de los elementos de nuestra página web. Estas reglas consisten en propiedades de fuentes, propiedades de posicionamiento, propiedades de bordes de los elementos, etc.

Por ejemplo, vamos a definir lo que es el contenido o estructura –sin definir la presentación- de una página web sencilla, con un título y tres párrafos:

```
<body>
      <h2>Este es el Título</h2>
      <p>Las hojas de estilo son los ficheros de texto con extensión
      css, en donde se definen los diferentes estilos de los distintos
      elementos</p>
      <p>Este es el segundo párrafo</p>
      <p>Este es el tercer párrafo y repetimos el primero y el segundo
      párrafos. </p>

</body>
```

Esta página se vería así:



Es interesante observar aquí –aunque más adelante lo trataremos en profundidad- que, aunque no se ha definido ninguna presentación para ningún elemento, sin embargo cada elemento de esta página web -`<body>`, `<h2>` o `<p>`- tiene un aspecto concreto. Por ejemplo, el fondo es blanco, hay un determinado margen, el tamaño de la letra del título es diferente de la letra del párrafo, etc… Los estilos permiten definir –aunque en realidad es re-definir o sobrecargar- la presentación o el estilo que cada navegador define por defecto para cada elemento web.

Ahora, vamos a definir un estilo para cada uno de los elementos. Construimos un fichero de texto separado, con extensión `css`. Para el estilo de cada elemento, utilizamos el **nombre del elemento** y **entre llaves**, y por medio de **pares propiedad-valor**, se van definiendo los valores de las diferentes propiedades que hacen relación a ese elemento, es decir:

```
Selector {
      nombreDeLaPropiedad : valorDeLaPropiedad ;
      nombreDeLaPropiedad : valorDeLaPropiedad ;
      nombreDeLaPropiedad : valorDeLaPropiedad ;
      …
}
```

Como por ejemplo

```
p {
      font-family: "Courier New";
      color: blue;
}
```

Si escribimos el fichero `default.css` siguiente:

```
body{
      background-color: #66FF33;
}
p {
      font-family: "Courier New", Courier, monospace;
      color: blue;
}
```

se está definiendo la presentación, de manera separada al contenido. Por supuesto que todas las propiedades de los elementos que no estén definidas en nuestro fichero conservarán los valores por defecto.

Con el código anterior estamos diciendo que el elemento `<body>` tenga ese color –verde claro- y que el elemento –o elementos `<p>` tengan la fuente `Courier New` con una serie de características, y de color azul.

El hecho de definir esta presentación no surte efecto hasta que “unimos” ambos ficheros –contenido y presentación: ficheros `htm` y `css`-, es decir, hasta que indicamos a la página web qué hoja de estilos define su presentación. Esto se hace por medio de una etiqueta `<link>` en la cabecera de la página donde se le indica el nombre de ese fichero de presentación u *hoja de estilos* junto con algunos atributos:

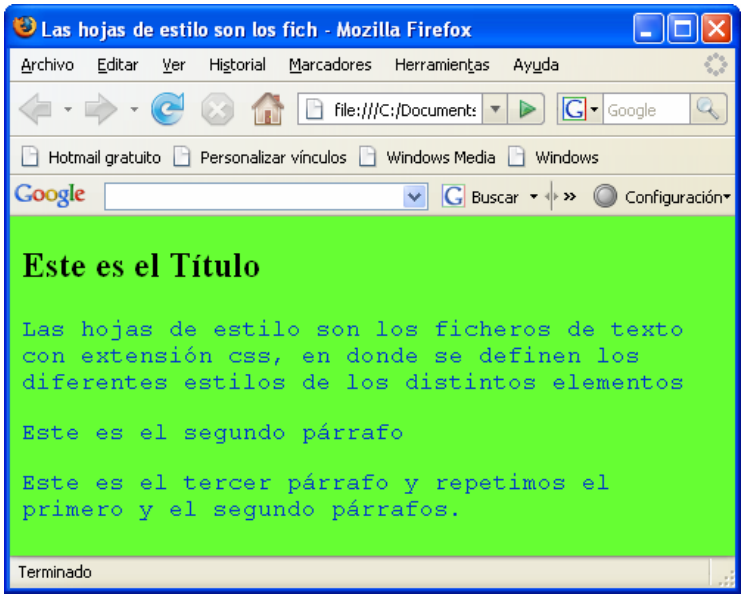
```
<link rel="stylesheet" type="text/css" href="default.css" />
```

La página web completa es:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
                "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">

<head>
       <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
       <title>Las hojas de estilo son los fich</title>
```

```
        <link rel="stylesheet" type="text/css" href="default.css" />
</head>

<body>
        <h2>Este es el Título</h2>
        <p>Las hojas de estilo son los ficheros de texto con extensión  css, en donde se
        definen los diferentes estilos de los distintos elementos</p>
        <p>Este es el segundo párrafo</p>
        <p>Este es el tercer párrafo y repetimos el primero y el segundo párrafos. </p>
</body>
</html>
```

La página anterior se vería de la siguiente manera:



Si queremos variar el color la fuente del elemento `<h2>`, bastaría con modificar la hoja de estilos `default.css` con el siguiente código:

```
body{
        background-color: #66FF33;
}
p {
        font-family: "Courier New", Courier, monospace;
        color: blue;
}
h2{
        color: red;
}
```

El aspecto de esta página web en un navegador es:

1.7 FORMAS DE DEFINICIÓN DE ESTILOS: LOS SELECTORES

Aunque trataremos los selectores en profundidad más adelante, es necesario hacer ahora una pequeña introducción.

Cuando deseamos aplicar un estilo a un determinado elemento, es necesario identificar ese elemento. Por eso, un estilo típico tiene un bloque de código donde se definen los estilos y un selector que permite identificar el estilo y aplicarlo a un elemento. Por ejemplo:

```
body{
   color: red;
   font-size: 10px;
}
p{
   background-color:lime;
   list-style:upper-roman;
}
.miClase {
   width: 150px;
   float: left;
}
#mainID {
   margin-left: 165px;
}
```

En éste ejemplo, los selectores son `body`, `p`, `.miClase` o `#mainID` y los bloques son el código que está entre llaves por cada selector. Cada línea de un bloque se estructura de la siguiente forma:

```
propiedad: valor;
```

Por ejemplo:

```
color: red
```

En un capítulo posterior trataremos los selectores con más profundidad. Aquí sólo explicaré lo necesario para poder comenzar a trabajar con estilos.

Para aplicar los estilos se utilizan tres formas diferentes, que se corresponden con los tres tipos de selectores:

a) **Clases**: Las clases permiten definir un estilo y asignarle un nombre. *El nombre de la clase se coloca como atributo en los objetos o elementos de nuestra página o páginas web en los que se desee aplicar dicho estilo.*

   Por ejemplo, si definimos el fichero `.css` siguiente:

```
h3{
  font-size: 10px;
}
.miClase{
  font-family:"Lucida Sans";
  color:red;
}
#miID {
  margin-left: 165px;
  font-family:"Courier";
  color: yellow;

}
```

Código 1

   o bien, si en el `<head>` de nuestro fichero html incluimos el bloque

```
<style>
  h3{
    font-size: 10px;
  }
  .miClase{
    font-family:"Lucida Sans";
    color:red;
  }
  #miID {
    margin-left: 165px;
    font-family:"Courier";
    color: yellow;
  }
<style>
```

Código 2

   si deseamos que un párrafo determinado tenga el tipo de letra Lucida Sans y color rojo, insertaríamos como atributo en dicho párrafo el nombre de la clase `miClase`, sin el punto inicial:

```
<p class=miClase”> Esto es un párrafo </p>
```

b) **Identificador**: *El estilo se aplica al elemento cuyo atributo `id` sea un identificador determinado*. Por ejemplo, si en un fichero incluimos el código de Código 1 o en el interior de la página html, en la cabecera, añadimos el código de Código 2, el objeto cuyo `id` sea `miID` tendrá ese margen izquierdo, y el tipo de letra será Courier y amarilla. Así, el párrafo siguiente tendrá ese "aspecto":

```
<p id=miID″> Esto es un párrafo </p>
```

c) **Redefinición de las etiquetas**: Cada elemento tiene una presentación por defecto. Por ejemplo, los elementos `<h1>` tienen un tamaño de letra diferente que los elementos `<h3>` aunque el tipo de letra por defecto sea el mismo. Para variar la presentación por defecto basta con redefinirlos bien en la hoja externa o bien en la propia página. Si incluimos como antes los códigos de las figuras 2 o 3 en el navegador, las propiedades definidas anteriormente variarán la presentación de todos los elementos `<h2>` de nuestro sitio Web o de la página respectivamente.

1.8 HERRAMIENTAS PARA TRABAJAR CON ESTILOS

Muchos diseñadores web ignoran la posibilidad de implementar el contenido y la presentación de sus sitios de manera separada. Es además muy posible que no salgan de este error hasta que deseen realizar cambios en la presentación de sus sitios.

Realmente, cuando se implementa un nuevo sitio y la herramienta que utilicemos en el diseño no nos aporte estas posibilidades, la tarea de separar los contenidos de la presentación se hace realmente complicada y se comprende bien que muchos diseñadores tiren "por la calle de en medio" para definir su presentación uniendo contenido y presentación.

En mi opinión, en este momento, Microsoft Expression Web –MEW a partir de ahora- es la mejor herramienta del mercado para trabajar con estilos o dicho de otra forma, el paquete que más ayuda al diseñador Web a separar el contenido de la presentación en sus sitios web. Sin embargo, también creo que Dreamweaver de Macromedia –ahora de Adobe- es también una magnífica herramienta. Quiero advertir que en todos los videotutoriales que aquí se muestran se ha utilizado MEW. Si no se conoce esta herramienta pero que quiere aprender, en breve aparecerán una serie de tutoriales y videotutoriales en esta misma web.

Pero para trabajar con estilos es necesario comprender bien cuando se deben aplicar los estilos en línea, en el propio documento o en una hoja de estilos externa. Además, cunado se trabaja con MEW es necesario comprender bien cuándo se aplican los estilos de manera automática o manual.

Como regla general, se puede decir que crear páginas Web utilizando los estilos en línea o en la propia página permite que la implementación o creación de las páginas sea más rápida ya que casi todo el trabajo de presentación lo puede hacer de manera automática

el programa de diseño con el que estemos trabajando y el diseñador no tiene que ocuparse de ello.

Definir los estilos en una hoja de estilos externa significa que hay que definir "manualmente" –aunque MEW nos ayuda con ventanas de diálogo donde prácticamente todo se puede hacer al golpe de ratón- dichos estilos para cada elemento de la página o sitio. Con esta forma de trabajo la creación es más lenta, pero sin embargo, como contrapartida, se gana muchísimo tiempo, se cometen muchos menos errores y, yo me atrevería a decir, que se mejora enormemente la posibilidad de mejorar la presentación del sitio o página cuando se desean realizar cambios en la presentación

En resumen: en la primera forma el trabajo de creación es más rápido pero resulta verdaderamente complejo modificar la presentación de nuestras páginas.

2 CAPITULO II: LOS FUNDAMENTOS DE LAS CSS

Quiero hacer una advertencia inicial: creo que no se puede seguir este tutorial si no se tiene unos conocimientos básicos del lenguaje XHTML o HTML.

Por otro lado, es muy importante conocer cómo un navegador va a aplicar nuestro código XHTML y los fundamentos básicos que van a forzar a que las cosas ocurran siempre de una manera determinada. Si no se conocen estos principios básicos, antes o después nos encontraremos con que no sabemos la razón por la que un navegador representa las cosas de manera diferente a la que nosotros deseábamos.

Estos conceptos fundamentales son los Bloques y los Elemento en Línea, la herencia y cascada y los box model o modelo de cajas.

Las reglas CSS las iremos explicando en el siguiente capítulo pero para establecer los fundamentos de esos capítulos es necesario explicar algo de las CSS en este momento. No se preocupe el lector si no entiende este capítulo perfectamente. Una rápida lectura al final del tutorial será suficiente para entender los conceptos aquí explicados.

Es muy posible que algunos de los ejemplos no se muestren de la misma forma en todos los navegadores porque es cierto que hay algunas diferencias entre ellos. Sin embargo, los videos ayudarán mucho para la comprensión de estas ideas. También quiero decir que profundizaré en todas estas ideas cuando ya hayamos explicado las reglas CSS en toda su extensión.

A continuación se hace referencia a dos videotutoriales de introducción a las CSS. En ellos, se hace un preámbulo a lo que significan las CSS y cómo se trabaja con ellas. Se hace un ejemplo muy sencillo pero que permite comprender el concepto de hoja de estilos.

En el video Introducción a CSS I de 9m 26seg.de duración se construye una página web y una hoja estilos con el block de notas. Se vinculan ambas y se explica cómo influyen los estilos en la presentación de la página.

En Introducción a CSS II de 10m 36seg. de duración se hace el mismo ejemplo del video anterior, pero trabajando con Microsoft Expression Web. En ejemplo, se trabaja con la herramienta escribiendo directamente código ayudándonos de las ayudas del Intelligence sense y también aprovechando las herramientas visuales que proporciona expression web.

2.1 BLOQUES y ELEMENTOS EN LÍNEA

Estoy seguro que las personas que lean estas líneas saben ya bastantes cosas del lenguaje XHTML, es muy posible que conozcan los conceptos de *bloque* y *elemento en línea*. Sin embargo, creo que es conveniente refrescar un poco algunas ideas sobre estos conceptos para clarificar algunas posibles confusiones que pueden darse posteriormente.

Para aclarar ideas:

- Identificaré el concepto de **etiqueta** con **elemento Web**. Así, `<p>`, `<h2>`, etc… son elementos Web o etiquetas.
- Llamaré **bloque** a la etiqueta más su contenido. Es decir, un bloque podría ser, por ejemplo:

```
<p> esto es un párrafo y este párrafo más las etiquetas
constituyen un bloque</p>
```

Para que nuestro código sea más claro y legible, es importante que coloquemos cada nuevo bloque en una nueva línea y cualquier contenido que les siga, también en una nueva línea.

Un bloque puede:

- Ser **simple** –como el ejemplo anterior-,
- **Contener otros bloques en su interior**. Por ejemplo:

```
<body>
<h1>Título</h1>
<p>Las manzanas caen desde las ramas de los árboles</p>
<p>El mejor jugador de fútbol ha sido Zinedine Zidane</p>
</body>
```

  En este ejemplo, `<body>` contiene tres bloques en su interior.

- **Contener elementos en línea**. Ejemplos comunes de elementos en línea son `<em>`, `<img>` y `<a>`. Los elementos en línea no pueden contener bloques en su interior y sólo pueden contener en su interior otros elementos en línea.

```
<body>
<h1>Título</h1>
<p>Las manzanas caen desde las ramas de los árboles</p>
      <p>El mejor jugador de fútbol ha sido <em>Zinedine
      Zidane</em></p>
</body>
```

  En este ejemplo, `<em>Zinedine Zidane</em>` es un elemento en línea.

2.1.1 ELEMENTOS y

Los estilos se pueden aplicar a cualquier elemento XHTML, ya sea un bloque o un elemento en línea. También es posible aplicar un estilo a un conjunto de bloques o sólo a una parte o sección de un bloque por medio de las etiquetas `<div>` o `<span>` respectivamente.

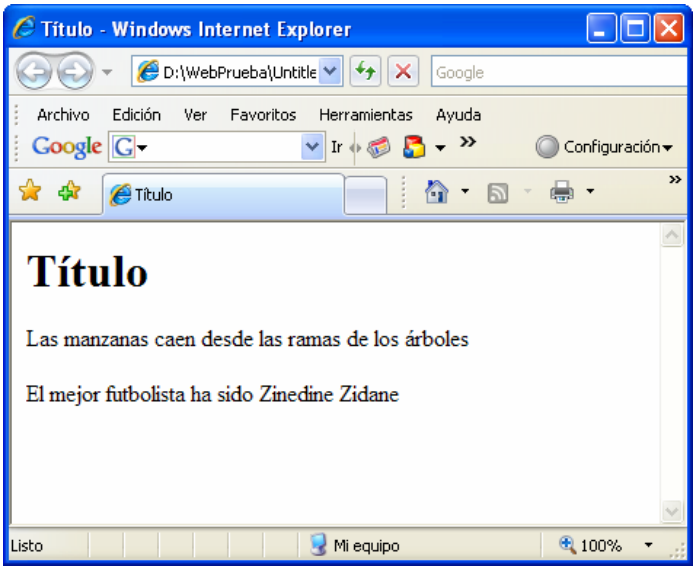
2.1.1.1 ELEMENTO

Se puede decir que el elemento `<div>` se utiliza a nivel de bloque y el elemento `<span>` a nivel de línea. Los elemento `<div>` y `<span>` pueden individualizarse con un atributo identificador o una clase –atributos `id` o `class`- .

Siguiendo con nuestro ejemplo anterior:

```
<body>
      <h1>Título</h1>
      <p>Las manzanas caen desde las ramas de los árboles</p>
      <p>El mejor jugador de fútbol ha sido Zinedine Zidane</p>
</body>
```

Aquí tenemos un bloque –el elemento `<body>`- que contiene tres bloques en su interior, que son un encabezamiento y dos párrafos. La visualización de este código es:



Cada elemento tiene una representación –un estilo- por defecto. Por eso, el tamaño del título es mayor que el de los párrafos. Por eso, el tipo de letra, su color y el del fondo, así como su tamaño y alineamiento los especifica, por defecto, cada navegador.

Para agrupar varios bloques en uno, se utiliza la etiqueta `<div>`. Voy a considerar ahora que los tres bloques de párrafo del ejemplo anterior son un solo bloque, con tres bloques en su interior. Voy a ponerle un atributo `id` para que poder dar un estilo de manera individual a este bloque. Nuestra página queda ahora de la siguiente manera:

```
<body>
      <div id="bloquePrueba">
```

```
            <h1>Título</h1>
            <p>Las manzanas caen desde las ramas de los árboles</p>
            <p>El mejor futbolista ha sido Zinedine Zidane</p>
      </div>
</body>
```

La visualización de esta página es la misma que anteriormente.

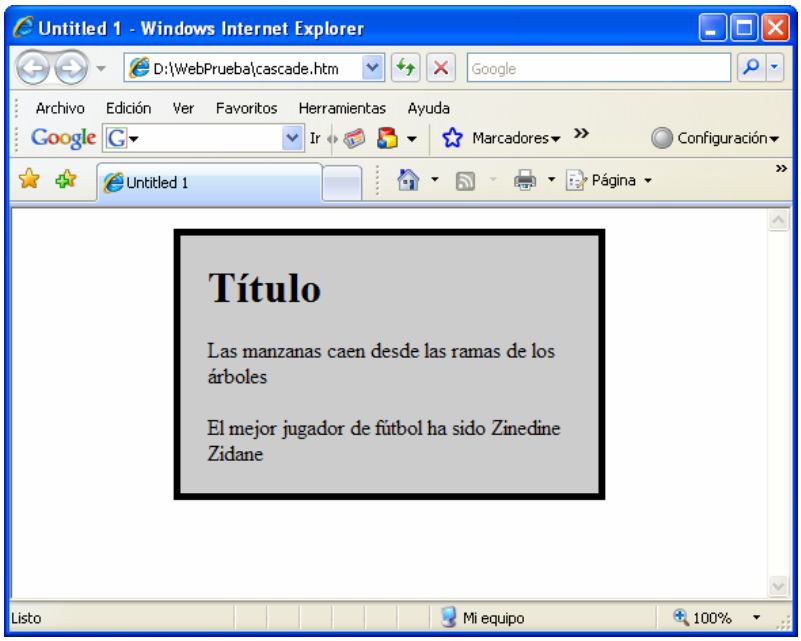
Supongamos que quiero que los vegetales del primer párrafo y al nombre del futbolista que aparece en el segundo párrafo tengan un aspecto diferente –por ejemplo, color de la letra o tamaño distinto- al resto del párrafo. Lo primero que tengo que hacer es utilizar tres elementos en línea, `<span>`, con unos atributos `class` individualizados para los vegetales y el futbolista, ya que lo lógico será aplicar a los dos animales el mismo estilo y al futbolista otro:

```
<body>
  <div id="bloquePrueba">
    <h1>Título</h1>
      <p>Las <span class=" vegetal ">manzanas </span>caen desde las
         ramas de los <span class=" vegetal ">árboles</span></p>
      <p>El mejor futbolista ha sido <span class="futbolista">Zinedine
         Zidane</span></p>
      </div>
</body>
```

La visualización sería la misma porque si no tiene ningún estilo declarado, los elementos `<div>` y `<span>` no tienen ninguna representación o estilo por defecto. Sin embargo, utilizando CSS podemos aplicar estilos al bloque `<div>` y estilos a los elemento en línea `<span>`. Por ejemplo, si definimos para bloque `<div>` –si no se especifica ningún atributo será aplicado por defecto a todos los elementos `<div>`- el siguiente estilo,

```
div {
      padding: 20px;
      margin: 0px auto;
      background: #CCCCCC;
      border: 5px solid #000000;
      width:50%;
}
```

el resultado es:



Vamos a modificar un poco el código XHTML. Voy a hacer que cada uno de los elementos esté contenido en un bloque `<div>`.

```
<body>
  <div>
    <h1>Título</h1>
  </div>
  <div>
    <p>Las <span class=" vegetal ">manzanas </span>caen desde las
ramas de los <span class=" vegetal ">árboles</span></p>
  </div>
  <div>
    <p>El mejor jugador de fútbol ha sido <span
class="futbolista">Zinedine Zidane</span></p>
  </div>
</body>
```

Ahora, a cada uno de los tres bloques se le aplicará el estilo definido. El resultado en un navegador es:

Como anteriormente hemos dicho, los bloques pueden contener a su vez otros bloques en su interior. Voy ahora a hacer un “anidamiento” de bloques en el título y aplicamos el mismo estilo:

```
<body>
 <div>
  <div>
   <div>
      <h1>Título</h1>
   </div>
  </div>
 </div>
 <div>
   <p>Las <span class=" vegetal ">manzanas </span>caen desde las ramas de los
      <span class=" vegetal ">árboles</span></p>
   <p>El mejor jugador de fútbol ha sido <span class="futbolista">Zinedine
       Zidane</span></p>
 </div>
</body>
```

El aspecto de este código es el siguiente:

Ya se ve que realmente se pueden hacer muchas cosas con un bloque `<div>`. En un capítulo posterior veremos todo lo que es posible hacer con estos elementos.

2.1.1.2 ELEMENTO

El elemento `<span>` se utiliza para aplica estilos a partes o secciones de un bloque. Voy a crear dos clases, `vegetal` y `futbolista`, que me permitan distinguir "visualmente" en mi página Web esos elementos del resto del texto. Para esto, volvemos a nuestra página inicial:

```
<body>
  <div id="bloquePrueba">
    <h1>Título</h1>
      <p>Las <span class="vegetal">manzanas </span>caen desde las ramas de
          los <span class="vegetal">árboles</span></p>
        <p>El mejor futbolista ha sido <span class="futbolista">Zinedine
          Zidane</span></p>
      </div>
</body>
```

Su aspecto era el de la figura:

Ahora especificamos un estilo diferente para las secciones de cada bloque que queremos destacar. En nuestro caso, destacaremos los vegetales y los futbolistas. Para eso, creamos una clase para cada uno de estos elementos. La hoja de estilos es:

```
div {
	padding: 20px;
	margin: 0px auto;
	background: #CCCCCC;
	border: 5px solid #000000;
	width:50%;
}
.planta {
	font-variant: small-caps;
	color: #0000FF;
}
.futbolista {
	font-variant: small-caps;
	text-decoration: underline;
	color: #FF3300;
}
```

El aspecto de mi página Web es la siguiente:

Ahora se ve claramente que los elementos `<span>` nos permiten aplicar un estilo diferente al estilo en el que esté incluido el elemento.

2.2 HERENCIA Y CASCADA

Es perfectamente posible e incluso deseable que un determinado elemento tenga más de un estilo aplicado. La herencia y la cascada describen cómo se aplican varios estilos a un elemento sin que haya conflictos.

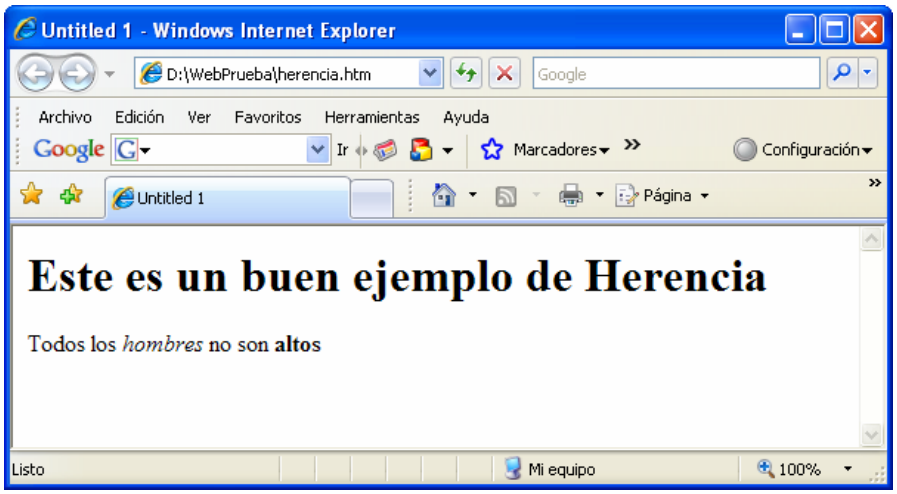
2.2.1 HERENCIA

La herencia trabaja en la forma en que se espera o a la que estamos acostumbrados a trabajar con código XHTML o HTML. Por ejemplo:

Tengo la siguiente página web:

```
<body>
      <h1>Este es un buen ejemplo de Herencia </h1>
      <p>Todos los <em>hombres</em> no son <strong>altos</strong></p>
</body>
```

La apariencia de esta página es la siguiente:



En términos de herencia, en el código anterior, se suele decir que el elemento `<body>` **es padre** de `<h1>`, `<p>`, `<em>` y `<strong>`. Dicho de otra forma: `<body>`, como bloque, contiene a los elementos `<h1>`, `<p>`, `<em>` y `<strong>`. Así mismo, se dice que `<h1>`, `<p>`, `<em>` y `<strong>` **son hijos** de `<body>`.

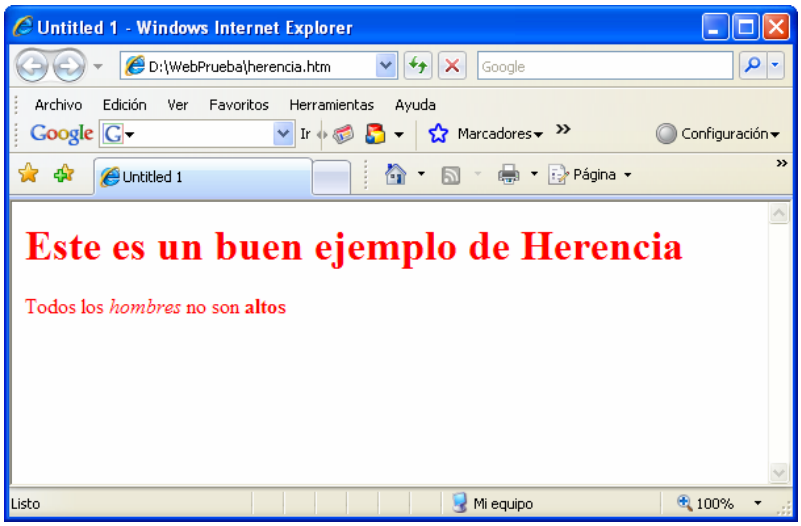
También se puede decir y considerar –pensando en padre, abuelo,…- que `<body>` es padre de `<h1>` y `<p>` mientras que a su vez `<p>` es padre de `<em>` y `<strong>`. Es posible que un padre tenga más de un hijo y sin embargo un hijo no puede tener más que un solo padre. Los hijos del mismo padre se denominan **hermanos**.

Las reglas que se aplican en la herencia son las siguientes:

Los estilos se heredan de padres a hijos. Por eso, todos los hijos utilizarán la misma fuente y el mismo color de letra que el padre a menos que se *sobrescriba* el estilo del hijo añadiéndole un nuevo estilo. Por ejemplo: si se declara un texto rojo para `<body>` , todo el texto que se contenga en `<body>` será rojo, incluyendo aquí a todos los hijos. Si en el ejemplo anterior lo hacemos, es decir si consideramos una hoja de estilos con el texto:

```
body {
      color: #FF0000;
}
```

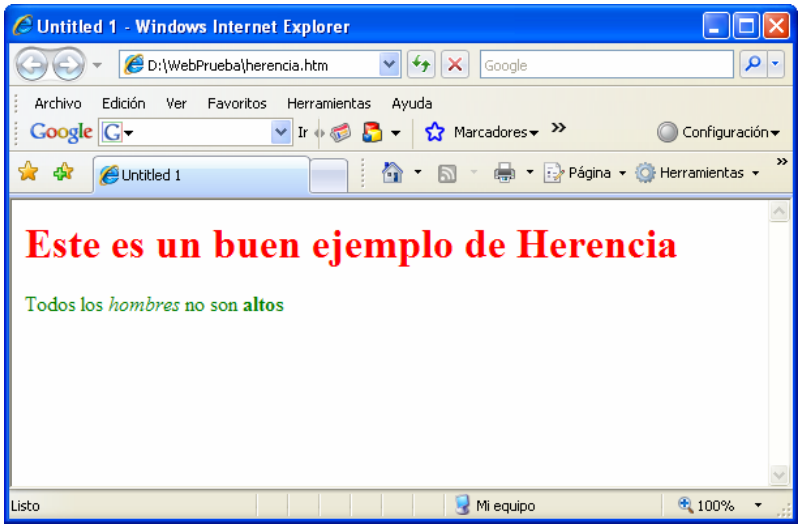
Lo que visualizaríamos es lo siguiente:



Si ahora aplicamos un estilo para el párrafo donde el color sea verde –es decir, si sobrescribimos el color de la fuente-,

```
body {
	color: red;
}
p{
	color:green;
}
```

entonces todo el texto contenido en el bloque <p> será verde, incluyendo los hijos de <p>. El texto del elemento <h1> permanecerá rojo:

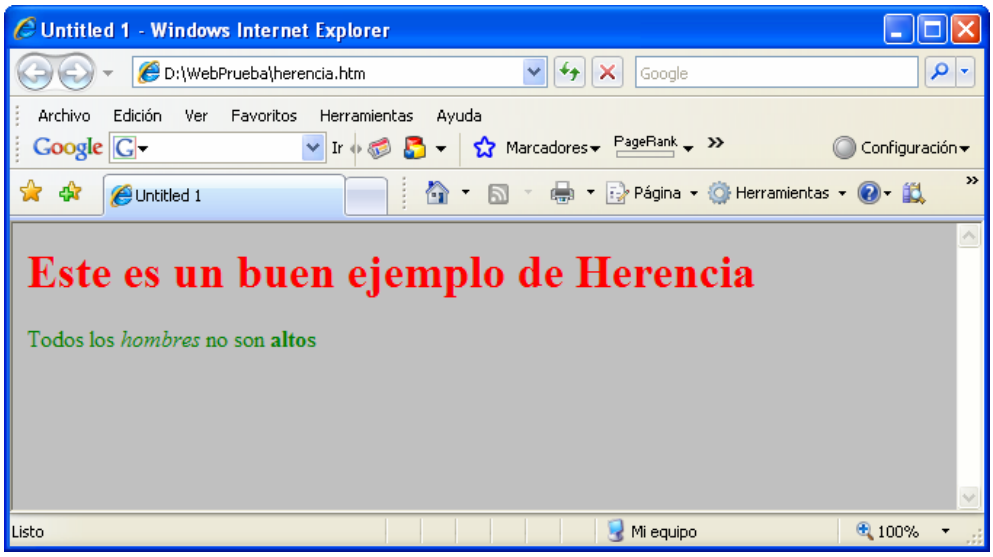


Lo mismo pasa con las demás propiedades: Por ejemplo, si definimos también un color de fondo para el <body>:

```
body {
	color: red;
	background:silver;
}
```

```
p{
	color:green;
}
```

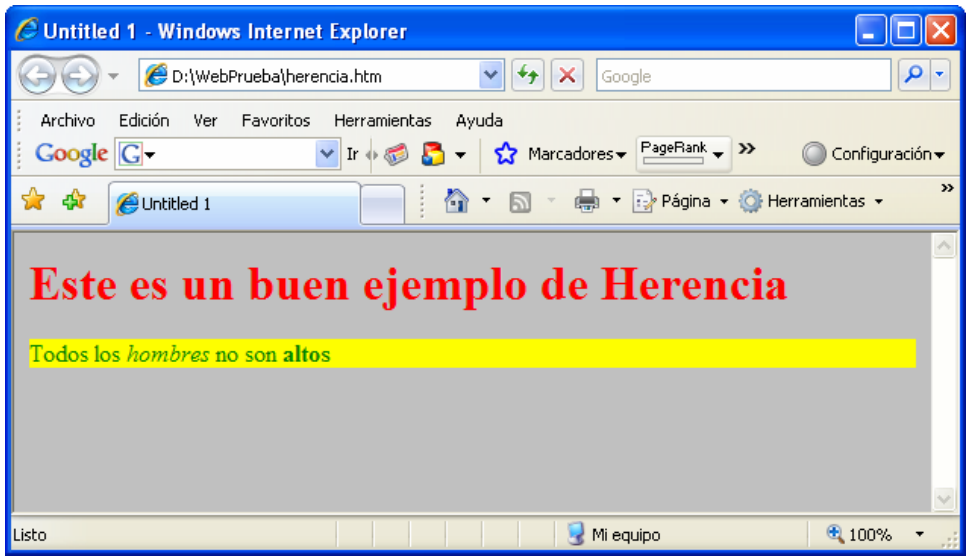
El resultado es el siguiente:



Aquí se ve que el fondo es el mismo para el padre que para los hijos porque ningún hijo ha definido –ha sobrescrito- esa propiedad CSS. Sin embargo, si ahora en el estilo del párrafo se define:

```
body {
	color: red;
	background:silver;
}
p{
	color:green;
	background:yellow;
}
```

El resultado es:

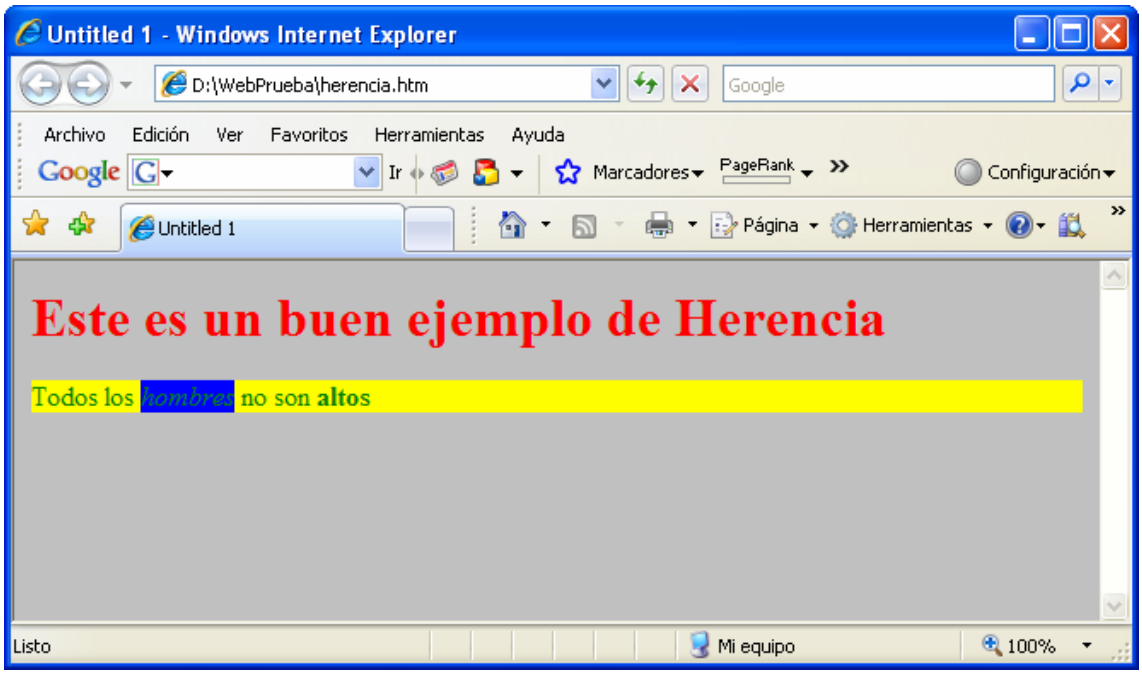


Es decir, que **los hijos heredan el estilo de los padres excepto cuando los hijos tengan un estilo definido con esa propiedad sobrescrita, es decir, puesta a otro valor**.

Vamos a llevar esto a su extremo: el elemento ha heredado el color del fondo de <p>. Vamos a ponerle otro color de fondo:

```
body {
      color: red;
      background:silver;
}
p{
      color:green;
      background:yellow;
}
em{
      background:blue;
}
```

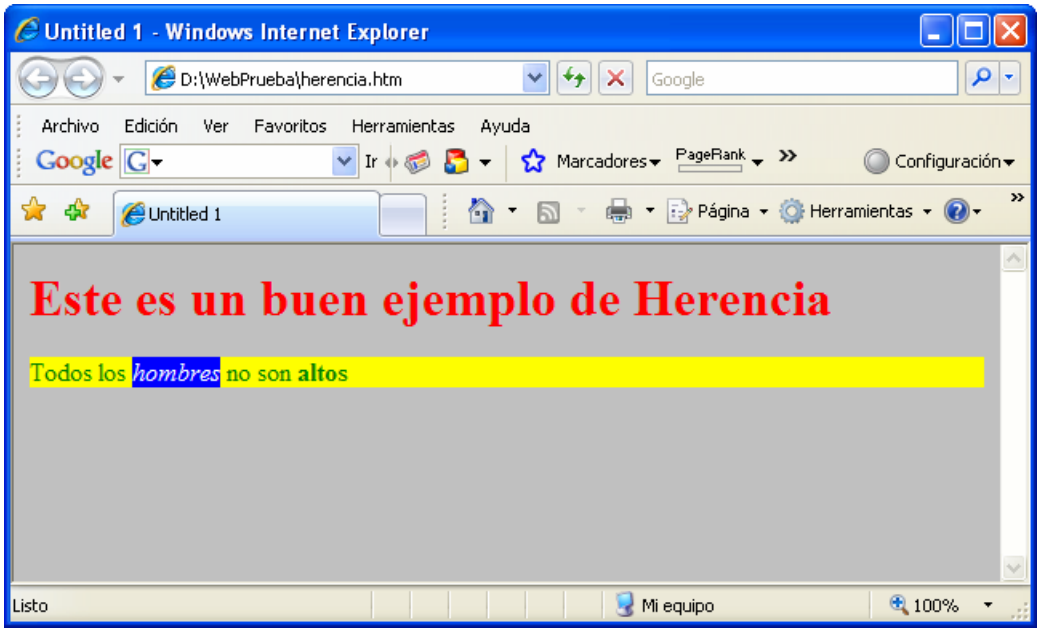
Lo veríamos de la siguiente forma:



Si fuese:

```
body {
	color: red;
	background: silver;
}
p{
	color:green;
	background:yellow;
}
em{
	background:blue;
	color:white;
}
```

Lo veríamos así:



Aquí vemos que el elemento `<strong>` es el único que hereda del padre `<p>` porque no tiene definidos el color y el fondo.

En resumen: los hijos heredan los estilos de los padres excepto aquéllas propiedades que tengan sobrescritas o redefinidas.

En el videotutorial Herencia en CSS de 7m 14seg. de duración se explica con detenimiento el mecanismo de la herencia en CSS. Cómo los hijos heredan los estilos de los padres y cómo estos lo transmiten a sus hijos. Pero si un hijo sobrescribe un estilo, el estilo que predomina es el que ha definido el hijo, es decir el del propio elemento. Todo esto se explica con un ejemplo.

2.2.2 CASCADA

Creo que la palabra inglesa *Cascade* es difícil de traducir al castellano.

Se fundamenta en el hecho de que el estilo que se aplica a un elemento es el resultado de la aplicación de varios estilos diferentes.

Esto se da a dos niveles diferentes:

- Una página puede tener más de una hoja de estilos. Por ejemplo, cuando se desea definir los estilos para una gran empresa y hay estilos diferentes para las distintas secciones de la empresa.
- Por otro lado:
    - El navegador proporciona una hoja de estilos por defecto,
    - El usuario puede proporcionar otra.
    - El diseñador de la página Web puede proporcionar una tercera.

Con la palabra *cascada* se quiere representar *precedencia* o *jerarquía* a la hora de aplicar los diferentes estilos, cuando hay más de una fuente donde se definen los estilos.

a) Concepto de Cascada en los estilos definidos por el diseñador, el usuario y el navegador.

- Estilos definidos por el navegador.

Aunque no seamos conscientes, cualquier persona que haya escrito una página Web está familiarizado con la hoja de estilos que tiene definida el navegador: son los estilos que se aplican por defecto a los diferentes elementos en una página. Así, cualquier página XHTML a la que no se haya aplicado ningún estilo, cuando se carga en un navegador se aplica una hoja de estilos externa: la que tiene definida el navegador. Por eso, cuando no se define ningún estilo porque el código XHTML es puro o limpio, los párrafos tienen un tipo de letra que es Times New Roman, tiene un tamaño determinado, el color de la letra es negro y el fondo blanco. Si se trata de un encabezamiento, la letra puede ser de diferente tipo, el tamaño también, normalmente se escribe en negrita, a lo mejor es mayúscula y dependiendo del tipo de encabezamiento puede estar en cursiva o no.

Es posible que los diseñadores de páginas Web hayan podido comprobar cómo algunos elementos se visualizan diferentes cuando los navegadores son distintos. La razón es que cada navegador aplica, por defecto, una hoja de estilos peculiar. Por eso, aunque muchas veces no se hace, es aconsejable definir todas las propiedades que sospechemos que pueden variar de un navegador a otro. Así por ejemplo, si deseamos una página con fondo blanco: podemos pensar que por defecto todos los navegadores, por defecto, van a representar la página con este color de fondo. Podría suceder que un determinado navegador representara el fondo de la página de un color gris claro y el resto de color blanco. En esos casos, es mejor no dejar al navegador que decida por nosotros y definamos el color blanco como color de fondo.

- Estilos definidos por el usuario.

Los usuarios pueden definir también sus propios estilos. Esto tiene mucho sentido: por ejemplo, cuando hay defectos de visión. Algunas páginas permiten que los usuarios modifiquen el tamaño de la letra o el contraste.

- Estilos definidos por el diseñador.

El diseñador puede y, en general crea, estilos.

Por ejemplo, puede definir un estilo para toda la página Web –para el elemento `<body>`- para que la letra sea de tipo Lucida, un poco más pequeña y además de color rojo. Pero entonces podríamos preguntar: ¿quién tiene ahora preferencia, los estilos que el navegador tiene por defecto o los del diseñador de la página Web?

Este es precisamente el problema que se plantea porque vemos que en cualquier página Web hay múltiples hojas de estilo para aplicar: los del diseñador, los del usuario y los del navegador. Es muy importante que los diseñadores de páginas Web conozcan los mecanismos de aplicación de los estilos porque si no es seguro que antes o después se encontrarán con problemas.

El concepto de Cascada, resuelve este caso: **el navegador da un peso "descendente" a las hojas de estilo en este orden: diseñador, usuario, navegador**. Es verdad que hay excepciones pero en general se puede afirmar lo anterior. De esta forma, se puede decir que el estilo definido por el diseñador es el que prevalece en caso de conflicto y sobrescribe el estilo definido por el usuario y éste a su vez, sobrescribe el estilo proporcionado por el navegador.

Vamos a "pedalear" un poco sobre esta cuestión, porque me parece importante.

Como hemos dicho, el usuario puede definir sus propios estilos. Pero el peso relativo de las hojas de estilo hace que los estilos definidos por el diseñador sean los que tienen preferencia sobre los demás. De esta forma, cuando haya conflictos –es decir cuando se hayan definido una misma propiedad a niveles distintos- se aplica el estilo que tenga más peso, por este orden: diseñador, usuario, navegador. Es decir, los estilos del diseñador tienen preferencia siempre sobre los demás, y los estilos del usuario tienen preferencia sobre los estilos del navegador.

Sin embargo, un usuario tiene la opción de utilizar la declaración `¡important` en un determinado estilo **para que la precedencia cambie**, es decir, **para que los estilos definidos por el usuario tengan precedencia sobre los definidos por el diseñador**. La verdad es que hay buenas razones para esto: puede ser que los estilos del usuario no sean estéticos pero puede ser que un usuario necesite cambios para acceder a una página, por ejemplo, porque necesita más contraste o un tamaño de letra más grande por dificultades de visión. La sintaxis para lograr esto es muy simple.

Por ejemplo, para cambiar la precedencia, es necesario cambiar el código:

```
body{

font-size:25px;
}
```

por el código:

```
body{

font-size:25px ¡important;
}
```

Es importante saber que la declaración `¡important` sólo se aplica a la propiedad que está en la misma línea. Por ejemplo:

```
body{
      background:blue;
      font-size:25px !important;
      border:20px;
      border-width:20px;
      color:red;
}
```

En este caso, sólo se aplica la "precedencia" del estilo del usuario al del diseñador en el tamaño de la fuente del elemento `<body>`, no en los demás casos. Realmente, se puede pensar del siguinte modo: **el uso de** `¡important` **sobrescribe el estilo definido por el diseñador con el estilo del usuario**.

b) Concepto de Cascada cuando hay varias hojas de estilo definidas por el diseñador.

La cascada también proporciona un criterio para conocer la precedencia cuando el diseñador de la página proporciona más de una hoja de estilos. Por ejemplo: imaginemos un sitio Web de una gran empresa. Es muy posible que el diseño necesite diferentes conjuntos de estilo según se trate de diferentes secciones de la empresa. Para esto se suele trabajar con varias hojas de estilo. Se suele crear una hoja de estilos global, que defina todos los estilos comunes a todos los documentos y otras hojas de estilo que definan los estilos de las secciones –quizá es buena idea crear una hoja de estilos por sección-. Por ejemplo: un periódico podría tener varias secciones y definir una hoja de estilos para cada una de ellas,

- `marketing.css`
- `ultimahora.css`
- `empresa.css`
- `deportes.css`

Probablemente, `empresa.css` debería definir los estilos comunes a todos el sitio para que, por ejemplo se pueda identificar la imagen corporativa de la empresa por medio de tipo y color de la letra, colores corporativos, etc. Los documentos que tengan que ver con el departamento de marketing deberían referenciar la

hoja de estilos `marketing.css` y lo mismo los departamentos de última hora y de deportes.

Como regla general, se puede decir que **cuando hay varias hojas de estilo definidas el navegador aplica la hoja de estilos por el orden en el cual se listan en la cabecera del documento XHTML**. Imaginemos, por ejemplo, que la cabecera de nuestro documento es la siguiente:

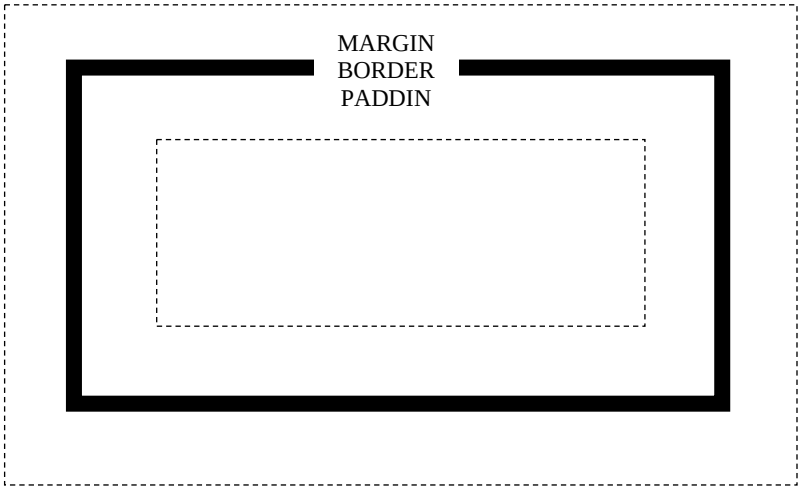
```
<head>
   <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
   <title>Periodico El Caso</title>
   <link rel="stylesheet" type="text/css" href="empresa.css" />
   <link rel="stylesheet" type="text/css" href="ultimahora.css" />
   <link rel="stylesheet" type="text/css" href="marketing.css" />
   <link rel="stylesheet" type="text/css" href="deportes.css" />
</head>
```

La precedencia viene dada por el orden. Así, cuando navegador tiene que representar un elemento,

1) Busca su estilo en la primera hoja de estilos, `empresa.css`.
   Si el estilo está definido en este fichero, se aplica dicho estilo y se ignoran los estilos que estén definidos para este elemento en alguna de las hojas de estilo siguientes.
2) En el caso de que no se encuentre la definición del estilo en esa hoja, se busca el estilo del elemento en la siguiente hoja de estilo, `ultimahora.css`.
3) El proceso ser repite hasta la última hoja de estilos.
4) En el caso de que el estilo de un determinado elemento esté definido en dos o más hojas de estilo, el navegador tiene en cuenta la hoja de estilos de más peso, es decir, la que ocupe el primer lugar en la cabecera de la página.

2.3 BOX MODEL (Modelo de cajas)

El Box Model es una herramienta de diseño para realizar una distribución visual en nuestro interfaz gráfico. **Se puede crear una caja alrededor de cualquier sección de un documento sin más que incluir la sección en una etiqueta** `<div>`. Una caja de este tipo tiene un montón de propiedades que son fácilmente manipulables por el diseñador.



La caja rodea un determinado contenido. Se puede definir el ancho y el alto o dejar que el navegador decida las dimensiones dependiendo del contenido, del tamaño de la ventana del navegador en cada momento y de otros factores que tienen que ver con la complejidad de la definición de la distribución que se haya definido para los elementos –en inglés y a partir de ahora, del layout-. Dicho de otro modo, el contenido puede "fluir" o estar en una posición y tamaño "rígidos".

Las propiedades más importantes son:

- `Border`: anchura del borde de la caja. Las CSS nos permiten dar distintos formatos a éste borde: `solid`, `dotted`, `dashed`, `none`, es decir, línea de puntos, continua, etc
- `Padding`: distancia entre el contenido y el borde de la caja.
- `Margin`: distancia entre el borde y el resto de los elementos de nuestra página.

Es importante destacar que se puede individualizar cualquiera de estas tres propiedades a cada uno de los lados. El nombre de estas propiedades serían: `Top-Border`, `Rigth-Border`, `Bottom-Border` y `Left-Border`. Los nombres correspondientes a cada uno de los lados de nuestro objeto se construyen de la misma forma. Por supuesto que

pueden utilizarse estas tres propiedades en un lado únicamente y no es necesario utilizarlo en el resto.

Sin embargo, es importante tener en cuenta que el tamaño final de nuestro objeto es la suma del área del contenido interno de nuestro objeto, del `border`, del `padding` y del `margin`.

Por ejemplo:

```
#caja {
      margin: 20px;
      border: 1px solid #000000;
      padding: 10px;
      width: 300px;
}
```

El ancho total de este elemento será:

- Lado izquierdo: 20px + 1px + 10px
- Contenido: 300px
- Lado derecho: 20px + 1px + 10px
- TOTAL: 362px

Cuando se aplica un color de fondo a un determinado elemento que tiene margen, borde y padding, el color de fondo sólo se aplica al contenido del elemento y al padding, no al margen.

Se puede ver un ejemplo muy clarificador en el siguiente video-tutorial:

Propiedades Border, margin y padding de 8m 22seg de duración. En este video tutorial se explican los conceptos de estas tres propiedades. Para la explicación se utiliza un ejemplo muy sencillo donde se aplican estas propiedades a un párrafo.

3 CAPÍTULO III: LAS REGLAS

3.1 INTRODUCCIÓN

En este capítulo vamos a entrar de lleno a estudiar cómo se definen los estilos.

Los estilos se definen por medio de reglas. **Cada regla está compuesta** de un **identificador** para distinguirla de las demás y un **bloque de código** donde propiamente se define el estilo.

Una regla define un estilo. Para aplicar un estilos a un elemento, dicho elemento tiene que estar "identificado". De manera muy general y simple –porque realmente no es así, pero para entender los fundamentos podemos pensar que es así- un elemento web se puede identificar de tres formas:

- Por el nombre del elemento.
- Por el atributo `ID`
- Por el atributo `class`.

Por ejemplo, para definir un estilo con un margen de 5px para las cabeceras de tipo `<h3>`, se utiliza la siguiente regla:

```
h3{ margin: 5px;}
```

La reglas tienen siempre dos partes: un selector o identificador y una definición.

- El **selector** es el nombre del elemento XHTML que sirve para identificar el o los elementos sobre los que se va aplicar el estilo o regla. En el ejemplo anterior el identificador es `h3` que identifica las cabeceras a las que se va aplicar el estilo.
- La **definición** es el bloque de código que define el estilo. Va siempre entre llaves y está compuesto por **declaraciones**. Pueden ser una –como mínimo- o varias declaraciones en cada definición. Cada declaración tiene a su vez dos partes: la **propiedad** y el **valor.** La propiedad es la propiedad del elemento XHTML sobre la que se desea definir el estilo: `margin`. El valor es el valor que se quiere aplicar a dicha propiedad: `5px`. El anterior estilo define un margen –propiedad `margin-` de `5 pixels` para todos los elementos `<h3>` de la página. Las declaraciones deben estar separadas entre sí por un punto y coma.

Otro ejemplo de regla:

```
h3{ margin: 0px; font-family: sans-serif; color: #22EE00; }
```

En este caso, la definición del estilo del elemento `<h3>` tiene tres declaraciones, cada una de ellas define una propiedad.

3.2 FORMAS ABREVIADAS

Las hojas de estilo son los ficheros de texto con extensión `css`, en donde se definen los diferentes estilos o reglas de los elementos de nuestra página web.

Normalmente, en las hojas de estilo se utiliza una línea por definición, para que la lectura sea un poco más clara. El aspecto del estilo anterior se lee mejor así:

```
h3{
      margin: 0px;
      font-family: sans-serif;
      color: #22EE00;
}
```

En general, la sintaxis de las CSS no es demasiado estricta: no hay problema con los espacios en blanco, o con escribir cada definición en una línea o varias. Sin embargo, **siempre hay que escribir los pares propiedad-valor, separados por dos puntos y acabado en punto y coma.**

Cada navegador define estilos por defecto para cada elemento. Por ejemplo, una cabecera `<h1>` tiene un tamaño de letra mayor que la que tiene un párrafo. Por eso, cada regla no tiene que definir valores para todas las propiedades que tenga ese elemento sino sólo para aquéllas que quiera modificar o sobrescribir.

Las CSS permiten varios caminos para hacer las mismas cosas que ayudan o bien a abreviar el código o bien a hacerlo más legible. Por ejemplo, los tres estilos siguientes definen lo mismo:

```
h3{
      margin: 0px;
{

h3{
      margin: 0px 0px 0px 0px;
}

h3{
      margin-top: 0px;
      margin-rigth: 0px;
      margin-bottom: 0px;
      margin-left: 0px;
}
```

Los tres estilos definen un margen de 0 pixeles.

- La primera versión es la más simplificada. Cualquier valor será aplicado a los cuatro márgenes.
- La segunda versión especifica un valor para cada uno de los márgenes, que por supuesto pueden ser distintos. En el caso de utilizar esta forma, es necesario explicitar los valores de cada uno de los cuatro márgenes. Además, el orden es el de las agujas del reloj comenzando por el margen superior.

- La tercera versión es la más larga pero sin embargo la más sencilla de leer: aquí, cada propiedad se especifica de manera individualizada. En esta última versión no hay problema en variar el orden de las propiedades. Lógicamente, esto último no se puede hacer con la segunda versión.

Hay formas abreviadas para cuando no se desea definir un estilo para todas las posibilidades de la propiedad. Por ejemplo, se podría definir un estilo de la siguiente manera:

```
h3{
      margin-top: 0px;
      margin-bottom: 0px;
}
```

Una forma más breve para definir estos dos valores en concreto es:

```
h3{
      margin: 0px 0px;
}
```

Creo sinceramente que no merece la pena conocerlas a fondo y creo también que es mejor no utilizarla porque puede llevar a errores. Ya vemos que la casuística puede llegar a infinito. En este caso, los dos márgenes que no se definen se quedarían en los valores que por defecto el navegador tenga definido.

Por supuesto que hay definidas más formas "cortas" para cuando no se desean definir todas las propiedades de un determinado elemento, pero –vuelvo a insistir- creo que es mejor no utilizarlas.

Sin embargo, creo que es bueno conocer las tres formas anteriores y porque los programadores utilizan las tres. Por supuesto que diferentes propiedades tienen distintos métodos abreviados. Por ejemplo, la propiedad `border`, puede definirse de manera abreviada así:

```
div {
      border: 15px dotted #FF8000;
}
```

Este estilo puede escribirse de la siguiente forma:

```
div {
      border-width:15px;
      border-style:dotted;
      border-color: #FF8000;
}
```

Pero también podría definirse explicitando los valores de cada propiedad:

```
div{
      border-top-width:15px;
      border-right-width:15px;
      border-bottom-width:15px;
      border-left-width:15px;
```

```
      border-top-style:dotted;
      border-right-style:dotted;
      border-bottom-style:dotted;
      border-left-style:dotted;

      border-top-color: #FF8000;
      border-right-color: #FF8000;
      border-bottom-color: #FF8000;
      border-left-color: #FF8000;

}
```

Podrían omitirse algunos valores de algunas propiedades. En ese caso, como anteriormente se ha explicado, el navegador aplicaría los valores por defecto.

Es muy importante destacar que no importa la complejidad o la forma abreviada que se utilice para definir un estilo. En cada definición se sigue siempre la siguiente regla:

```
selector {
      propiedad : valor ;
      propiedad : valor ;
      …
      …
}
```

3.3 PROPIEDADES

Vamos ahora a explicar algunas propiedades para los diferentes elementos. Sólo vamos a estudiar dos o tres que creo que son suficientes para comprender perfectamente lo que son las CSS. Para profundizar más es mejor tener a mano una guía de referencia.

3.3.1 COLOR

Los valores de la propiedad `color` son los mismos que se utilizan en el lenguaje XHTML. Para definir un color en un estilo, se pueden utilizar los nombres de los colores `aqua`, `black`, `blue`, …, `olive`, `orange`, `red`,… o bien definir los colores en formato RGB por medio de un valor hexadecimal, con un rango que va desde el valor `#000000` (blanco) al valor `#FFFFFF` (negro).

Las CSS también permiten formas cortas para los valores hexadecimales. Es posible reducirlos a tres. En realidad, cada uno de los valores se “duplica”, es decir:

El valor `#55FF44` se puede expresar como `#5F4`.

El color RGB también puede expresarse en formato porcentual o decimal, con tres valores que van desde el 0 al 255. Así, los tres valores siguiente son el mismo:

```
#0F0      #00FF00      rgb(0%,100%,0%)      rgb(0,255,0)
```

Sin embargo, creo que todas estas formas abreviadas, a veces, tienen poco sentido cuando se trata de diseñar porque habitualmente escribiremos nuestros estilos con la ayuda de una herramienta gráfica, a golpe de ratón. Es importante conocer todas estas formas de abreviar el código porque una vez que terminemos de utilizar nuestras herramientas gráficas, es posible que deseemos "retocar" directamente el texto de nuestro código.

Ejemplo:

La siguiente regla define un estilo para los párrafos, en el que el color de fondo es azul y el color de la letra es blanco:

```
p{
      background-color : blue ;
      color : #000000 ;
}
```

En el video tutorial Propiedad Color de 7m 54seg de duración, se explican con un ejemplo las propiedades Color y Background-color que proporcionan un color al texto y a los objetos respectivamente.

3.3.2 TAMAÑO (SIZE)

Lo más habitual es utilizar valores en píxels (`px`), porcentaje (`%`) o en `em`

a) Los píxels definen valores absolutos, mientras que el porcentaje y los em definen valores relativos. De esta forma, es el propio navegador el que define el tamaño final dependiendo de las características de cada navegador y de la propiedad en cuestión. Sin embargo un valor de `100px` significa que será el valor de `100 px` según la pantalla y el sistema operativo del usuario. Es decir, en todos los navegadores se verá con el mismo tamaño.

b) Muchas veces se define un valor porcentual para el tamaño de la anchura del elemento para que el propio navegador calcule ese valor dependiendo de la anchura de la ventana del navegador en cada momento, o del elemento que a su vez contenga a ese elemento, es decir su contenedor o "padre". La forma sería:

```
div {
      width: 50%;
}
```

Generalmente, una caja desciende –es "hija"- del elemento `<html>` y de `<body>` que muchas veces no tienen definido un estilo determinado. En el caso anterior, este objeto `<div>` tendrá la mitad de la anchura de la página. Sin embargo, si esta caja está a su vez contenida en otra caja que tenga también una anchura del `50%`, entonces la anchura de esta caja será la mitad de la mitad de la anchura de la página en ese momento, es decir, tendrá una anchura igual a la cuarta parte de la anchura de la página.

La herencia –ver capítulo anterior- se aplica también cuando se utiliza el valor porcentual para los tamaños. Por ejemplo, si se declara en el elemento `<body>` un

tamaño de letra de `40px` y el tamaño de letra del elemento párrafo es del `80%`, entonces, el navegador calcula el tamaño de la letra del párrafo y se escribe con un tamaño de `32 px`, es decir, el `80%` de `40`.

c) `em` es similar a los porcentajes, Un `em` es el tamaño de la altura de la letra del sistema operativo del usuario que tiene por defecto, o bien la altura del tamaño de la letra del elemento "padre". Por ejemplo,

```
body{
	font-size: 20px;

}

h3{
	font-size: 1.5em;
}
```

El tamaño de la letra del encabezamiento `<h3>` es de `30px`, es decir, que `1.5em` equivale a un valor porcentual de `150%`.

En el video tutorial Propiedad size de 8m 59seg. De duración se explican algunas propiedades que tienen que ver con las distancias o tamaños como pueden ser `width`, `font-size`, `margin`, etc... También se explica que podemos definir los tamaños de los objetos con diferentes unidades, pixeles, % y em.

4 CAPÍTULO IV: LOS SELECTORES

4.1 INTRODUCCIÓN

Como hemos venido diciendo hasta ahora, cada regla tiene dos partes: **selector** y **definición**.

- El selector identifica a qué elemento de la página Web el estilo.
- La definición le dice al navegador qué cambiar y cómo con respecto a la representación por defecto.

Como muestra, en el siguiente ejemplo, el selector le dice al navegador que aumente el tamaño predeterminado del contenido del elemento `h2` en un 120%.

```
h2 {
      font-size:120%;
}
```

En el ejemplo, `h2` es el selector y está seguido por la definición que es el bloque entre paréntesis que define el estilo. Dicho bloque está compuesto por una o varias sentencias de la forma

```
propiedad : valor ;
```

Por tanto, para definir el estilo para un elemento, la sintaxis es:

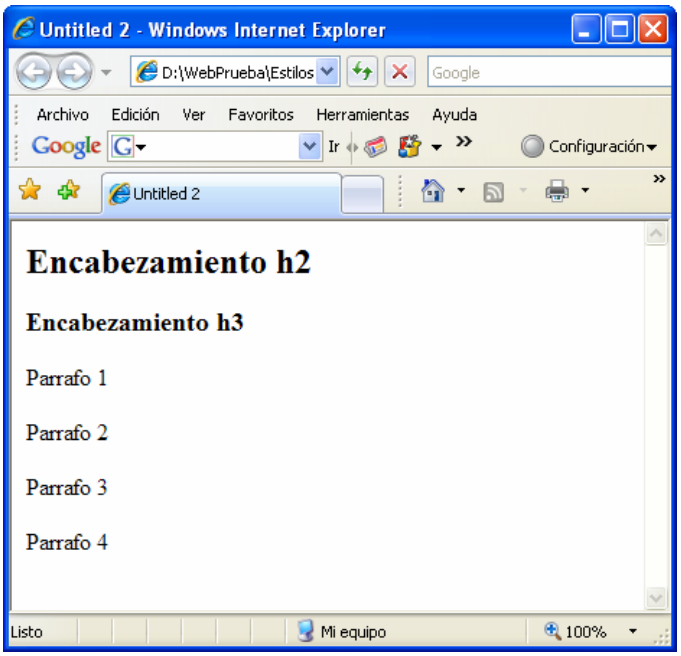
```
elemento {
      propiedad : valor ;
      propiedad : valor ;
      propiedad : valor ;
      …
      …
}
```

En el siguiente apartado comenzamos a estudiar los tipos de selectores y posteriormente estudiaremos con detalle los bloques.

Para este capítulo vamos a trabajar con la siguiente página web:

```
<body>
      <h2>Encabezamiento h2</h2>
      <h3>Encabezamiento h3</h3>
      <p>Parrafo 1</p>
      <p>Parrafo 2</p>
      <p>Parrafo 3</p>
      <p>Parrafo 4</p>
</body>
```

La representación de esta página es:



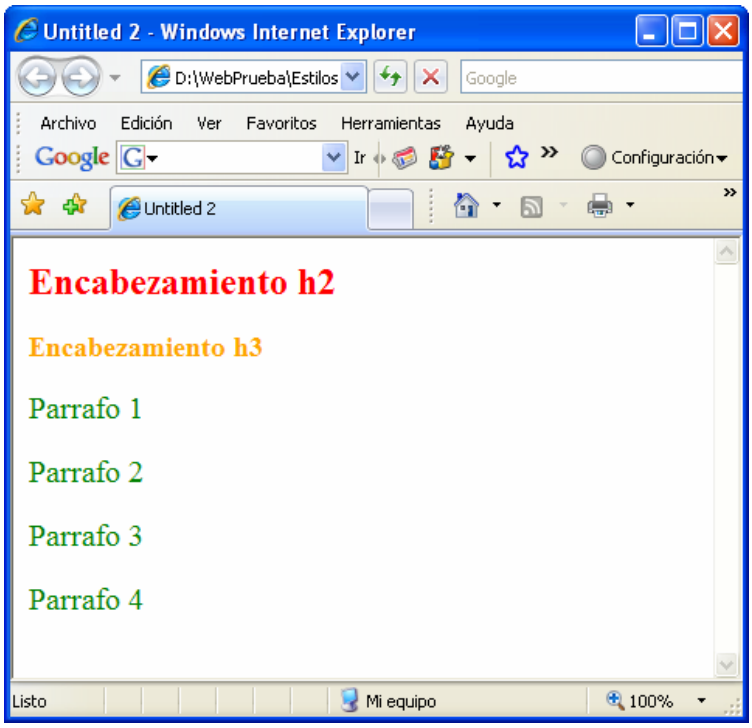
Lo primero que salta a la vista es que el propio navegador define unos estilos por defecto como son, por ejemplo el tamaño y color de la letra de cada elemento, el color de fondo, etc…

Todos los estilos que se definan se sobrescriben a los valores o estilos por defecto. Por eso, si definiésemos un estilo con letra de color rojo para los párrafos, estaríamos sobrescribiendo la propiedad `color` de los párrafos.

Una hoja de estilos típica podría ser la siguiente:

```
body{
  font-size: 40px;
  color:blue;
}
h2{
  color: red;
}
h3{
  color:orange;
}
p{
  font-size : 20px;
  color : green;
}
.miClase{
  font-family:"Lucida Sans";
  color:red;
}
#miID {
  margin-left: 165px;
  font-family:"Courier";
  color: blue;
}
```

Si se aplica esta hoja de estilos a la página anterior, la representación en Internet Explorer es:



En el ejemplo anterior, son selectores `body`, `h2`, `h3`, `p`, `.miClase` o `#mainID` y los bloques son el código que está entre llaves por cada selector.

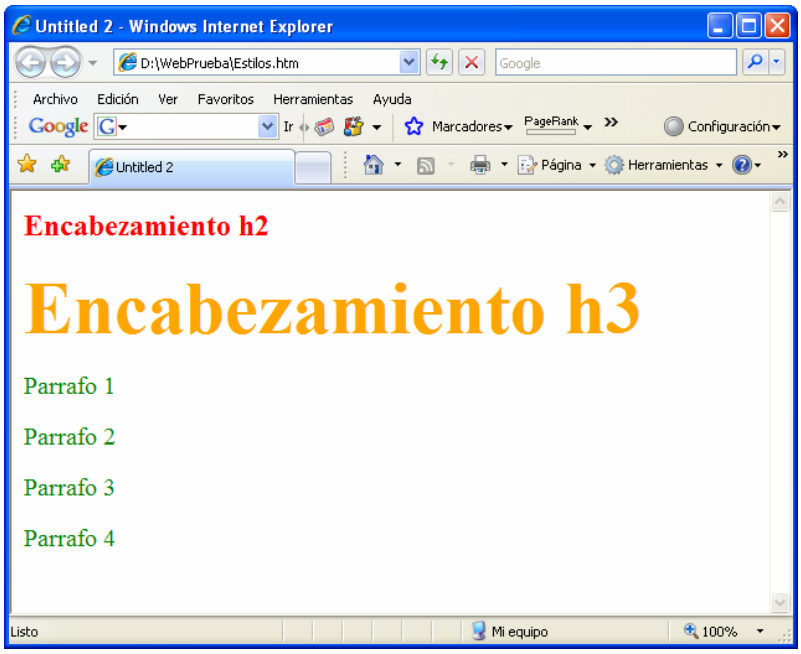
Para aplicar los estilos se utilizan tres formas diferentes, que se corresponden con los tres tipos de selectores:

a) Selectores de tipo o redefinición de las etiquetas: Como hemos venido diciendo, los navegadores presentan cada elemento con un estilo por defecto. Por ejemplo, los elementos `<h1>` tienen un tamaño de letra diferente –y mayor- que los elementos `<h3>` aunque el tipo de letra por defecto sea el mismo. Además, es interesante saber que los estilos que define cada navegador para los diferentes elementos no son iguales. Por eso, a veces un párrafo se representa de manera diferente en dos navegadores distintos. Para variar la presentación por defecto de un elemento basta con redefinir su estilo. Para eso, se utilizan los selectores de tipo, que nombran el elemento o etiqueta XHTML a la que aplicar el estilo. Si en el ejemplo anterior queremos modificar el tamaño que se aplica por defecto al elemento `h3`, podríamos escribir:

```
body{
  font-size: 40px;
  color:blue;
}
h2{
  color: red;
}
h3{
  color:orange;
  font-size:60px;
}
p{
  font-size : 20px;
  color : green;
```

```
}
.miClase{
  font-family:"Lucida Sans";
  color:red;
  font-size:100px;
}
#miID {
  margin-left: 165px;
  font-family:"Courier";
  color: blue;
}
```

La representación es:



En el ejemplo, `h2`, `body`, `h3` y `p` son selectores de tipo: cada uno de los tres bloques define –en realidad, re-define- un estilo para los elementos `<h2>`, `<body>`, `<h3>` y `<p>`, respectivamente.

En el video tutorial Selectores I de 3m 28seg. de duración se explica el primer tipo de selector, que es la etiqueta o el nombre del elemento. Esta es una primera forma de identificar el elemento al que se desea aplicar el estilo.

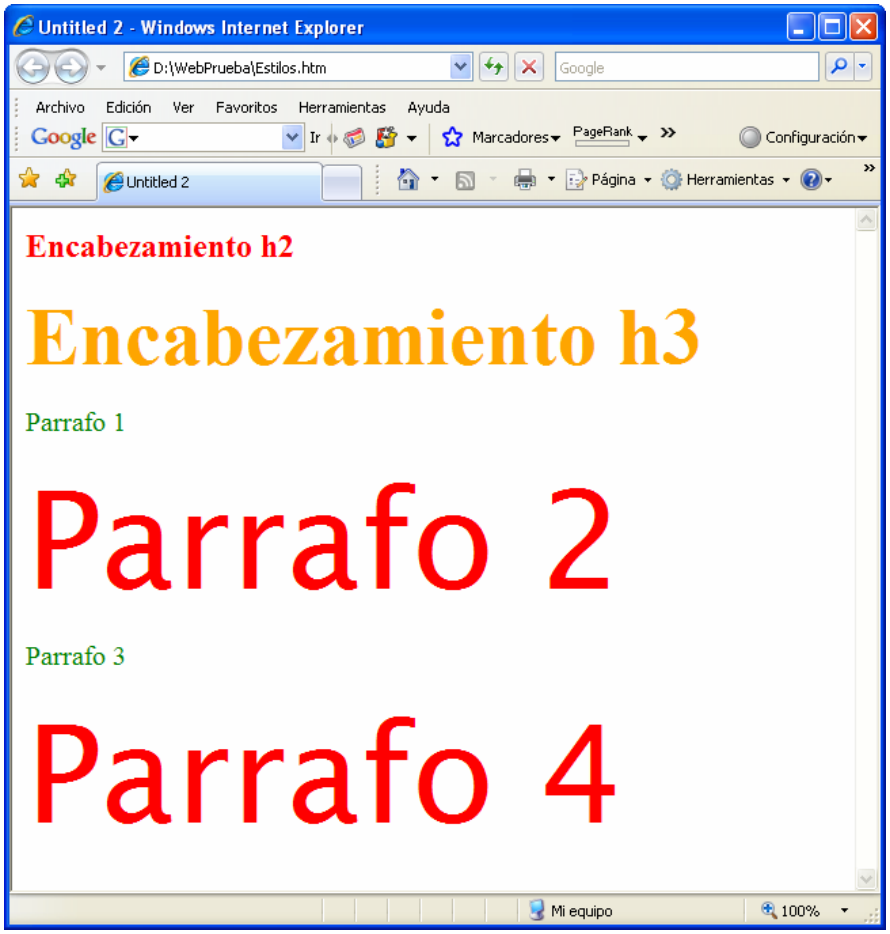
b) Selectores de clase o clase: Las clases –o los selectores de clase- se identifican con el nombre de la clase precedido por un punto. Por ejemplo, .miClase. Permiten definir un estilo y asignarlo a diferentes elementos. Por ejemplo, si queremos resaltar los párrafos 2 y 4 el nombre de la clase se coloca como atributo class en esos párrafos. El estilo se aplica a todos los elementos que tengan ese atributo class**.**

Por ejemplo, si utilizamos el mismo fichero `.css` anterior y modificamos nuestra página web a:

```
<body>
  <h2>Encabezamiento h2</h2>
  <h3>Encabezamiento h3</h3>
```

```
    <p>Parrafo 1</p>
    <p class="miClase">Parrafo 2</p>
    <p>Parrafo 3</p>
    <p class="miClase">Parrafo 4</p>
</body>
```

La página se vería del siguiente modo:



Es decir, se inserta como atributo en dicho párrafo el atributo `class` de valor el nombre de la clase –en este caso `miClase-`, sin el punto inicial:

```
<p class=miClase”> Párrafo 2 </p>
```

De esta forma podemos personalizar el estilo de un párrafo determinado.

En el video tutorial Selectores II de 4m 13seg. de duración se explica cómo definir estilos mediante clases. Estos estilos serán aplicados a aquéllos elementos que tengan un atributo con el nombre de la clase. Se explica también que cuando hay conflicto la clase sobrescribe el estilo definido para el elemento.

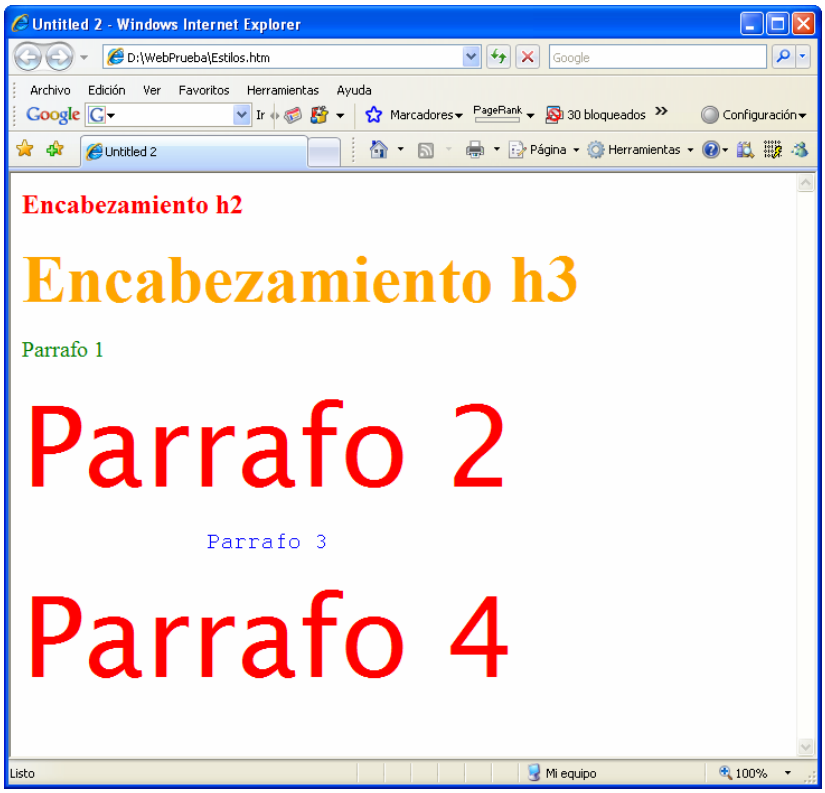
c) Identificador o selector de ID: se identifican por el símbolo # que precede al nombre del identificador. El estilo se aplica al elemento cuyo atributo id sea ese identificador. Por ejemplo, si modificamos el fichero .css anterior incluimos el código

```
<p id=”miID”> Párrafo 3</p>
```

el objeto cuyo `id` es `miID` tendrá el margen izquierdo definido en la hoja de estilos. El tipo de letra será `Courier` y azul. El fichero `.css` quedará:

```
<body>
  <h2>Encabezamiento h2</h2>
  <h3>Encabezamiento h3</h3>
  <p>Parrafo 1</p>
  <p class="miClase">Parrafo 2</p>
  <p id=”miID”>Parrafo 3</p>
  <p class="miClase">Parrafo 4</p>
</body>
```

Su representación es:



En el videotutorial Selectores III de 3m 28seg. de duración se explica los identificadores. Al final del video se hace un pequeño resumen de los tipos de selectores.

4.2 SELECTORES SEPARADOS POR COMAS

Si se desea definir el mismo estilo en varios elementos, se escriben los selectores correspondientes separados por comas. Por ejemplo, si se desea que el margen de los elementos `<h1>`, `<h2>` y `<h3>` sea `15 pixels`, entonces se puede definir de la siguiente manera:

```
h1,h2,h3{
```

```
      margin: 15px;
}
```

La anterior definición es equivalente a ésta:

```
h1{
      margin: 15px;
}

h2{
      margin: 15px;
}

h3{
      margin: 15px;
}
```

Claro que la primera forma es más clara y abreviada. Esto se puede hacer no sólo cuando los estilos sean exactamente iguales, sino cuando tengan alguna propiedad en común. En este sentido es equivalente escribir:

```
h1{
      margin: 15px;
      color: orange;
      font-family: sans-serif;
      font-size: 30px;
}

h2{
      margin: 15px;
      color: orange;
      font-family: sans-serif;
      font-size: 20px;
}

h3{
      margin: 15px;
      color: orange;
      font-family: sans-serif;
      font-size: 15px;
}
```

Que escribir:

```
h1,h2,h3{
      margin: 15px;
      color: orange;
      font-family: sans-serif;
}

h1{
      font-size: 30px;
}

h2{
      font-size: 20px;
}
```

```
h3{
	font-size: 15px;
}
```

4.3 SELECCION CONTEXTUAL

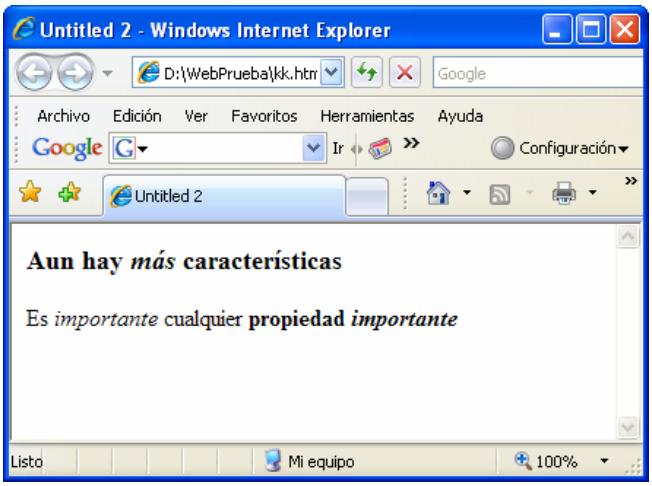
4.4 SELECTOR HIJO

No siempre el estilo que se define se aplica a todas las instancias de un determinado elemento.

Por ejemplo:

```
<body>
  <h3>Aun hay <em>más </em>características </h3>
  <p>Es <em>importante</em> cualquier <strong>propiedad <em>importante</em></strong></p>
</body>
```

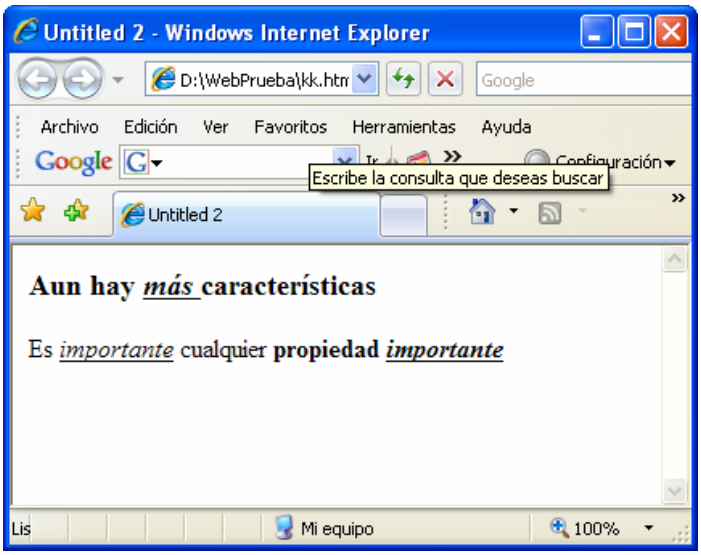
Se representa de la siguiente manera:



Si queremos definir un estilo para subrayar únicamente los elementos `<em>` que están contenidos en párrafos pero no en el resto de los elementos, no podemos hacerlo de siguiente forma:

```
em{

	text-decoration:underline;
}
```

La representación del fichero html anterior es:



En las CSS, el selector para el elemento que está contenido en un párrafo es:

```
p em
```

es decir,

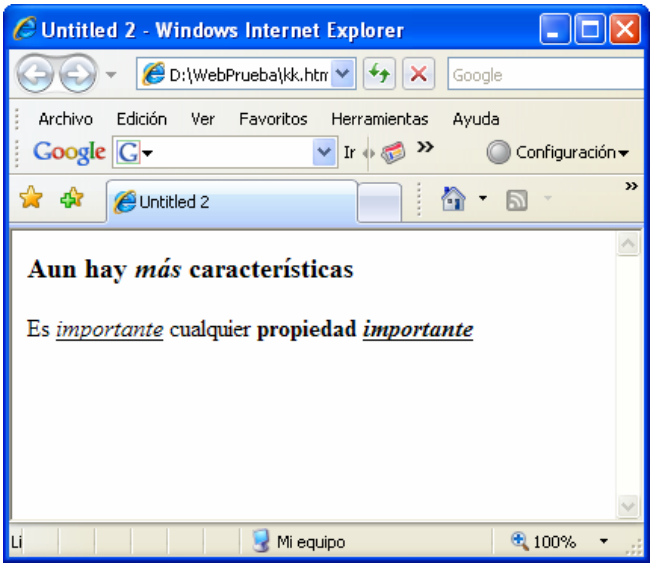
```
selector_padre selector_hijo
```

ambos separados por un espacio en blanco, no por una coma.

Por lo tanto, nuestra hoja de estilos será:

```
p em{
      text-decoration:underline;
}
```

La representación en el navegador es la que deseábamos:



4.5 SELECTOR HIJO DIRECTO

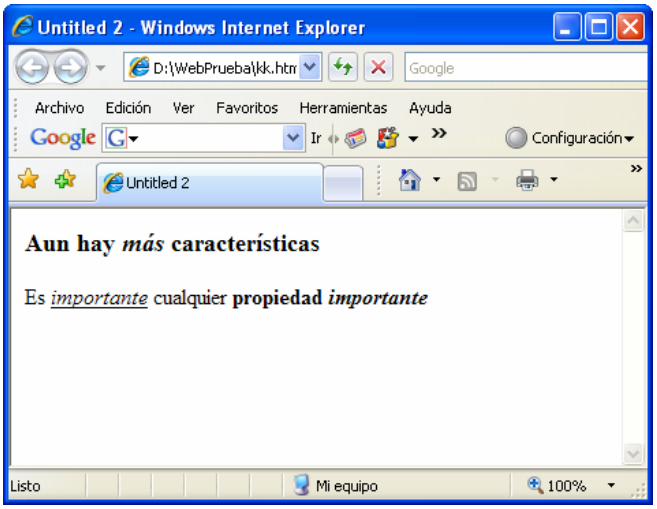
Pero imaginemos que sólo queremos subrayar los elementos que están contenidos en un párrafo, pero que no están en negrita, es decir, los que son hijos directos –no "nietos" del elemento párrafo, dicho de otra forma, los elementos que están contenidos únicamente en un párrafo. Para referirse a un elemento "hijo directo" las CSS utilizan el siguiente selector:

```
selector_padre > selector_hijo
```

es decir, lo mismo que antes pero con el símbolo > entre ambos elementos, padre e hijo. Nuestra hoja de estilos quedaría:

```
p > em{

      text-decoration:underline;
}
```

Y en el navegador:

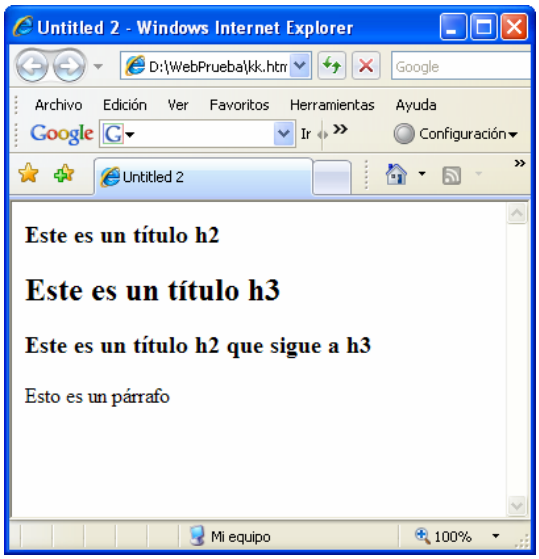


4.6 SELECTOR ELEMENTOS ADYACENTES

La página web con el siguiente contenido:

```
<body>
      <h2>Este es un título h2</h2>
      <h3>Este es un título h3 </h3>
      <h2>Este es un título h2 que sigue a h3</h2>
      <p>Esto es un párrafo</p>
</body>
```

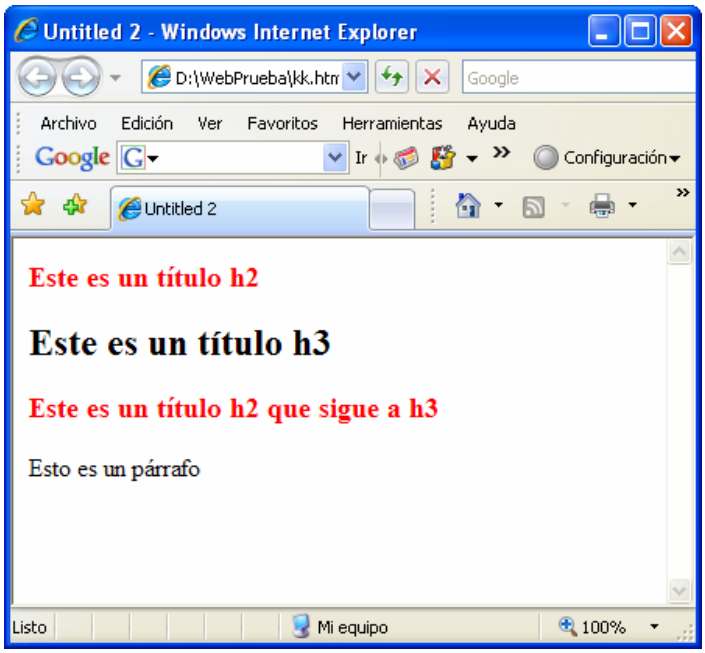
Se visualizará, si no se le aplica ninguna hoja de estilos, de la siguiente forma:



Si queremos definir un estilo para los elementos h2 para que el color de la letra sea rojo, escribiríamos una hoja de estilos con el siguiente código:

```
h3{
	color:red;
}
```

Lo veríamos así enm el navegador:



Pero si queremos que sólo se vean en rojo los títulos `<h3>` que siguen a encabezamientos `<h2>`, tenemos que utilizar el selector:

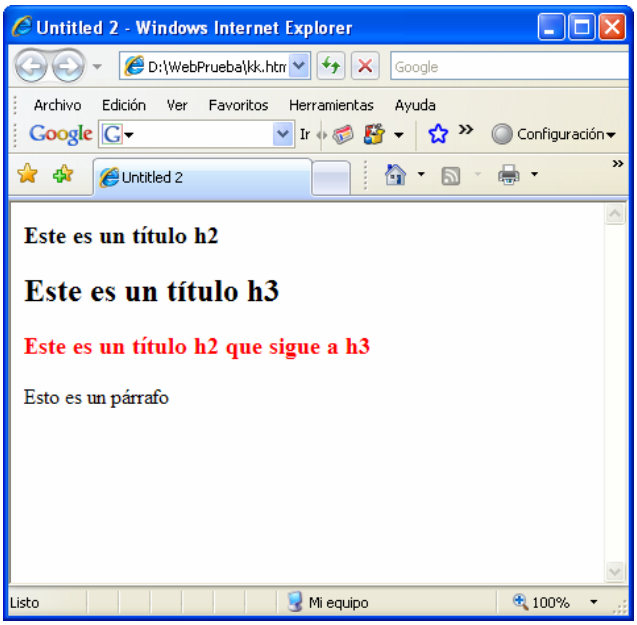
```
h2 + h3
```

Por lo tanto el código quedaría:

```
h2 + h3{
```

```
        color:red;
}
```

Y veríamos la página:



Podríamos preguntarnos cuándo utilizar selectores ID o selectores de clase. En mi opinión, se deben utilizar selectores de clase cuando se necesite aplicar un estilo múltiples veces dentro de un documento. Los selectores ID se deben utilizar para estilos que aparecen solamente una vez dentro de un documento.

5 CAPÍTULO V: LAS CAJAS o BOXES. POSICIONAMIENTO

Las cajas CSS son una herramienta que proporciona potencia y flexibilidad al diseñador web. Sin embargo, si no se conoce bien cómo trabajan, las cajas se pueden convertir en una fuente de problemas. Además, según el navegador que se utilice hay diferencias en cómo distribuyen los elementos en la pantalla del usuario final. El propósito de este capítulo es proporcionar al lector un sólido conocimiento de cómo trabajan las cajas y cómo se posicionan en la pantalla. Más adelante estudiaremos las diferencias entre los navegadores.

5.1 FLOW (FLUJO)

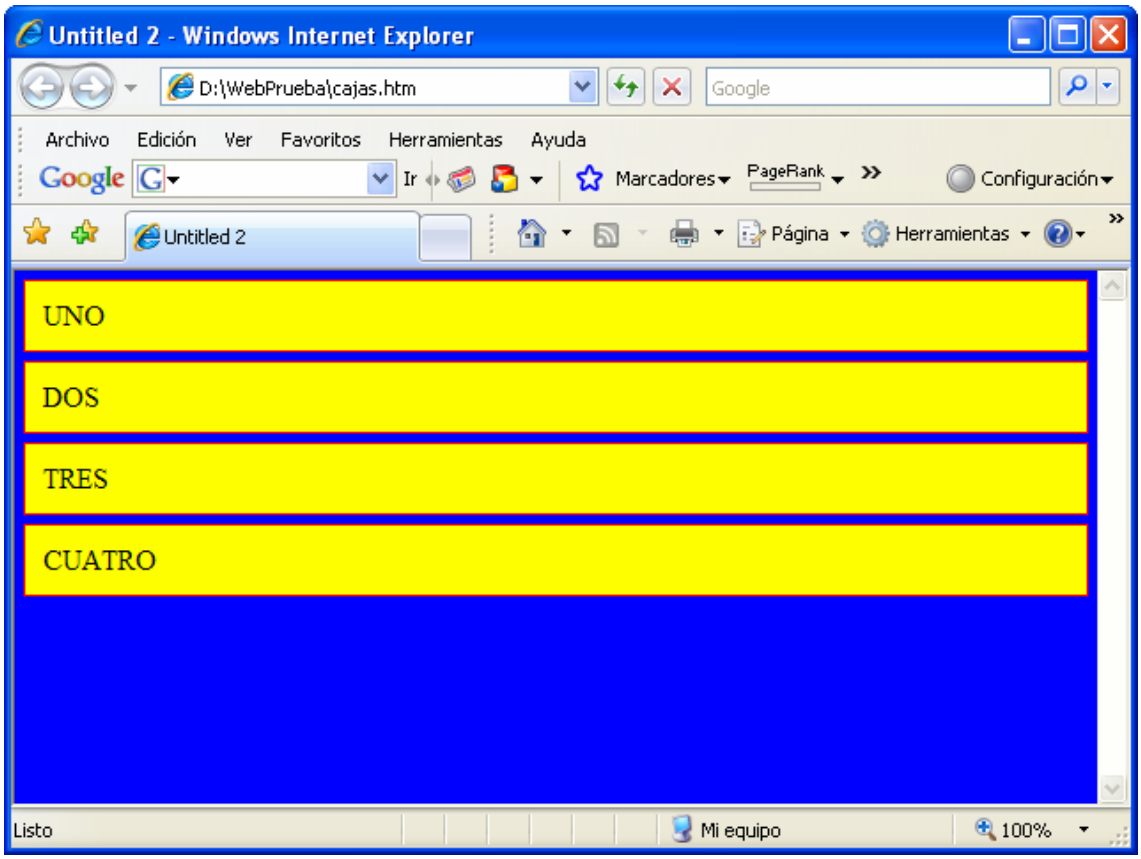
El flujo en un documento web es muy simple y es el concepto más importante para entender cómo distribuyen las CSS los objetos o elementos en la pantalla.

El la distribución de los elementos en un flujo es algo parecido a lo que sucede con un procesador de textos: la primera palabra que se escribe se coloca en la esquina superior izquierda de la pantalla y cada nueva palabra que se añade se colocara a la derecha de la última palabra escrita. Cuando una palabra ya no cabe en esa línea, se salta a la siguiente. La distribución de elementos web en una página es igual: el primer elemento se colocará en la esquina superior izquierda de la pantalla. Cada nuevo elemento que se añada será ubicado en la parte derecha del último elemento colocado. Cuando se añade un elemento y no haya espacio suficiente en la derecha del último elemento, se colocará en la parte izquierda de una nueva línea.

Sin embargo, las cajas tienen un comportamiento un poco diferente, porque cada caja ocupa todo el ancho de una página. Por ejemplo:

```
<body>
      <div class="caja">UNO</div>
      <div class="caja">DOS</div>
      <div class="caja">TRES</div>
      <div class="caja">CUATRO</div>
</body>
```

```
body {
      margin: 0px;
      padding: 0px;
      color: black;
      background-color: blue;
}
.caja {
      padding: 10px;
      margin: 5px;
      background-color: yellow;
      border: 1px solid red;
      width:20%;
}
```



Cada bloque ocupa siembre una línea. Esto es así a menos que se diga lo contrario, un bloque. Por lo tanto, el siguiente elemento no "cabe" en esa línea y se coloca debajo de él, en la siguiente línea. En este caso, cada bloque tiene un `margin` de `5px` y un `border` de `1px`, que se tienen en cuenta tanto en el comienzo como en el final del bloque.

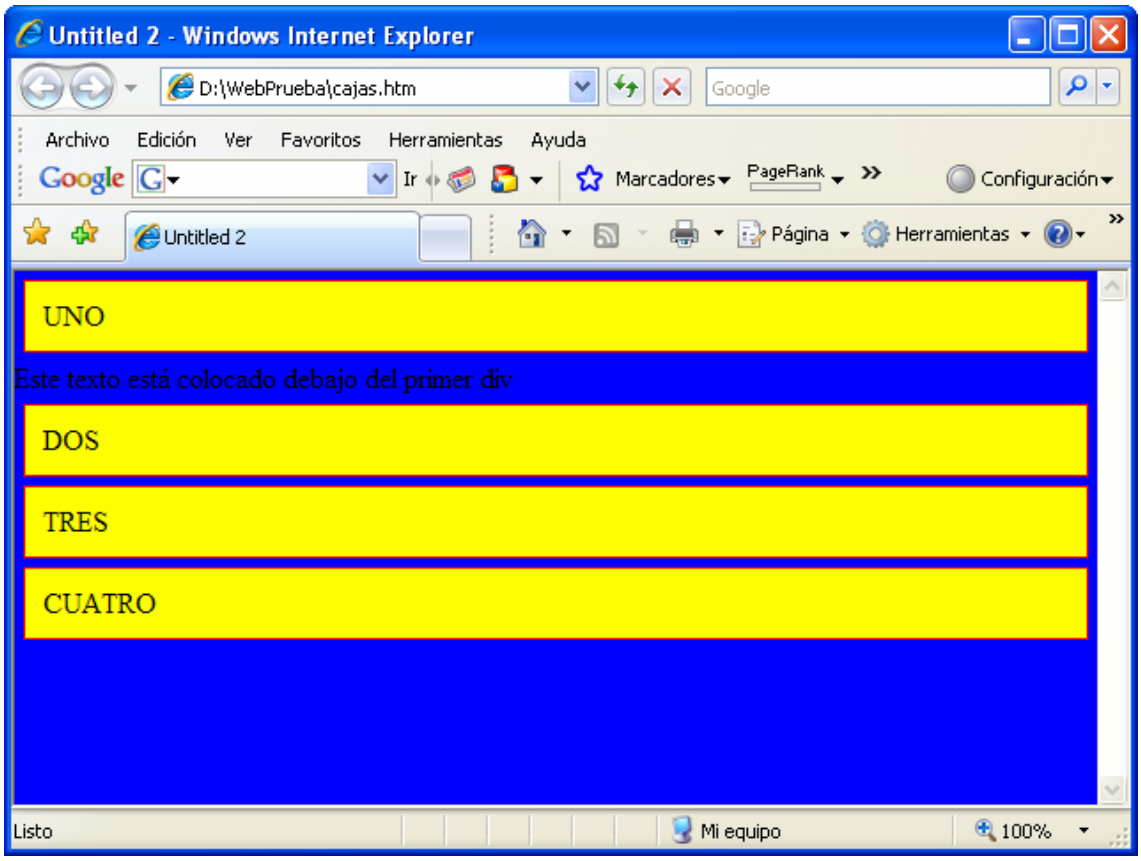
En el caso anterior, la representación es así porque `<div>` es un elemento a nivel de bloque y por lo tanto, cada caja comienza una nueva línea. Los elementos se apilan uno debajo de otro.

En el videotutorial Posicionamiento static de 6m 22seg. de duración se explica el `flow layout` o distribución en flujo, que es el posicionamiento por defecto en la web. Corresponden al valor `static` de la propiedad `position`. Con este posicionamiento el primer elemento se coloca en la esquina superior izquierda y los siguientes elementos se van colocando, en la misma línea pero a la derecha del último elemento. Si no cabe en esa línea se salta a la siguiente. Es lo más parecido a un procsador de textos que escribe cada palabra a la derecha de la anterior y cuando no cabe en una línea salta hasta la siguiente. Se explica también que los bloques o elementos ocupan todo el ancho de la página, por lo que el siguiente elemento se coloca en la siguiente línea. En cada línea sólo puede haber un elemento div

Si colocamos un texto entre bloques:

```
<body>
	<div class="caja">UNO</div>
	Este texto está colocado debajo del primer div
	<div class="caja">DOS</div>
	<div class="caja">TRES</div>
	<div class="caja">CUATRO</div>
</body>
```

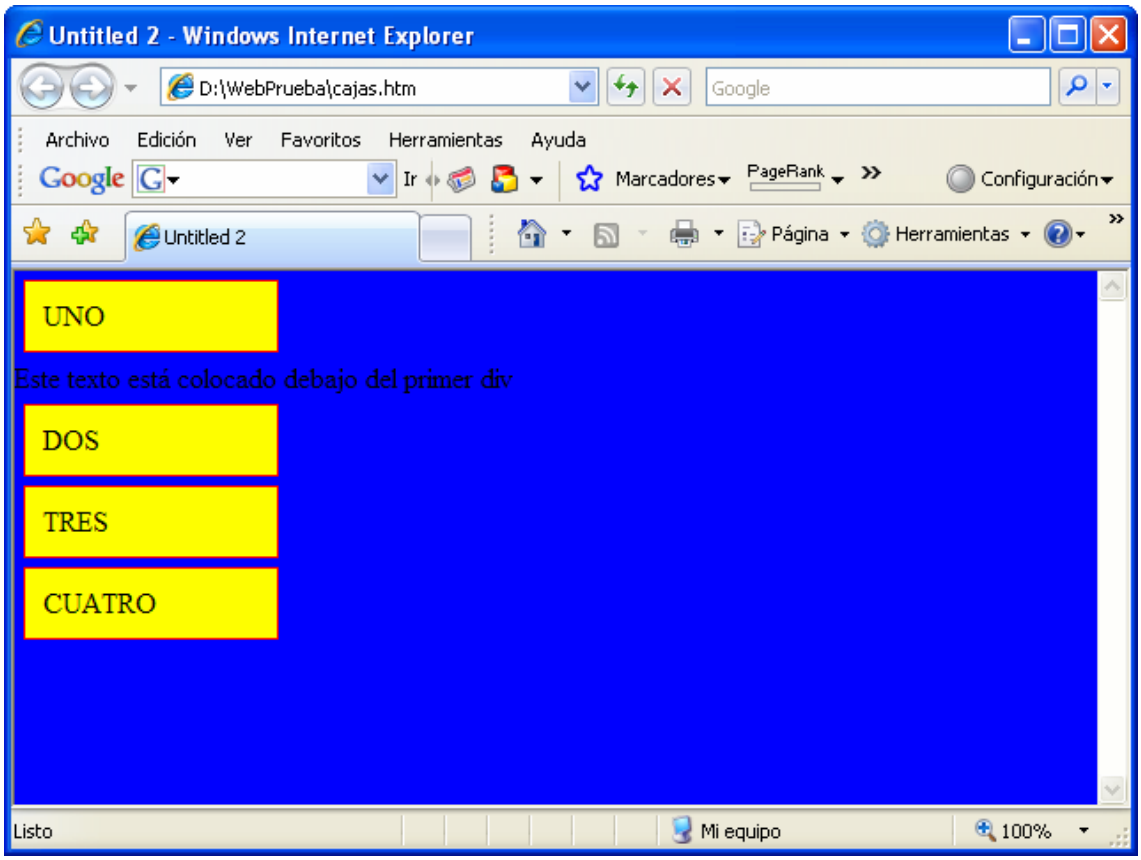
La página se vería así:



Vamos ahora a definir la anchura de los bloques `<div>`, por ejemplo, al `20%` de la anchura de la página.

```
body {
	margin: 0px;
	padding: 0px;
	color: black;
	background-color: blue;
}
.caja {
	padding: 10px;
	margin: 5px;
	background-color: yellow;
	border: 1px solid red;
	width:20%;
}
```

Esta página se visualiza así:



Vemos que los elementos se colocan apilados en vertical, no uno al lado del otro. En realidad sí están "fluyendo" como si fueran palabras de un procesador de textos, ya que **cada bloque**, aunque no tenga toda la anchura **siempre comienza en una nueva línea y cualquier otro contenido o elemento es colocado en la siguiente línea**.

5.2 TIPOS DE POSICIONAMIENTO

El posicionamiento de un elemento puede ser:

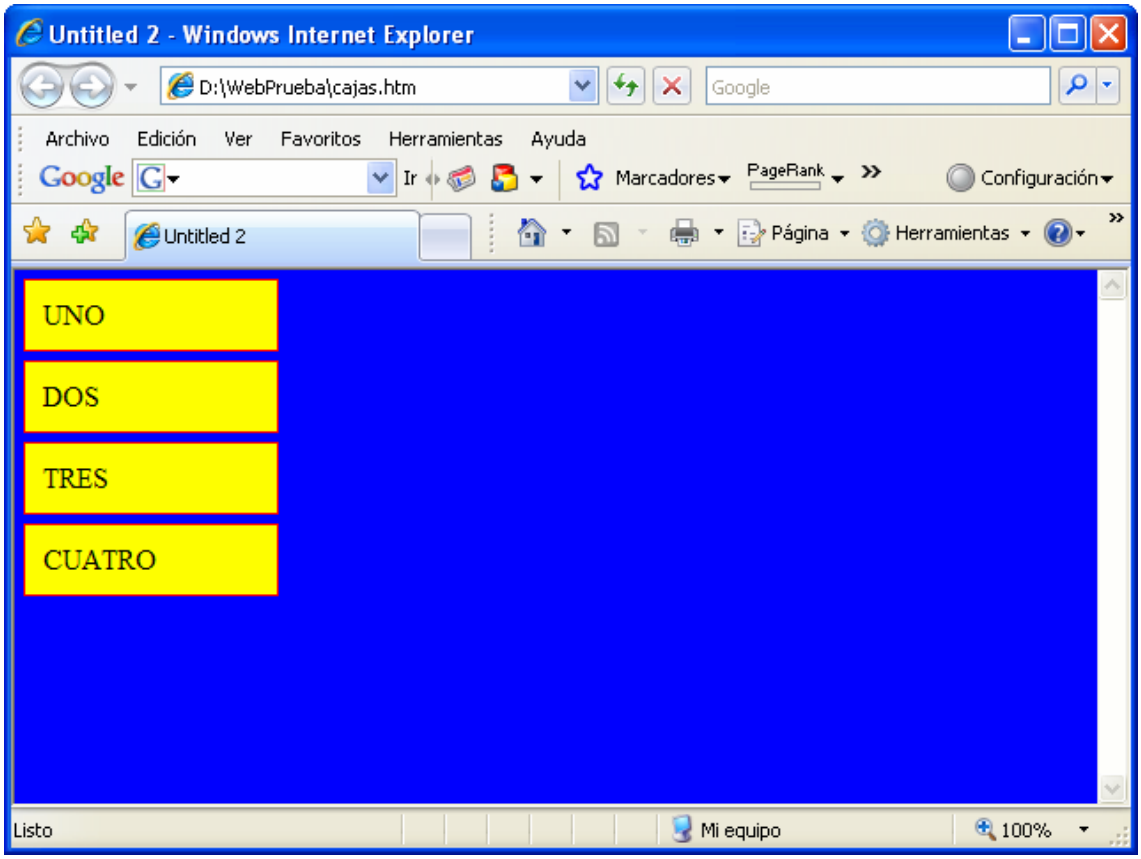
- Estático: `static`
- Relativo: `relative`
- Absoluto: `absolute`
- Fijo: `fixed`

5.2.1 POSICIONAMIENTO ESTATICO: static

Con el posicionamiento estático, los elementos fluyen como en un procesador de textos. **Las cajas son elementos un poco especiales porque ocupan siempre todo el ancho de la página** –aunque sus dimensiones sean menores al ancho- y "fluyen" como si fuesen otros elemento cualesquiera: esto tiene como resultado que las cajas se colocan cada una debajo de la anterior. **No es necesario declarar** este tipo de posicionamiento en nuestra hoja de estilos porque **el posicionamiento estático es el posicionamiento por defecto**. En todo caso, se escribe de la siguiente manera –aunque repito que no es necesario-:

```
body {
	margin: 0px;
	padding: 0px;
	color: black;
	background-color: blue;
}
.caja {
	position: static;
	padding: 10px;
	margin: 5px;
	background-color: yellow;
	border: 1px solid red;
	width:20%;
}
```

La visualización de la página del ejemplo anterior con esta hoja de estilos es:



Para explicar los demás tipos de posicionamiento voy a crear una clase diferente en la hoja de estilos:

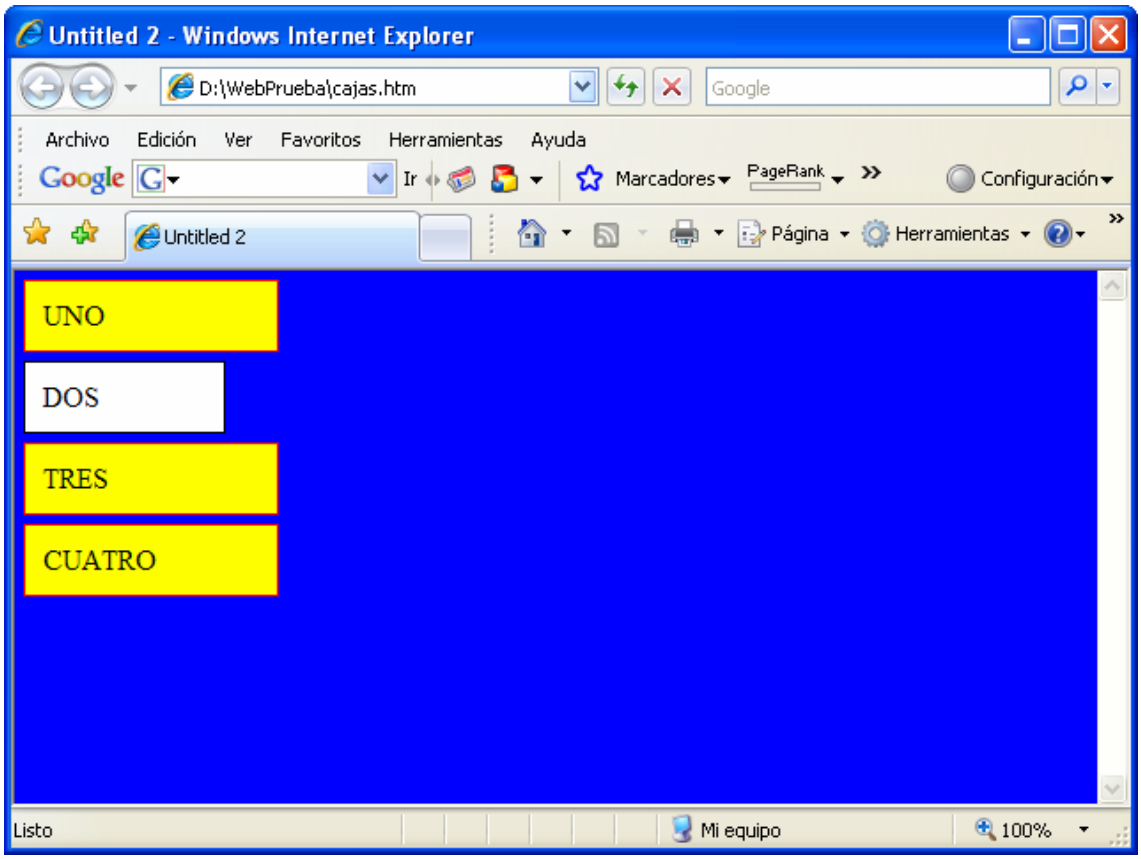
(Fichero `cajas.html`)

```
<body>
	<div class="caja">UNO</div>
	<div class="cajaDiferente">DOS</div>
	<div class="caja">TRES</div>
	<div class="caja">CUATRO</div>
</body>
```

(Fichero `cajas.css`)

```
body {
	margin: 0px;
	padding: 0px;
	color: black;
	background-color: blue;
}
.caja {
	position: static;
	padding: 10px;
	margin: 5px;
	background-color: yellow;
	border: 1px solid red;
	width:20%;
}
.cajaDiferente {
	padding: 10px;
	margin: 5px;
	background-color: white;
	border: 1px solid black;
	width:15%;
}
```

Se vería de la siguiente forma:



En este caso, como no se ha dicho lo contrario, el posicionamiento de la caja DOS también es estático de modo que se incluye en el flujo de la página como los demás elementos (al ser una caja ocupa todo el ancho de la página). Y además, como es el posicionamiento por defecto, el código siguiente tiene exactamente la misma representación:

Aunque ya se ha hecho una referencia anterior, un ejemplo parecido se explica en el video anterior llamado Posicionamiento `static`.

```
body {
	margin: 0px;
	padding: 0px;
	color: black;
	background-color: blue;
}
.caja {
	position: static;
	padding: 10px;
	margin: 5px;
	background-color: yellow;
	border: 1px solid red;
	width:20%;
}
.cajaDiferente {
	position:static;
	padding: 10px;
	margin: 5px;
	background-color: white;
	border: 1px solid black;
```

```
	width:15%;
}
```

Si añadiéramos la línea en negrita, es decir, si especificamos que el posicionamiento de la caja va a ser `static`, la visualización sería la misma porque es **el posicionamiento** `static` **es el posicionamiento por defecto y se puede omitir**.

5.2.2 POSICIONAMIENTO RELATIVO: relative

Si no se define ningún desplazamiento, los elementos con un posicionamiento relativo se ubican de la misma forma que si su posicionamiento fuese estático.

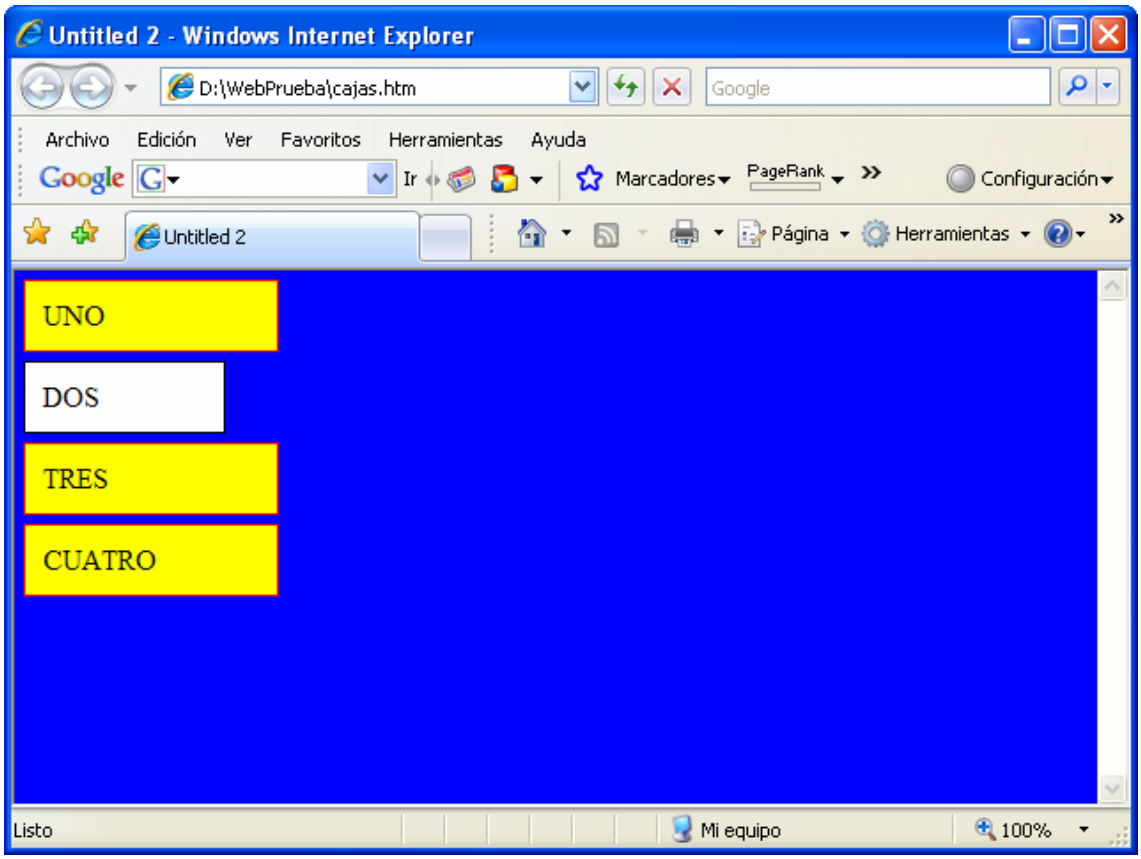
Así, la página:

```
<body>
	<div class="caja">UNO</div>
	<div class="cajaDiferente">DOS</div>
	<div class="caja">TRES</div>
	<div class="caja">CUATRO</div>
</body>
```

Con la hoja de estilos:

```
body {
	margin: 0px;
	padding: 0px;
	color: black;
	background-color: blue;
}
.caja {
	position: static;
	padding: 10px;
	margin: 5px;
	background-color: yellow;
	border: 1px solid red;
	width:20%;
}
.cajaDiferente {
	position:relative;
	padding: 10px;
	margin: 5px;
	background-color: white;
	border: 1px solid black;
	width:15%;
}
```

Se visualiza:

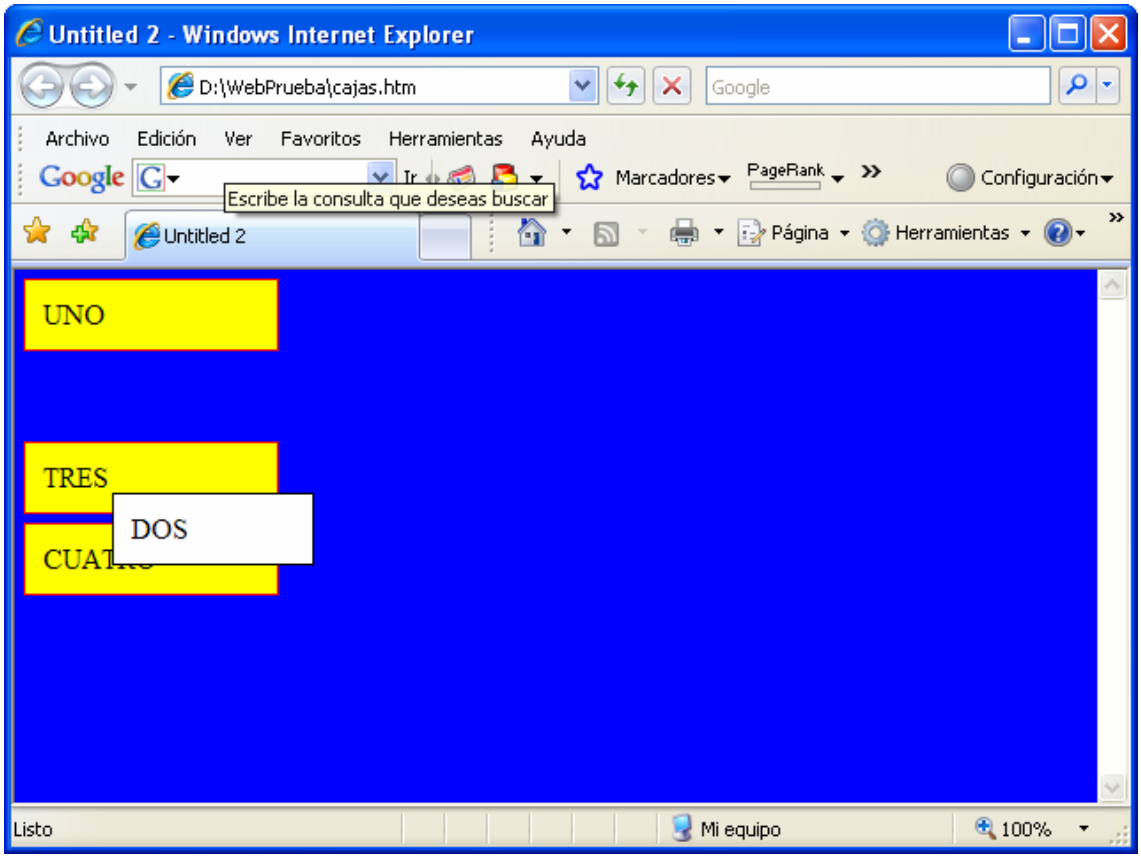


Sin embargo, con las propiedades, `top`, `rigth`, `bottom` y `left` se puede definir un desplazamiento para que el objeto **se desplace en esas cantidades desde la posición inicial que le corresponde**. Por ejemplo: si cambiamos el código de nuestra página de estilos a:

Veremos exactamente lo mismo que antes. Sin embargo,

```
body {
	margin: 0px;
	padding: 0px;
	color: black;
	background-color: blue;
}
.caja {
	position: static;
	padding: 10px;
	margin: 5px;
	background-color: yellow;
	border: 1px solid red;
	width:20%;
}
.cajaDiferente {
	position:relative;
	top:75px;
	left:50px;
	padding: 10px;
	margin: 5px;
	background-color: white;
	border: 1px solid black;
	width:15%;
}
```

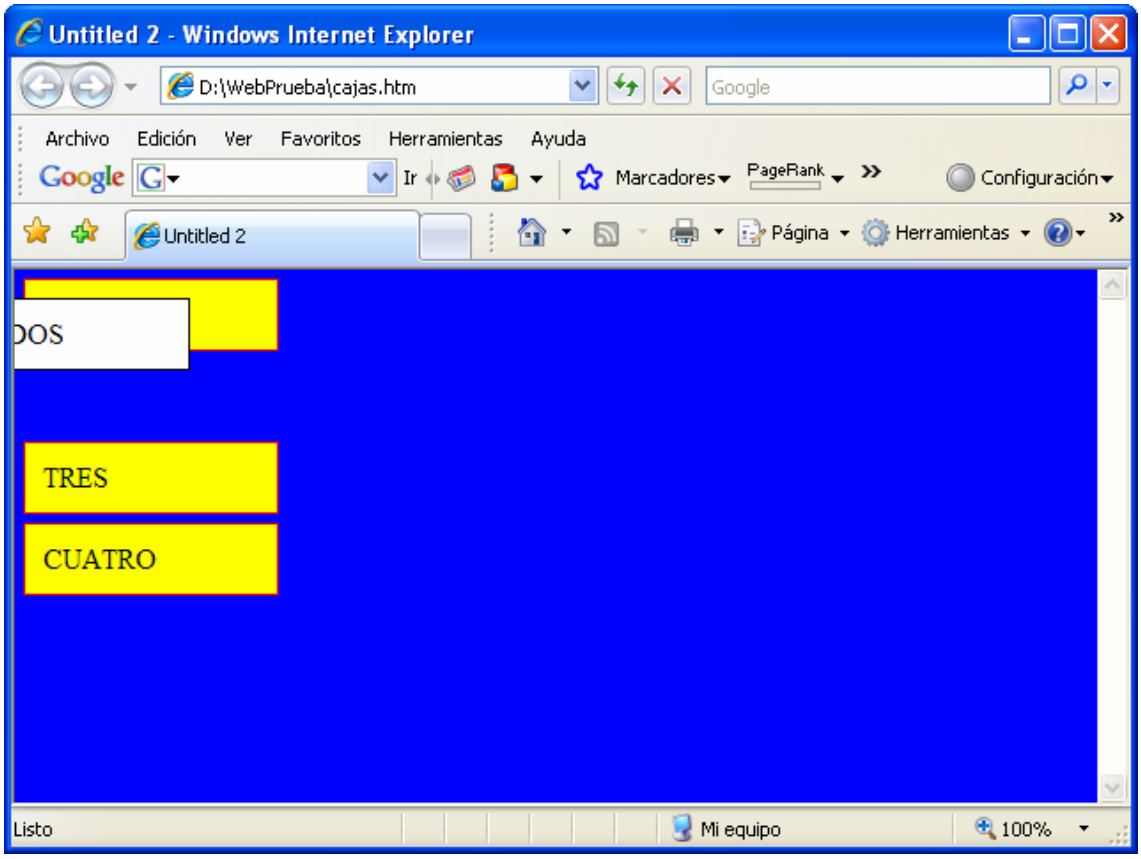
El navegador desplaza la caja blanca `75px` hacia abajo y `50px` hacia la derecha desde la posición inicial. Se ubica donde quede, aunque sea “encima” de otros elementos, que no por eso variarán su posición.



De la misma forma podemos utilizar cualquiera de las propiedades antes enumeradas. En el siguiente ejemplo, la caja DOS se desplaza 35px hacia arriba y 20px a la izda:

```
body {
	margin: 0px;
	padding: 0px;
	color: black;
	background-color: blue;
}
.caja {
	position: static;
	padding: 10px;
	margin: 5px;
	background-color: yellow;
	border: 1px solid red;
	width:20%;
}
.cajaDiferente {
	position:relative;
	bottom:35px;
	right:20px;
	padding: 10px;
	margin: 5px;
	background-color: white;
	border: 1px solid black;
```

```
	width:15%;
}
```



¿Qué está sucedido realmente? Pues sencillamente que **en el posicionamiento relativo el elemento se desplaza desde la posición que le correspondería a un posicionamiento estático las magnitudes que se declaran en el estilo** `-top` (hacia abajo), `rigth` (hacia la izquierda), `left` (hacia la derecha) y `bottom` (hacia arriba)-. Si en ese lugar hay otro elemento, se colocara encima de él.

En resumen: se colocan las cuatro cajas como si el posicionamiento fuese estático –también la que tiene posicionamiento relativo- y ésta última **se desplaza** desde esta posición lo que indiquen las propiedades `top`, `rigth`, `left` y `bottom`.

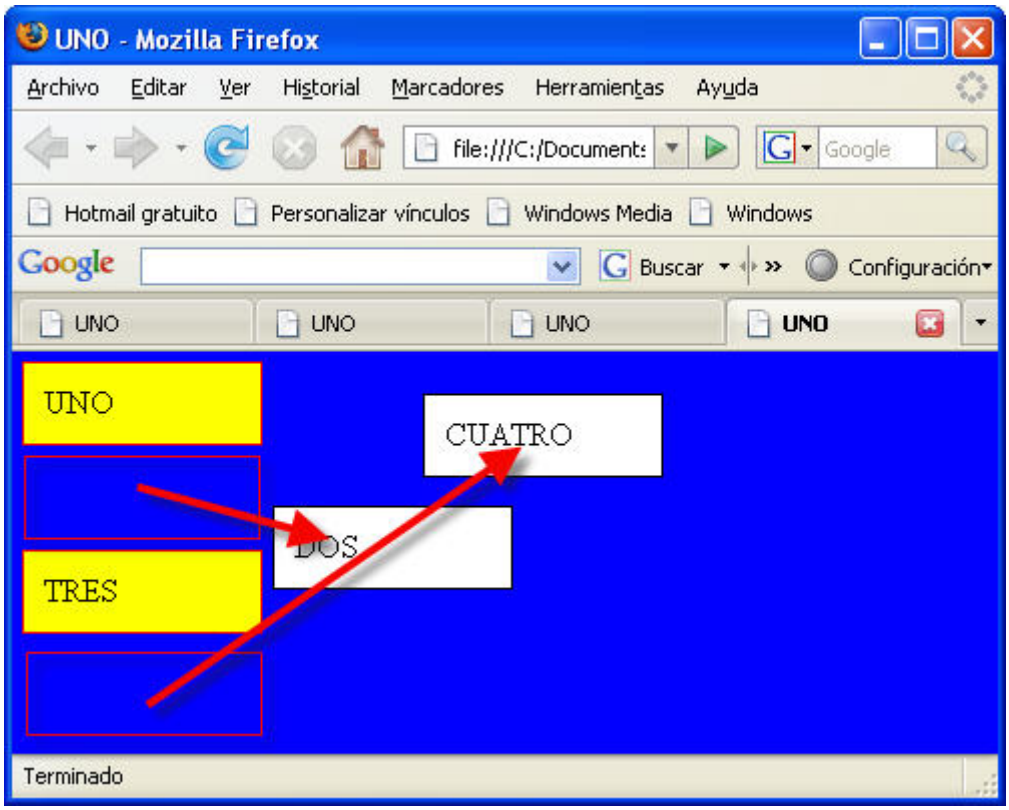
Para que quede un poco más claro, vamos a aplicar un posicionamiento relativo a las cajas segunda y cuarta. Si, por ejemplo, cambiamos esas propiedades en nuestra hoja de estilos a las siguientes:

```
<body>
	<div class="caja">UNO</div>
	<div class="cajaSegunda">DOS</div>
	<div class="caja">TRES</div>
	<div class="cajaCuarta">CUATRO</div>
</body>
```

```
body {
	margin: 0px;
```

```
	padding: 0px;
	color: black;
	background-color: blue;
}
.caja {
	position::static;
	padding: 10px;
	margin: 5px;
	background-color: yellow;
	border: 1px solid red;
	width:20%;
}
.cajaSegunda {
	position: relative;
	top: 25px;
	left: 125px;
	padding: 10px;
	margin: 5px;
	background-color: #FFFFFF;
	border: 1px solid #000000;
	width:20%;
}
.cajaCuarta {
	position: relative;
	bottom: 125px;
	left: 200px;
	padding: 10px;
	margin: 5px;
	background-color: #FFFFFF;
	border: 1px solid #000000;
	width:20%;
}
```

La página se vería en el explorador de la siguiente forma –por supuesto que las flechas y las cajas vacías se han dibujado posteriormente para intentar expresar que las cajas se desplazan desde la posición que les corresponderían con un posicionamiento estático-:



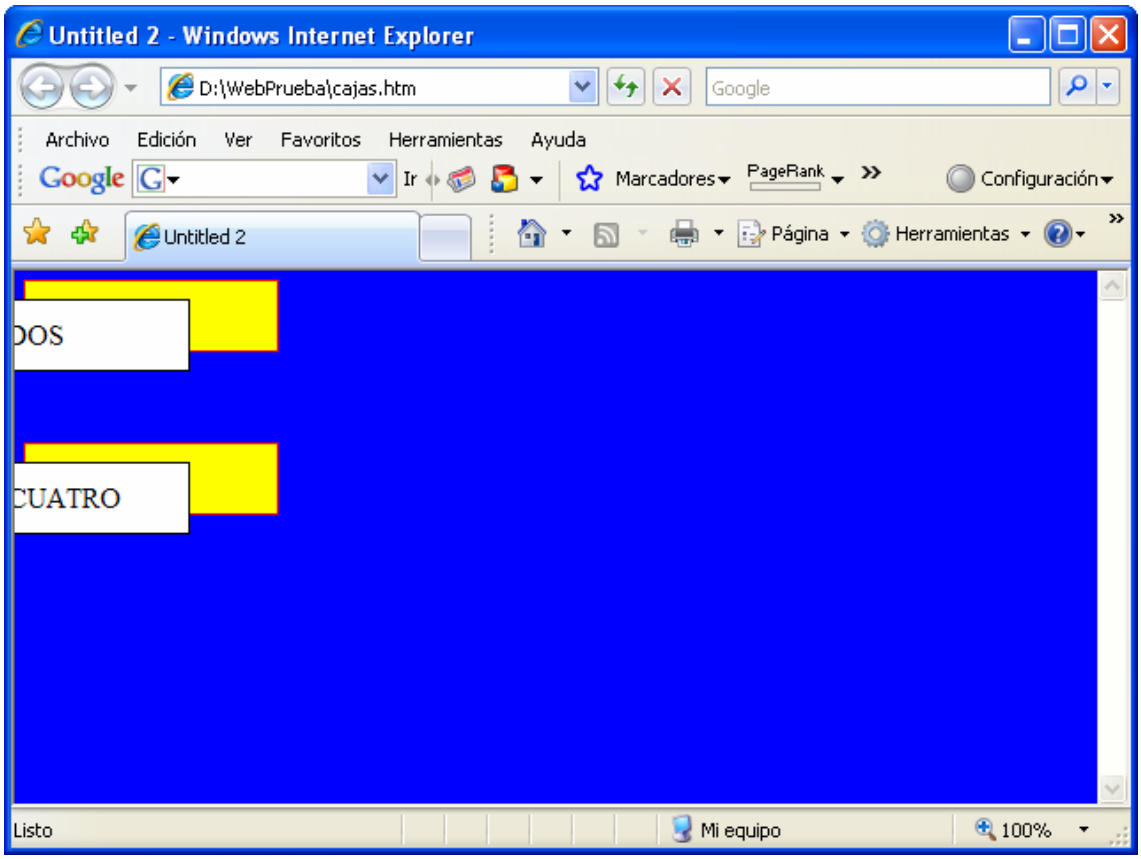
aaaa

Si aplicamos dicha clase a la caja CUATRO, las dos cajas se desplazaran lo mismo. Sería:

```
<body>
	<div class="caja">UNO</div>
	<div class="cajaDiferente">DOS</div>
	<div class="caja">TRES</div>
	<div class="cajaDiferente">CUATRO</div>
</body>
```

```
body {
	margin: 0px;
	padding: 0px;
	color: black;
	background-color: blue;
}
.caja {
	position: static;
	padding: 10px;
	margin: 5px;
	background-color: yellow;
	border: 1px solid red;
	width:20%;
}
.cajaDiferente {
	position:relative;
	bottom:35px;
	right:20px;
```

```
        padding: 10px;
        margin: 5px;
        background-color: white;
        border: 1px solid black;
        width:15%;
}
```



En el videotutorial Posicionamiento relative de 4m 45seg. de duración se explica la distribución relativa, que es igual que el `flow layout` o distribución en flujo pero en el que un elemento, después de ubicarse en la posición de flujo que le corresponde, se desplaza relativamente lo que indiquen sus propiedades `top`, `left`, `right` y `bottom`.

5.2.3 POSICIONAMIENTO ABSOLUTO: absolute

El posicionamiento absoluto elimina el flujo y coloca los elementos y las cajas a partir de la esquina superior izquierda en las posiciones marcadas por las propiedades: `top`, `rigth`, `bottom` y `left` . En el siguiente ejemplo, la caja DOS se desplaza `35px` hacia abajo y `100px` hacia la izquierda.

El resto de los elementos, se ubican como si los elementos con posicionamiento absoluto no existiesen.

Por ejemplo, si en la página htm anterior:

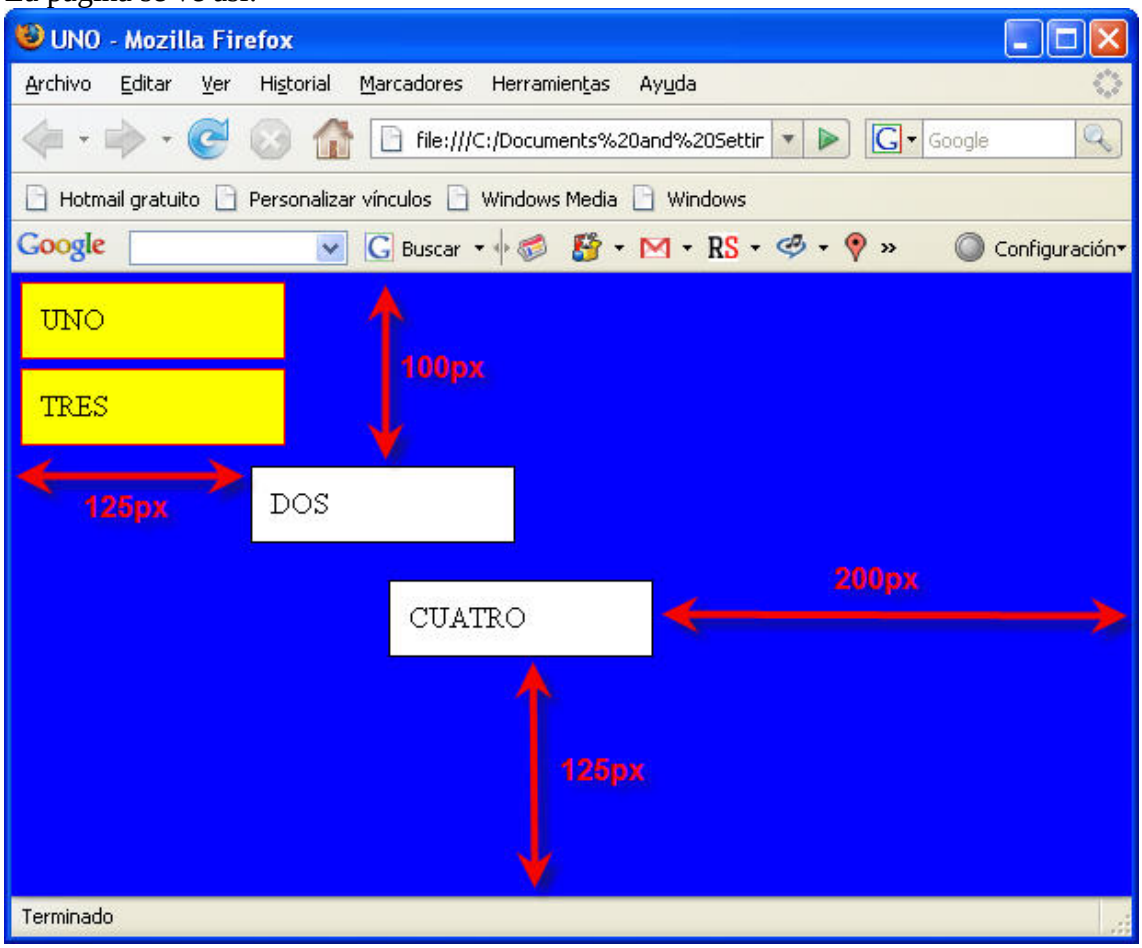
```
<body>
      <div class="caja">UNO</div>
      <div class="cajaSegunda">DOS</div>
      <div class="caja">TRES</div>
      <div class="cajaCuarta">CUATRO</div>
</body>
```

Le aplicamos una hoja de estilos parecida a la anterior pero donde se han posicionado las cajas 2 y 4 de manera absoluta:

```
body {
      margin: 0px;
      padding: 0px;
      color: black;
      background-color: blue;
}
.caja {
      position::static;
      padding: 10px;
      margin: 5px;
      background-color: yellow;
      border: 1px solid red;
      width:20%;
}
.cajaSegunda {
      position: absolute;
      top: 100px;
      left: 125px;
      padding: 10px;
      margin: 5px;
      background-color: #FFFFFF;
      border: 1px solid #000000;
      width:20%;
}
.cajaCuarta {
      position: absolute;
      bottom: 125px;
      left: 200px;
      padding: 10px;
      margin: 5px;
      background-color: #FFFFFF;
      border: 1px solid #000000;
```

```
        width:20%;
}
```

La página se ve así:



Es fácil ver que se han colocado las cajas 1 y 3 manera estática, como si no existiesen más elementos y la 2 y 4 se han colocado con un posicionamiento absoluto.

Dicho de otro modo, los elementos que tengan un posicionamiento absoluto, pierden su lugar en el “flujo”.

También es incesante destacar que cuando se trabaja con posicionamiento absoluto, al modificar las dimensiones de la ventana del explorador, las distancias se mantienen constantes. Por eso, si hemos definido que `right` es `125px`, significa que sea cual sea el tamaño, la distancia del lado derecho al objeto será siempre de `125px`. Lo mismo sucede con las otras propiedades.

Todo esto se verá mejor en el video Posicionamiento absolute de 3m 19seg. de duración. En este tipo de posicionamiento, cada elemento de coloca en una posición que corresponde a las coordenadas del elemento respecto a los lados del navegador. Esas coordenadas se definen por medio de las propiedades, `top`, `right`, `bottom` y `left`. El objeto se ubica en el lugar exacto de las coordenadas y su "sitio" correspondiente en la distribución de flujo se "salta", es decir, no se le tiene en cuenta.

5.2.4 POSICIONAMIENTO FIJO: fixed.

Si el posicionamiento de un elemento es `fixed` su ubicación es igual que el posicionamiento `absolute` pero con una diferencia fundamental: cuando se hace scroll sobre la página web, ese elemento `fixed` permanece en posición fija en la pantalla mientras que cualquier otro elemento se mueve con el scroll.

Este posicionamiento es un poco más complejo de entender aunque seguro que todos lo hemos visto en alguna página Web. Es igual que el absoluto, pero el elemento, cuando se hace scroll en la página, queda fijo en su posición.

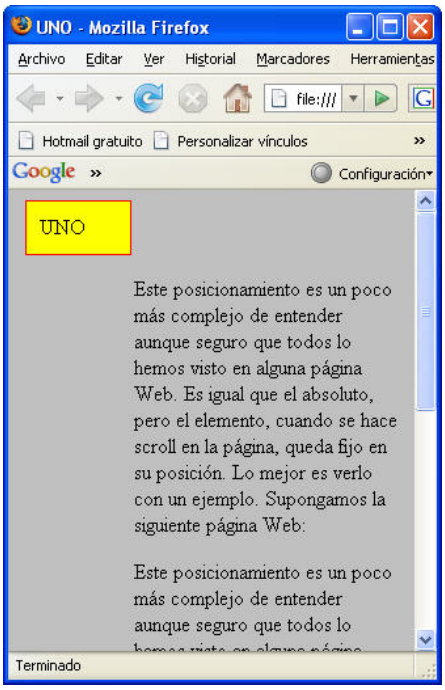
Lo mejor es verlo con un ejemplo. Supongamos la siguiente página Web:

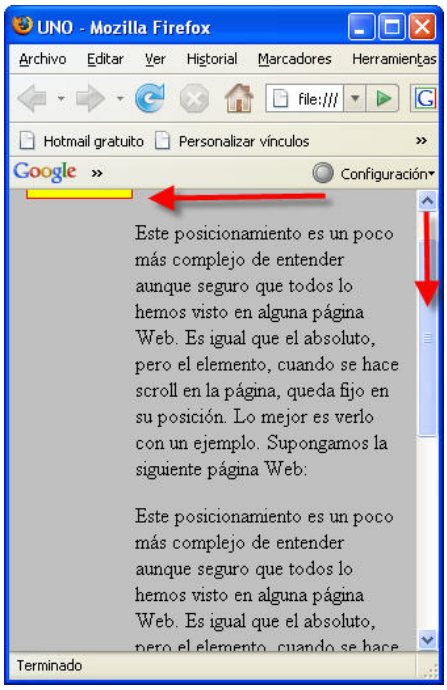
```
<body>
	<div class="caja">UNO</div>
	<p>Este posicionamiento es un poco más complejo de entender aunque seguro que
	todos lo hemos visto en alguna página Web. Es igual que el absoluto, pero el
	elemento, cuando se hace scroll en la página, queda fijo en su posición. Lo mejor
	es verlo con un ejemplo. Supongamos la siguiente página Web:
	</p>
	<p>Este posicionamiento es un poco más complejo de entender aunque seguro que
	todos lo hemos visto en alguna página Web. Es igual que el absoluto, pero el
	elemento, cuando se hace scroll en la página, queda fijo en su posición. Lo mejor
	es verlo con un ejemplo. Supongamos la siguiente página Web:
	</p>
	<p>Este posicionamiento es un poco más complejo de entender aunque seguro que
	todos lo hemos visto en alguna página Web. Es igual que el absoluto, pero el
	elemento, cuando se hace scroll en la página, queda fijo en su posición. Lo mejor
	es verlo con un ejemplo. Supongamos la siguiente página Web:
	</p>
</body>
```

Si a esta página, le asociamos la siguiente hoja de estilos:

```
body{
	background-color:silver;
}
p{
	margin-left:30%;
}
.caja {
	position:static;
	top:0px;
	left:0px;
	padding: 10px;
	margin: 5px;
	background-color: yellow;
	border: 1px solid red;
	width:20%;
}
```

La página se ve como en la figura de la izquierda.



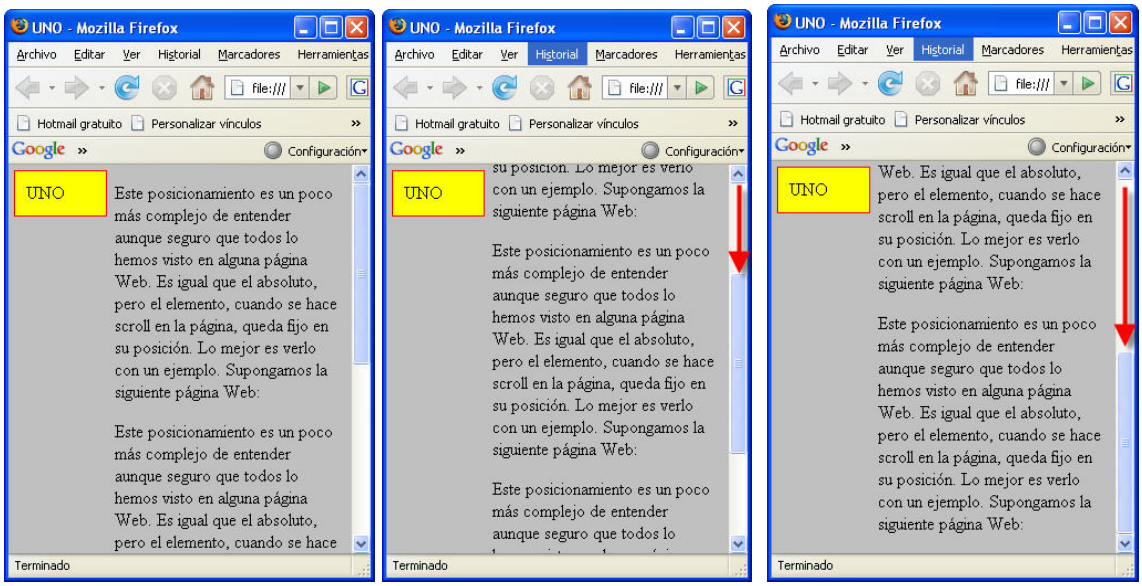


Si en esta ventana pulsamos sobre la barra de desplazamiento para hacer un scroll, el resultado es el que se representa en la figura de la izquierda. Vemos que se desplaza la caja y el texto.

Sin embargo, si variamos el posicionamiento de la caja a `fixed`,

```
body{
	background-color:silver;
}
p{
	margin-left:30%;
}
.caja {
	position:fixed;
	top:0px;
	left:0px;
	padding: 10px;
	margin: 5px;
	background-color: yellow;
	border: 1px solid red;
	width:20%;
}
```

tendremos:



Podemos observar en la secuencia que al hacer scroll, todo el contenido de la ventana se desplaza pero la caja permanece fija.

Un ejemplo parecido a éste se hace en el video Posicionamiento fixed de 3m 36seg. de duración. Se explica el posicionamiento fijo o `fixed`, que permite hacer scroll de una pantalla manteniendo elementos fijos.

Además se puede ver también el video DistribucionResumen que explica resumidamente los posicionamientos `static`, `relative`, `fixed` y `absolute`.

5.2.5 FLOAT

Antes de que hubiese un estándar CSS compatible, los diseñadores utilizaban el atributo `align` para que un texto fluyese alrededor de una imagen. El estándar recomienda ahora la utilización de `float` en su lugar.

`float` es una propiedad en sí misma, no un valor de la propiedad de posicionamiento, aunque de hecho, se utiliza para posicionar elementos en pantalla.

Un objeto con la propiedad `float` a `left`, hace que el contenido fluya a partir de ese objeto hacia el lado derecho y puesto a `rigth`, al revés.

Vamos a verlo con un ejemplo. Tenemos la página:

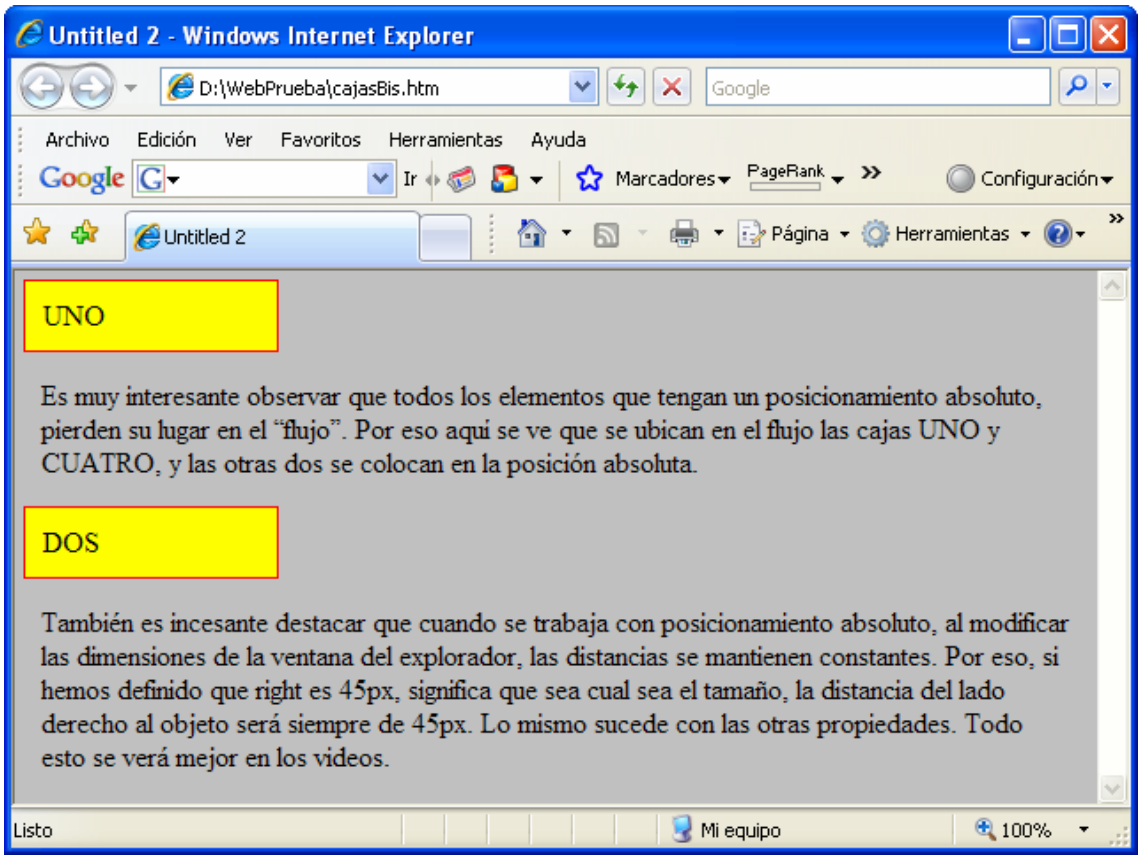
```
<body>

      <div class="cajaIzquierda">UNO</div>

      <p>Es muy interesante observar que todos los elementos que
      tengan un posicionamiento absoluto, pierden su lugar en el
      “flujo”. Por eso aquí se ve que se ubican en el flujo las cajas
      UNO y CUATRO, y las otras dos se colocan en la posición
      absoluta.</p>
```

```
	<div class="cajaDerecha">DOS</div>

	<p>También es incesante destacar que cuando se trabaja con posicionamiento absoluto, al modificar las dimensiones de la ventana del explorador, las distancias se mantienen constantes. Por eso, si hemos definido que right es 45px, significa que sea cual sea el tamaño, la distancia del lado derecho al objeto será siempre de 45px. Lo mismo sucede con las otras propiedades. Todo esto se verá mejor en los videos.</p>

</body>
```

En la hoja de estilos siguiente las dos clases son iguales y no se hace ninguna referencia a posicionamiento, por lo que será un flujo o distribución `static`, que es la que se proporciona por defecto y en donde cada objeto se sitúa en una nueva línea, ocupando toda la línea, aunque, de hecho tenga una anchura como para “caber” en la línea anterior.

```
body {
	margin: 0px;
	padding: 0px;
	color: black;
	background-color:silver;
}
p{
	margin:5px;
	padding:10px;
}
.cajaIzquierda {
	padding: 10px;
	margin: 5px;
	background-color: yellow;
	border: 1px solid red;
	width:20%;
}
.cajaDerecha {
	padding: 10px;
	margin: 5px;
	background-color: yellow;
	border: 1px solid red;
	width:20%;
}
```

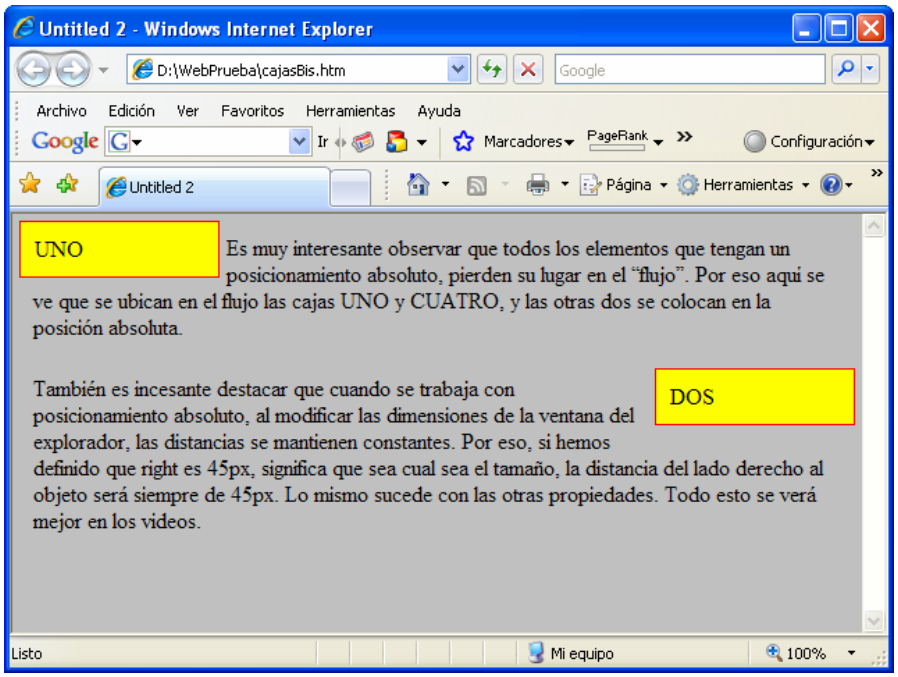
Hace que nuestra página se vea así:



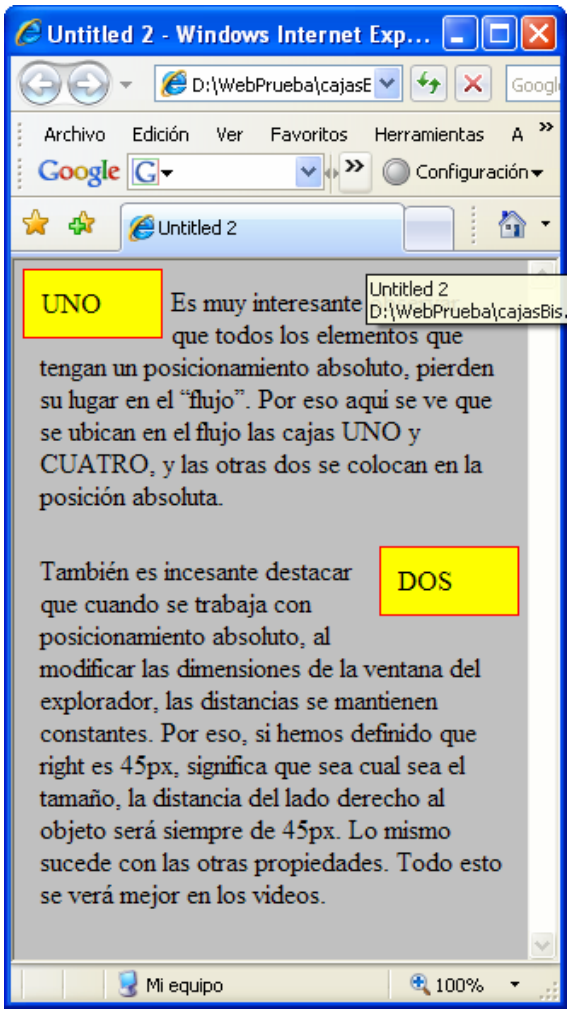
Cada caja y cada texto son un elemento y ocupan una nueva línea y la "agotan", es decir, cada nuevo elemento se ubica en una nueva línea. Si queremos que el texto fluya a la derecha y a la izquierda de las cajas, añadimos las líneas que están en negrita:

```
body {
	margin: 0px;
	padding: 0px;
	color: black;
	background-color:silver;
}
p{
	margin:5px;
	padding:10px;
}
.cajaIzquierda {
	float:left;
	padding: 10px;
	margin: 5px;
	background-color: yellow;
	border: 1px solid red;
	width:20%;
}
.cajaDerecha {
	float:right;
	padding: 10px;
	margin: 5px;
	background-color: yellow;
	border: 1px solid red;
	width:20%;
}
```

El resultado es el siguiente:



Modificando el tamaño del explorador obtenemos:



Como vemos, el texto fluye a la derecha y a la izquierda de las cajas.

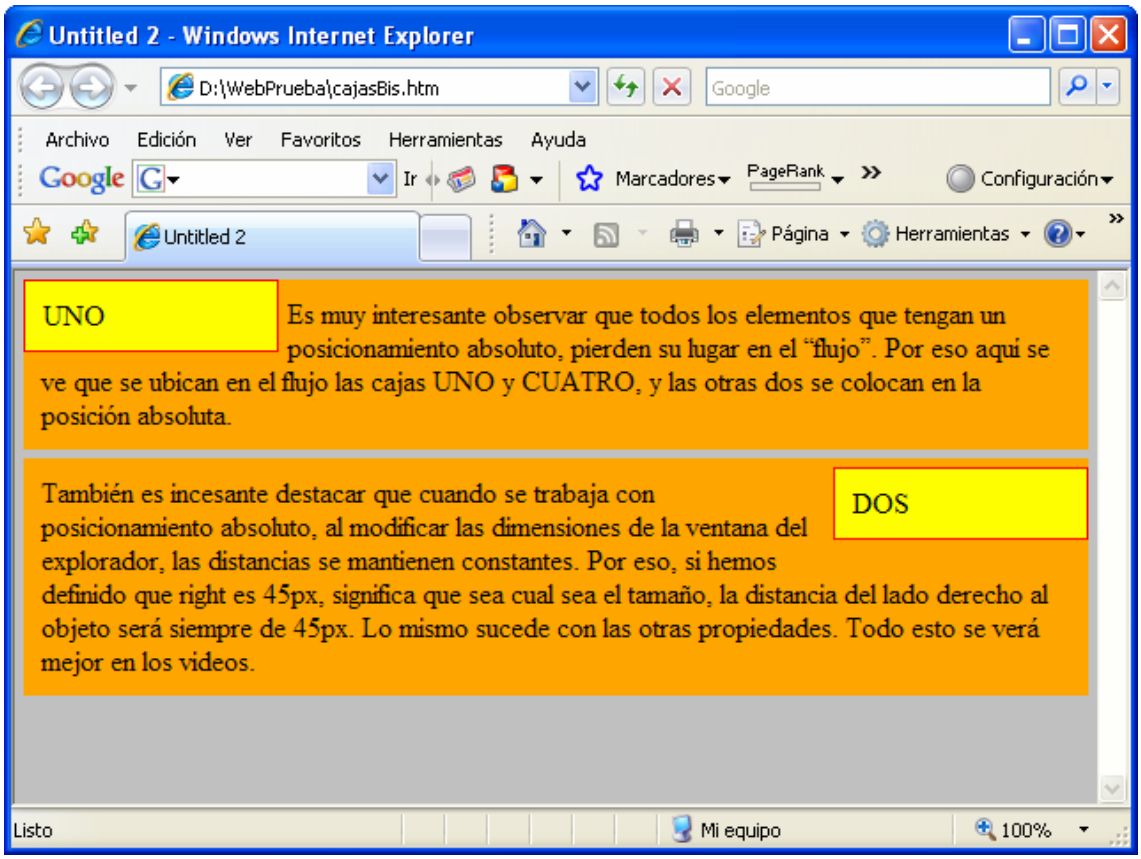
`float` nos da más control que `align` porque utiliza bordes, márgenes relleno y otras propiedades.

Hay un aspecto que es importante destacar aquí. Sólo el contenido del texto fluye alrededor de un objeto. Cualquier borde, imagen de fondo, o color de fondo se extiende bajo la imagen.

Para ver esto, vamos a poner un color naranja de fondo de los párrafos. Nuestra hoja de estilos queda ahora de la siguiente forma:

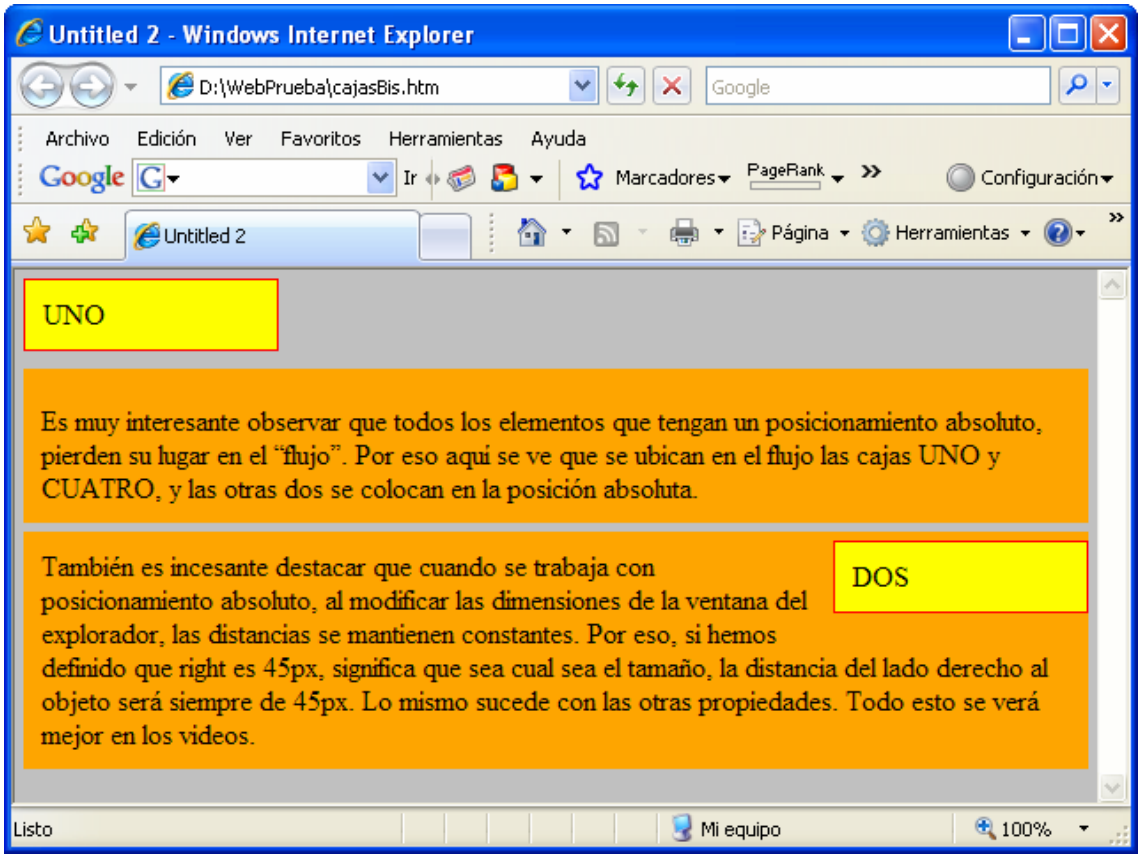
```
body {
	margin: 0px;
	padding: 0px;
	color: black;
	background-color:silver;
}
p{
	margin:5px;
	padding:10px;
	background-color:orange;
}
.cajaIzquierda {
	float:left;
	padding: 10px;
	margin: 5px;
	background-color: yellow;
	border: 1px solid red;
	width:20%;
}
.cajaDerecha {
	float:right;
	padding: 10px;
	margin: 5px;
	background-color: yellow;
	border: 1px solid red;
	width:20%;
}
```

La página se verá así:



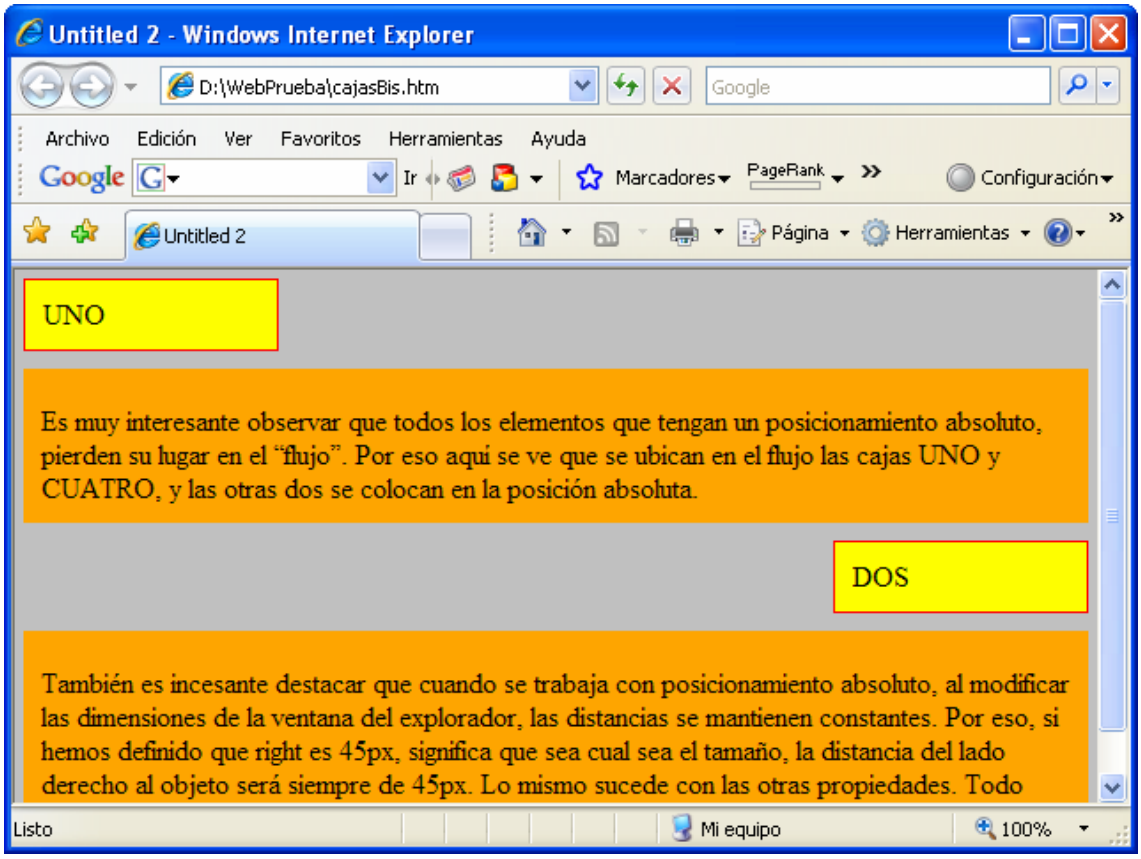
Si el comportamiento que resulta no es el deseado, entonces existe la opción de aplicar la propiedad `clear` al elemento que fluye, en este caso el párrafo

```
body {
	margin: 0px;
	padding: 0px;
	color: black;
	background-color:silver;
}
p{
	clear:left;
	margin:5px;
	padding:10px;
	background-color:orange;
}
.cajaIzquierda {
	float:left;
	padding: 10px;
	margin: 5px;
	background-color: yellow;
	border: 1px solid red;
	width:20%;
}
.cajaDerecha {
	float:right;
	padding: 10px;
	margin: 5px;
	background-color: yellow;
	border: 1px solid red;
	width:20%;
}
```



Podemos aplicar `clear` a ambos, izquierda y derecha:

```
body {
	margin: 0px;
	padding: 0px;
	color: black;
	background-color:silver;
}
p{
	clear:both;
	margin:5px;
	padding:10px;
	background-color:orange;
}
.cajaIzquierda {
	float:left;
	padding: 10px;
	margin: 5px;
	background-color: yellow;
	border: 1px solid red;
	width:20%;
}
.cajaDerecha {
	float:right;
	padding: 10px;
	margin: 5px;
	background-color: yellow;
	border: 1px solid red;
	width:20%;
}
```



Este ejemplo se explica en el videotutorial Posicionamiento float de 9m 22seg. de duración

5.2.6 MÁS DE POSICIONAMIENTO

Vamos ahora a hacer que fluyan los elementos, en nuestro caso las cajas.

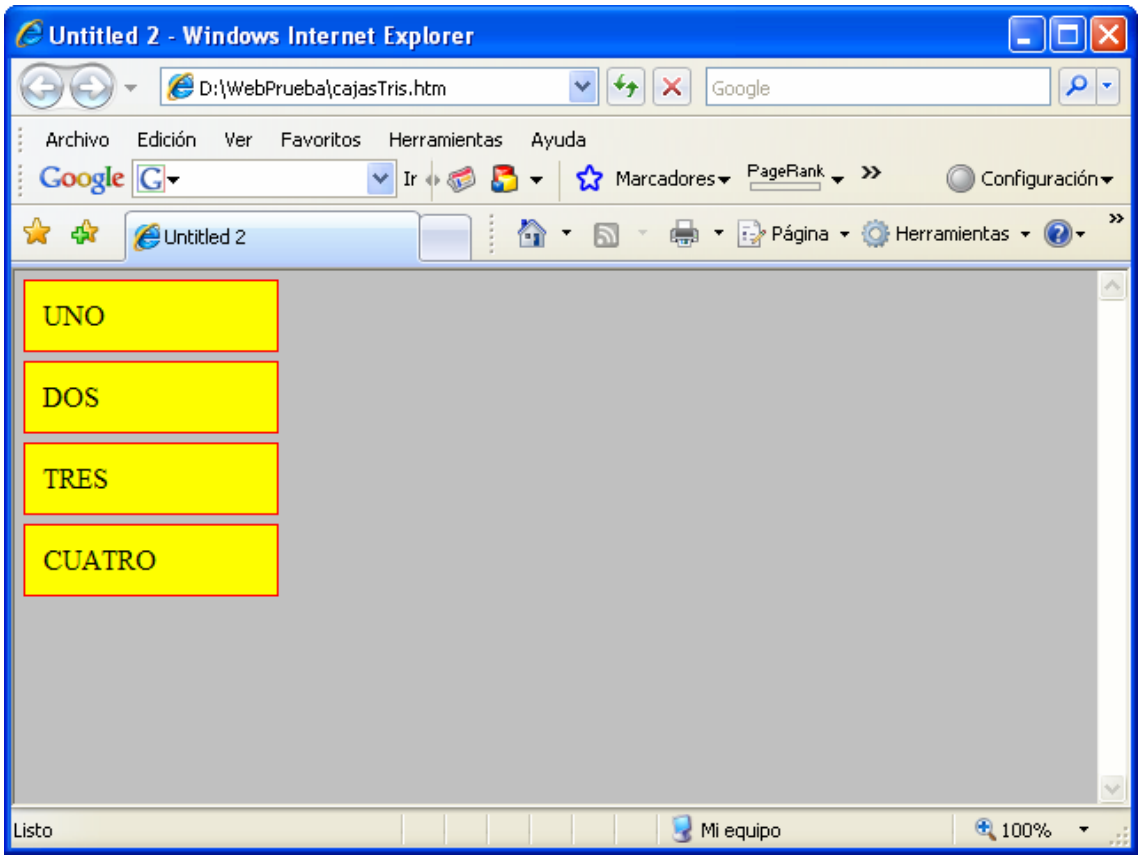
Si tenemos la página:

```
<body>

      <div class="caja">UNO</div>
      <div class="caja">DOS</div>
      <div class="caja">TRES</div>
      <div class="caja">CUATRO</div>


</body>
```

Con la hoja de estilos

```
body {
      margin: 0px;
      padding: 0px;
      color: black;
      background-color:silver;
}

.caja {
      padding: 10px;
      margin: 5px;
      background-color: yellow;
      border: 1px solid red;
      width:20%;
}
```

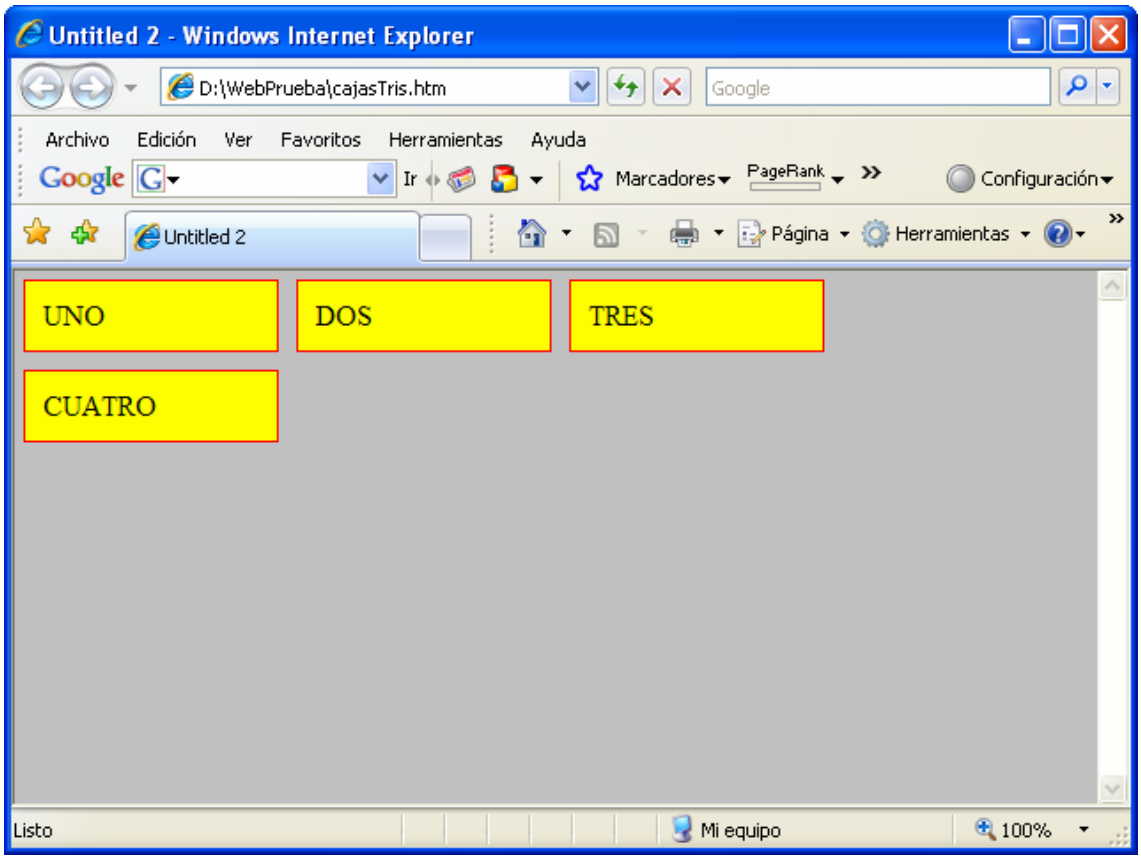
La representación, como sabemos es:



Si ahora añadimos una línea:

```
body {
	margin: 0px;
	padding: 0px;
	color: black;
	background-color:silver;
}

.caja {
	float:left;
	padding: 10px;
	margin: 5px;
	background-color: yellow;
	border: 1px solid red;
	width:20%;
}
```

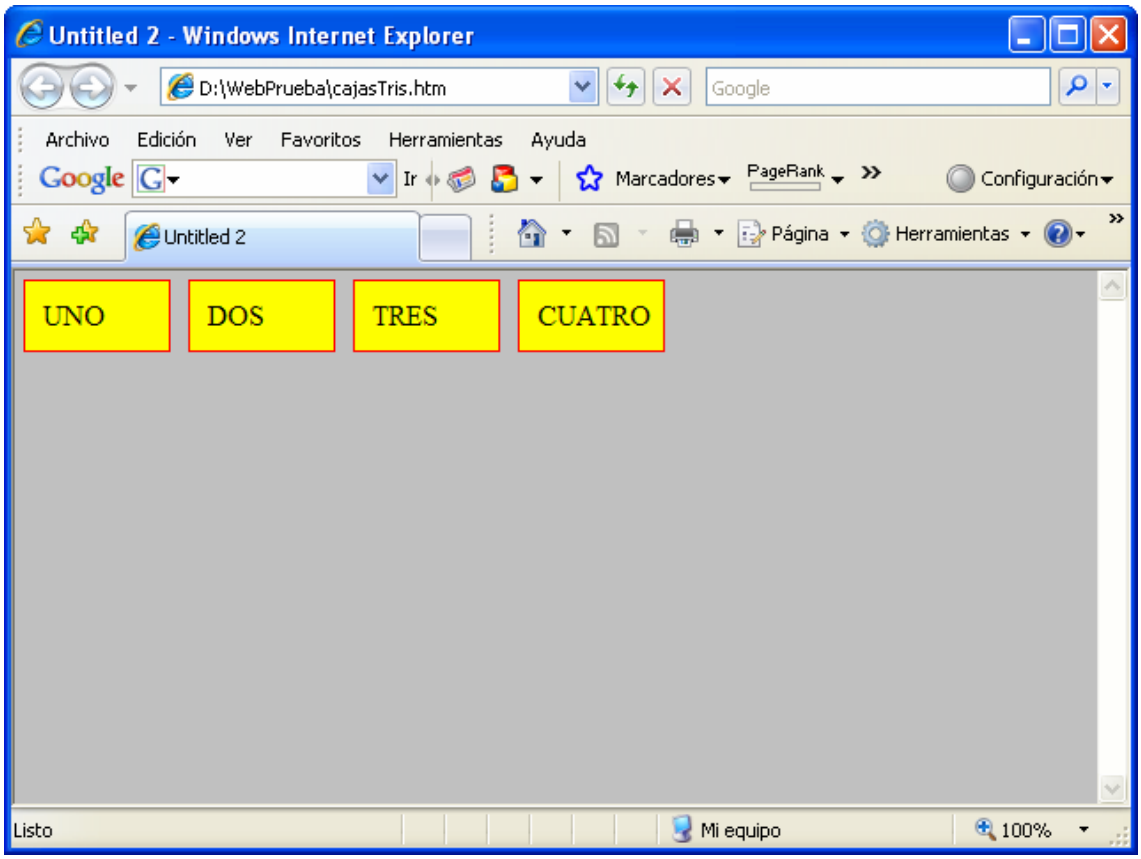
Se vería así:



Si disminuimos un poco la anchura de los elementos:

```
body {
	margin: 0px;
	padding: 0px;
	color: black;
	background-color:silver;
}

.caja {
	float:left;
	padding: 10px;
	margin: 5px;
	background-color: yellow;
	border: 1px solid red;
	width:10%;
}
```

Se ve así:

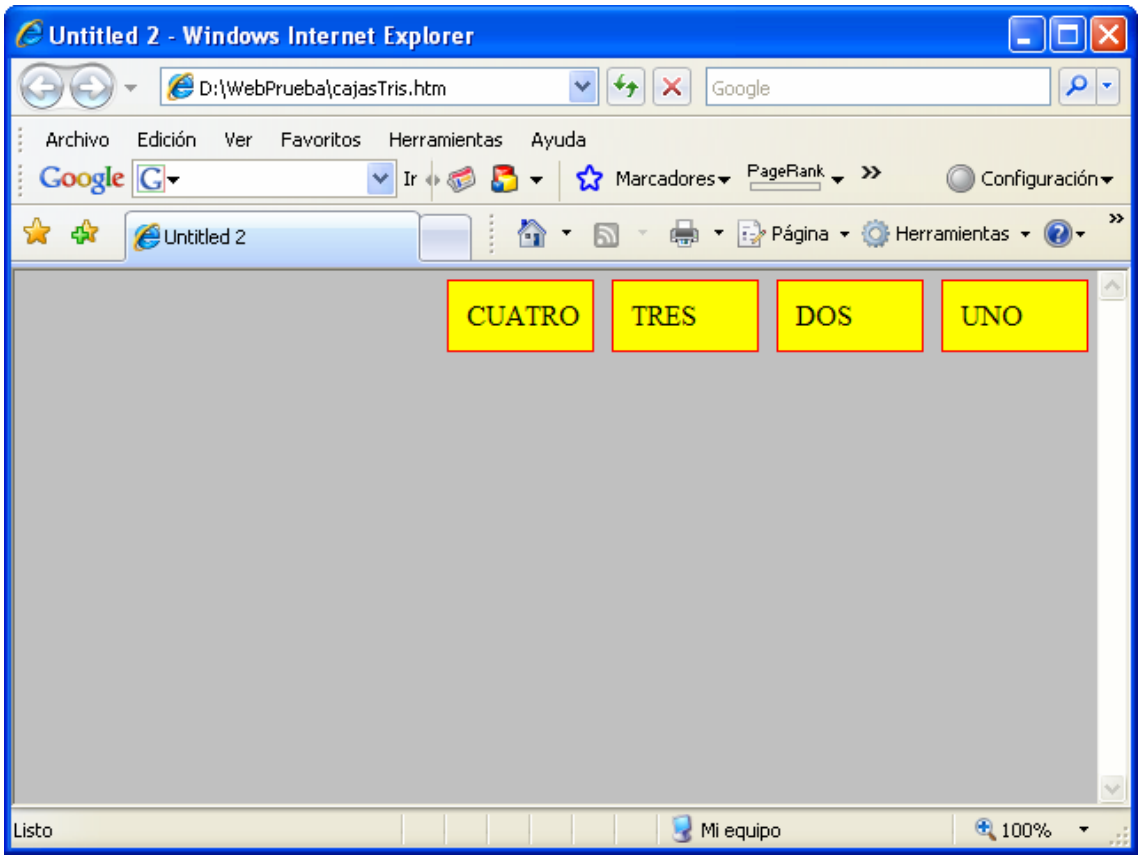


Las cajas se van colocando como las letras en un procesador de textos. Cuando ya no hay espacio para la siguiente caja en esa línea, se pasa a la línea inmediatamente inferior.

Y si cambiamos el `float` de las cajas a la derecha:

```
body {
	margin: 0px;
	padding: 0px;
	color: black;
	background-color:silver;
}

.caja {
	float:right;
	padding: 10px;
	margin: 5px;
	background-color: yellow;
	border: 1px solid red;
	width:10%;
}
```

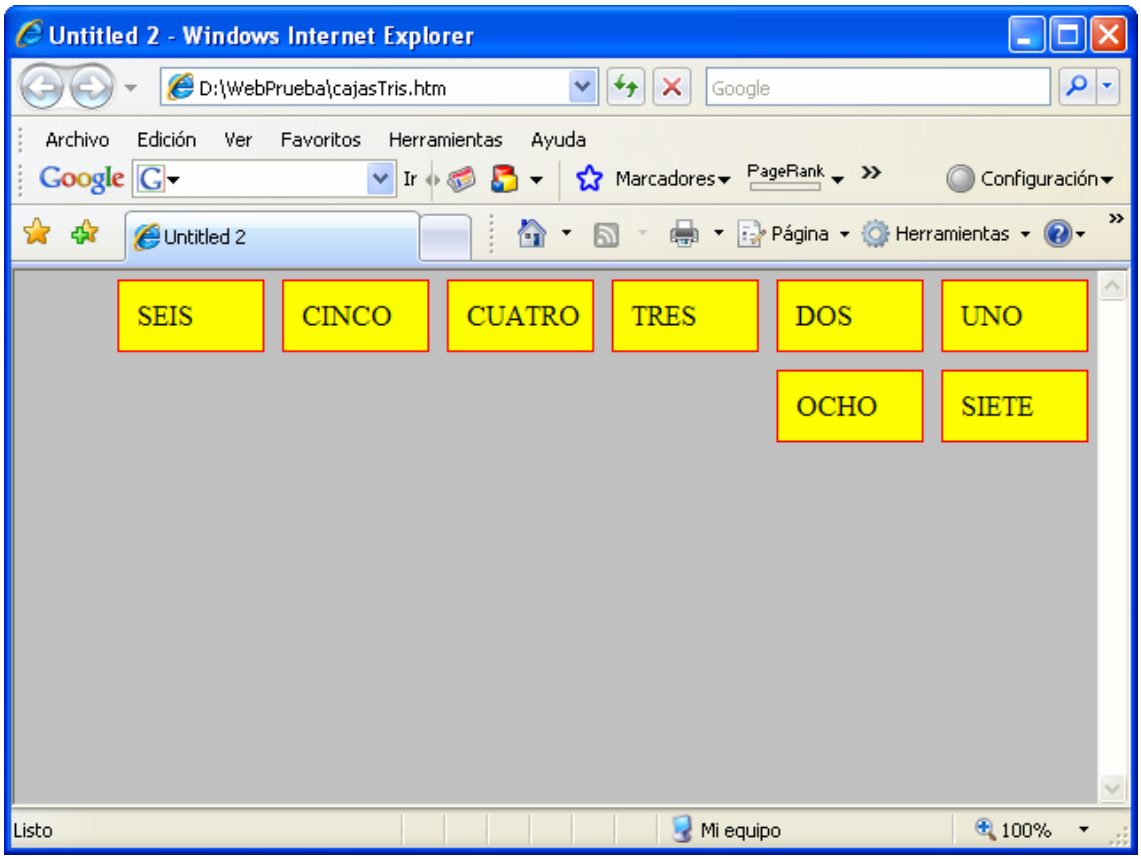
Se ve así:



Ahora sucede lo mismo, pero la primera caja se coloca tan alto como puede en la parte derecha y desde esa posición va fluyendo hacia la izquierda. Cuando acaba una línea y no caben más elemento se pasa a la siguiente línea comenzando de nuevo por la derecha.

Lo veremos mejor si cambiamos un poco la página para poner más elementos:

```
<body>

	<div class="caja">UNO</div>
	<div class="caja">DOS</div>
	<div class="caja">TRES</div>
	<div class="caja">CUATRO</div>
	<div class="caja">CINCO</div>
	<div class="caja">SEIS</div>
	<div class="caja">SIETE</div>
	<div class="caja">OCHO</div>
</body>
```



Exagerando un poco más las cosas, poniendo una hoja de estilos donde se han cambiado la anchura de las cajas a `60px` y estrechando mucho la anchura del explorador tenemos:

```
body {
	margin: 0px;
	padding: 0px;
	color: black;
	background-color:silver;
}

.caja {
	float:right;
	padding: 10px;
	margin: 5px;
	background-color: yellow;
	border: 1px solid red;
	width:60px;
}
```

Este ejemplo se explica en el videotutorial Posicionamiento float II. de 7m 11seg. de duración. Es continuación del anterior y se siguen explicando algunos aspectos de la propiedad `float`.

5.3 INDICE Z

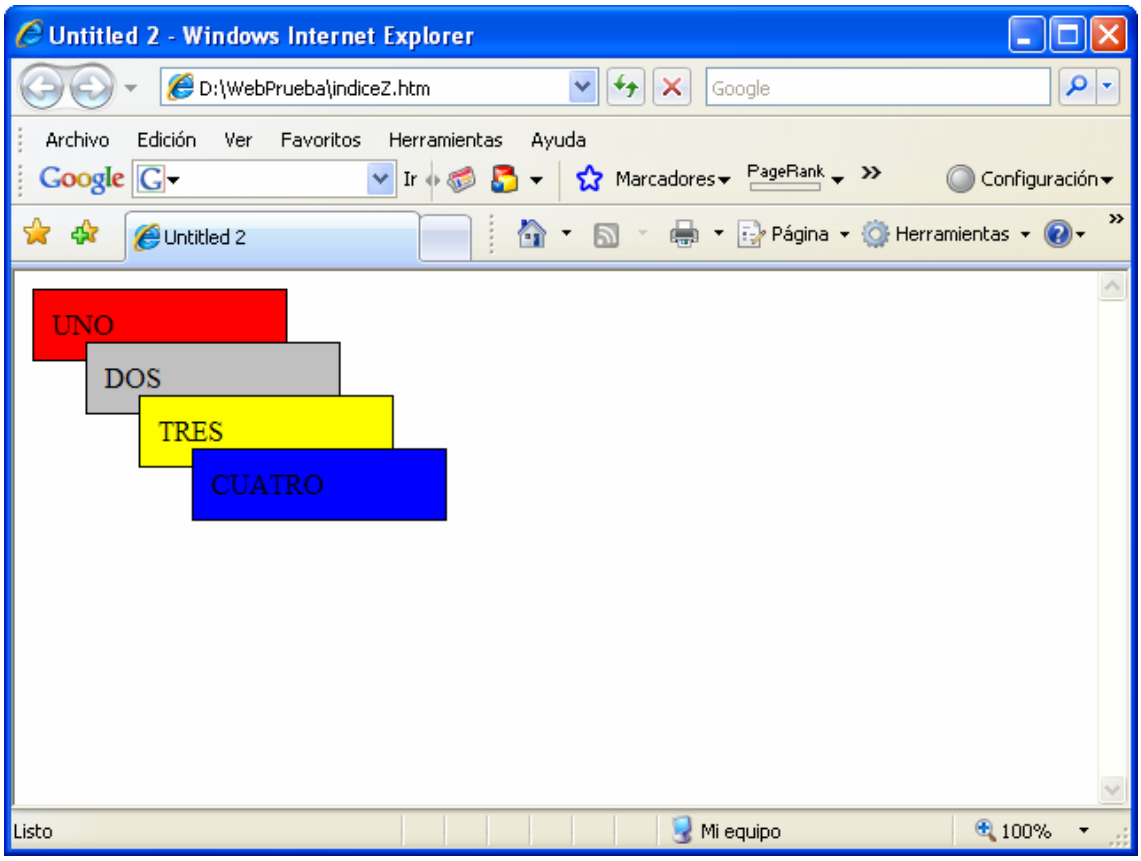
Las CSS permiten definir cómo estarán posicionados los elementos, a la izquierda, a la derecha, arriba o abajo en la pantalla y también permiten controlar qué objetos estarán por encima o por debajo de otros, es decir, permiten controlar el orden de colocación de los elemento “en vertical. Sería la tercera dimensión, la variable z.

La idea básica es muy sencilla: un elemento con una variable z grande está por debajo – estará tapado- por otro elemento con z más pequeña. En el siguiente ejemplo, se crean varias cajas posicionadas de manera absoluta para que se superpongan. Cada elemento `<div>` está posicionado de manera absoluta con referencia a la esquina superior izquierda de su elemento contenedor, que en este caso es `<html>`

```
<body>
<div id="caja1">UNO</div>
<div id="caja2">DOS</div>
<div id="caja3">TRES</div>
<div id="caja4">CUATRO</div>
</body>


#caja1 {
      position: absolute;
      top: 10px;
      left: 10px;
      padding: 10px;
      background-color: red;
      border: 1px solid black;
      width: 20%;

}
#caja2 {
      position: absolute;
      top: 40px;
      left: 40px;
      padding: 10px;
      background-color:silver;
      border: 1px solid black;
      width: 20%;
}
#caja3 {
      position: absolute;
      top: 70px;
      left: 70px;
      padding: 10px;
      background-color: yellow;
      border: 1px solid black;
      width: 20%;
}
#caja4 {
      position: absolute;
      top: 100px;
      left: 100px;
      padding: 10px;
      background-color: blue;
      border: 1px solid black;
```

```
        width: 20%;
}
```

Esta página se ve así:



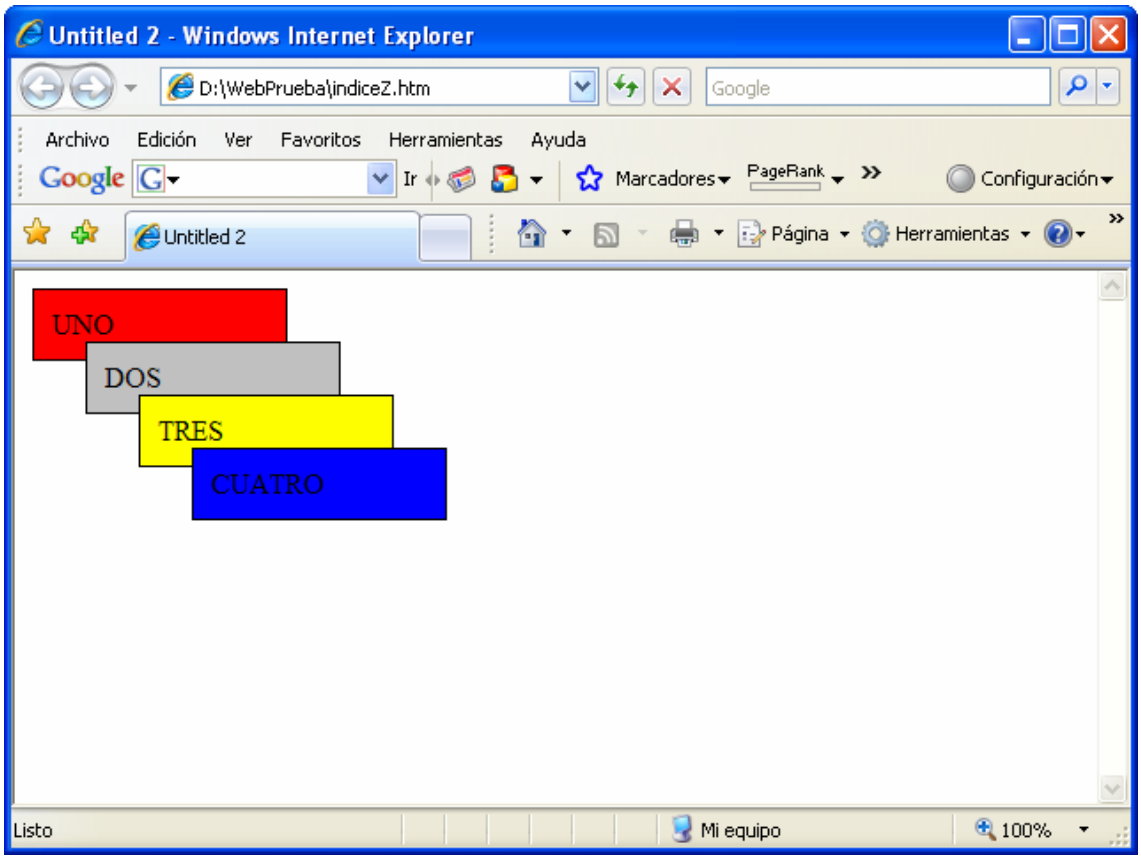
Los elementos se colocan en el orden en que vienen escritos en la página html, ya que todos los elemento pertenecen al mismo elemento contenedor que es `<html>`. En este caso, no hay ningún índice z que controle su posición.

La propiedad `z-index` es una propiedad que tiene un objeto para indicar al navegador en que posición está respecto a los demás elementos. Piense en una baraja de cartas: **el elemento con índice z-index más bajo estarán por debajo de los elemento con este índice más elevado**. Si dos elemento tienen el mismo índice z-index, entonces se colocarán en el orden en que esté escrito su código en la página Web, es decir, el primero que se escriba estará por debajo de los demás. Si hay un elemento que tiene definido un índice z-index positivo y el resto de los elementos no tiene definido un z-index, el primero estará por encima de los demás, que estarán, a su vez, colocados por el orden por defecto anteriormente descrito. Si ese índice es negativo, el elemento se posicionará por debajo del resto. Sin embargo, aunque está permitido, mi consejo es que **no se dé nunca valores negativos al z-index de un elemento** porque en algunos navegadores puede dar problemas. Por ejemplo, en Mozilla un elemento con z-index negativo lo coloca por debajo del canvas y puede que no se vea.

Vamos a añadir esa propiedad a los identificadores de las cajas del anterior ejemplo. Si queremos que se vean en el mismo orden que antes daremos a los que está situados más abajo, un número más pequeño: recuerde: a más pequeño, más abajo. La primera, la roja debe tener el índice menor.

```
#caja1 {
	position: absolute;
	top: 10px;
	left: 10px;
	padding: 10px;
	background-color: red;
	border: 1px solid black;
	width: 20%;
	z-index: 1;
}
#caja2 {
	position: absolute;
	top: 40px;
	left: 40px;
	padding: 10px;
	background-color:silver;
	border: 1px solid black;
	width: 20%;
	z-index: 2;
}
#caja3 {
	position: absolute;
	top: 70px;
	left: 70px;
	padding: 10px;
	background-color: yellow;
	border: 1px solid black;
	width: 20%;
	z-index: 3;
}
#caja4 {
	position: absolute;
	top: 100px;
	left: 100px;
	padding: 10px;
	background-color: blue;
	border: 1px solid black;
	width: 20%;
	z-index: 4;
}
```

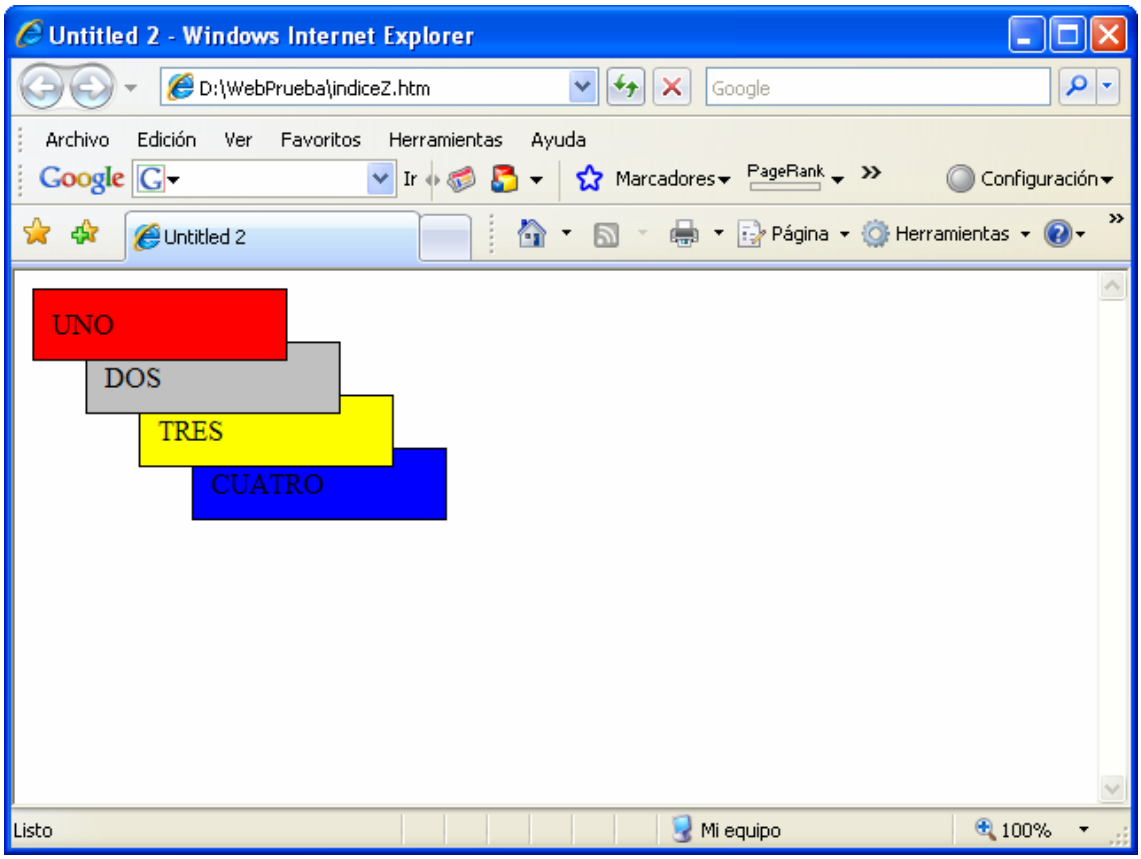
Se verá así:



Si deseamos que estén situados, de abajo arriba en el orden CUATRO, TRES, DOS, UNO, les daremos los índices 1, 2,3 y 4 respectivamente.

```
#caja1 {
	position: absolute;
	top: 10px;
	left: 10px;
	padding: 10px;
	background-color: red;
	border: 1px solid black;
	width: 20%;
	z-index: 4;
}
#caja2 {
	position: absolute;
	top: 40px;
	left: 40px;
	padding: 10px;
	background-color:silver;
	border: 1px solid black;
	width: 20%;
	z-index: 3;
}
#caja3 {
	position: absolute;
	top: 70px;
	left: 70px;
	padding: 10px;
	background-color: yellow;
```

```
	border: 1px solid black;
	width: 20%;
	z-index: 2;
}
#caja4 {
	position: absolute;
	top: 100px;
	left: 100px;
	padding: 10px;
	background-color: blue;
	border: 1px solid black;
	width: 20%;
	z-index: 1;
}
```



Una advertencia. Los índices no tienen porqué ir seguidos. En el anterior ejemplo, si hubiésemos dado los valores 4, 34, 567 y 1000 respectivamente al z-idex de los cuatro elementos la página Web se vería exactamente igual.

Este ejemplo de explica en el videotutorial Propiedad z-index de 8m 32seg. de duración. Se representan 4 bloques de distinto color para explicar el funcionamiento de la propiedad.

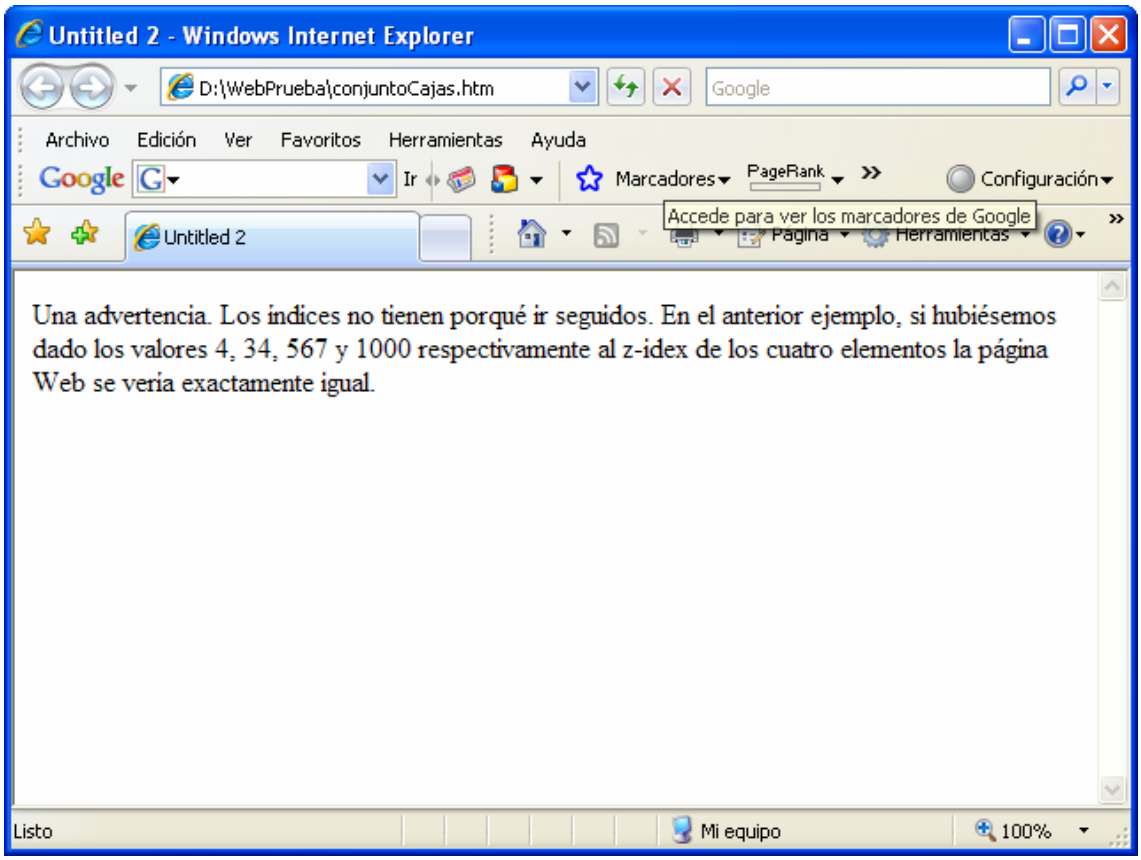
5.4 MÁS POSICIONAMIENTO: CÓMO MANEJAR CONJUNTOS DE BLOQUES O CAJAS

5.4.1 Dos filas

Consideremos la página Web siguiente:

```
<body>
      <div>Una advertencia. Los índices no tienen porqué ir seguidos.
      En el anterior ejemplo, si hubiésemos dado los valores 4, 34,
      567 y 1000 respectivamente al z-index de los cuatro elementos la
      página Web se vería exactamente igual.</div>
</body>
```

Esta página, como no tiene ningún estilo aplicado, se ve de la siguiente manera:

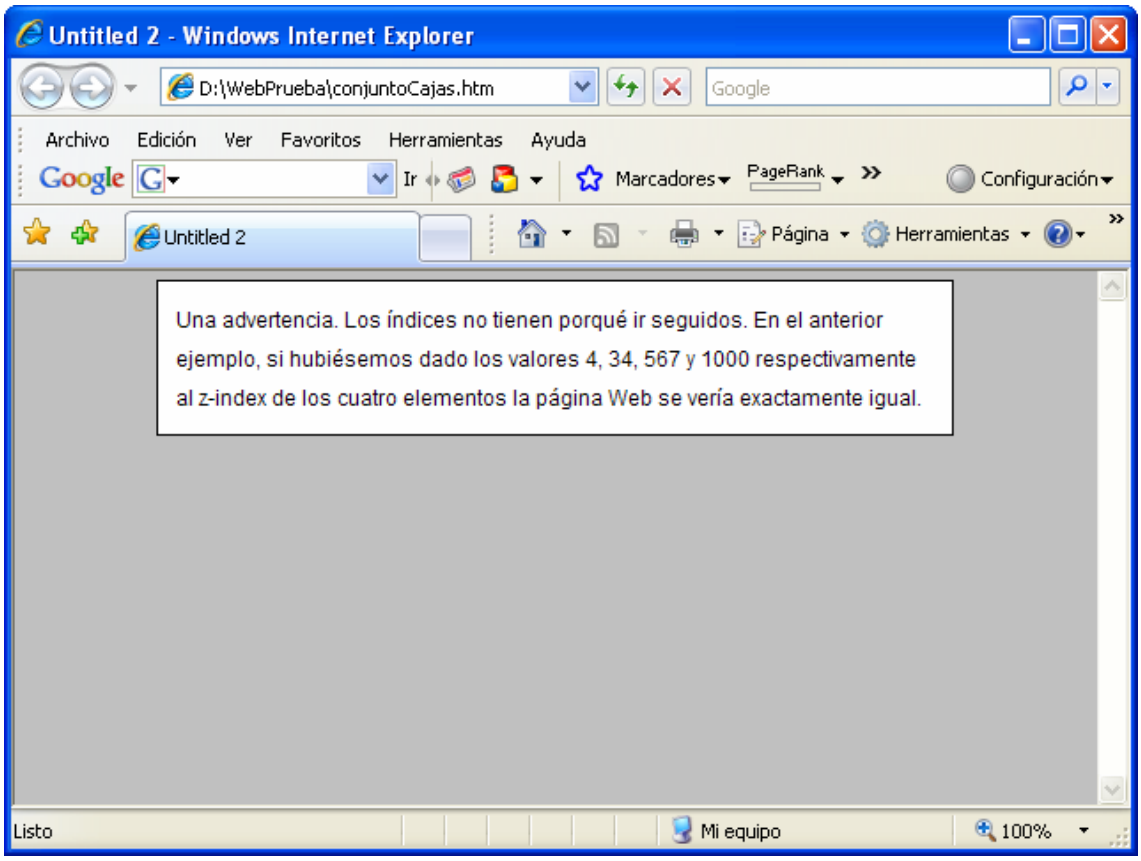


Vamos a definir una hoja de estilos externa. Queremos dar un fondo gris a la página y mostrar el texto en una caja de fondo blanco y centrada. Añadimos un atributo id al bloque y consideramos la hoja de estilos siguiente:

```
<body>
      <div id="caja">Una advertencia. Los índices no tienen porqué ir
      seguidos. En el anterior ejemplo, si hubiésemos dado los valores
      4, 34, 567 y 1000 respectivamente al z-idex de los cuatro
      elementos la página Web se vería exactamente igual.</div>
</body>
```

```
body {
	margin: 0px;
	padding: 0px;
	font-family: Arial, Helvetica, sans-serif;
	font-size: 12px;
	line-height: 22px;
	color: black;
	background-color: silver;
}
#caja {
	padding: 10px;
	margin: 5px auto 5px auto;
	background-color: white;
	border: 1px solid black;
	width: 70%;
}
```

Nuestra página se ve así:



Ahora vamos a presentar dos filas. Consideremos la página web siguiente:

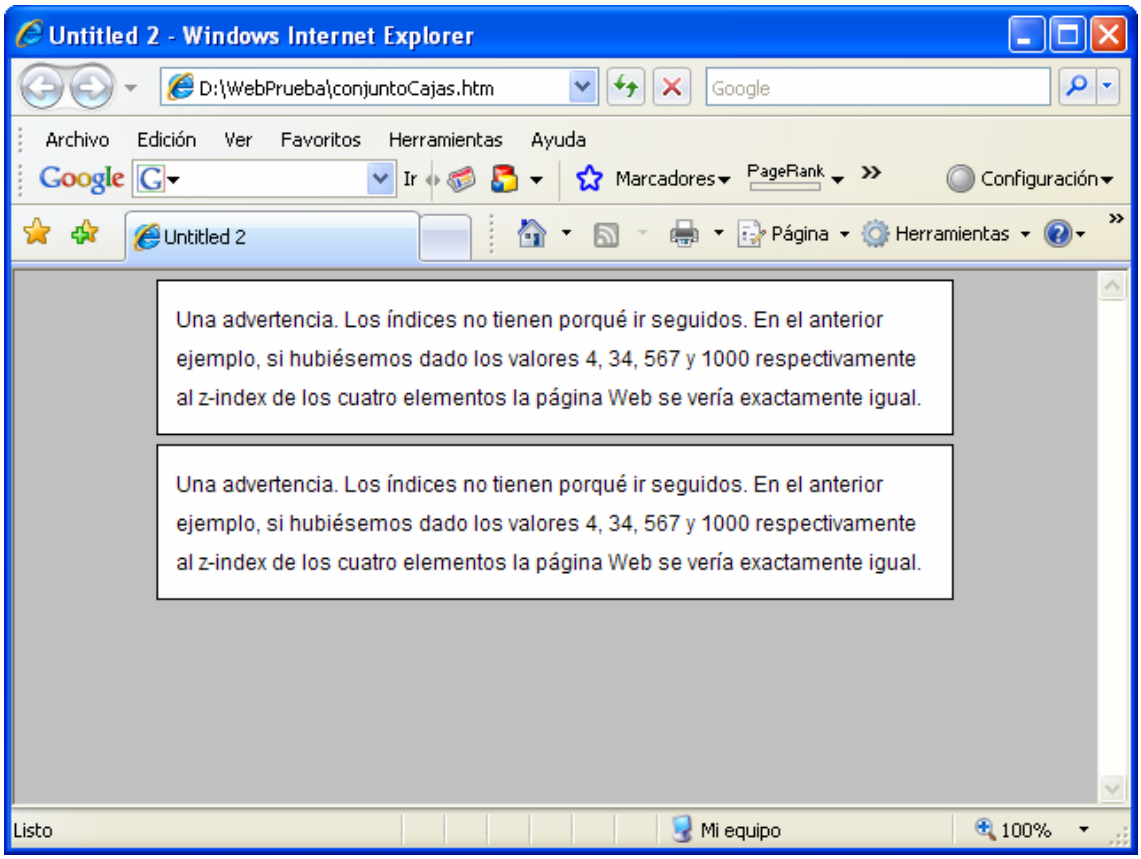
```
<body>
	<div id="caja1">Una advertencia. Los índices no tienen porqué ir
	seguidos. En el anterior ejemplo, si hubiésemos dado los valores
	4, 34, 567 y 1000 respectivamente al z-index de los cuatro
	elementos la página Web se vería exactamente igual.</div>

	<div id="caja2">Una advertencia. Los índices no tienen porqué ir
	seguidos. En el anterior ejemplo, si hubiésemos dado los valores
```

```
      4, 34, 567 y 1000 respectivamente al z-index de los cuatro
      elementos la página Web se vería exactamente igual.</div>

</body>
```

Y la hoja de estilos:

```
body {
      margin: 0px;
      padding: 0px;
      font-family: Arial, Helvetica, sans-serif;
      font-size: 12px;
      line-height: 22px;
      color: black;
      background-color: silver;
}


#caja1 {
      padding: 10px;
      margin: 5px auto 5px auto;
      background-color: white;
      border: 1px solid black;
      width: 70%;
}
#caja2 {
      padding: 10px;
      margin: 5px auto 5px auto;
      background-color: white;
      border: 1px solid black;
      width: 70%;
}
```

Esto se ve así:



Este ejemplo está en el video tutorial Cómo trabajar con un conjunto de cajas: Posicionamiento I de 7m 54 seg. de duración. Se explica cómo trabajar con un conjunto de cajas. Para ello, se implementa una página web con una sola caja centrada. Se procura que el contenido esté separado de la presentación. Se explica con este ejemplo a implementarla utilizando únicamente una hoja de estilos.

5.4.2 Dos columnas. Menú y cuerpo del documento

Ahora deseamos implementar con estilos dos columnas Eesto es muy común cuando, por ejemplo, se desea que una de ellas sea un menú.

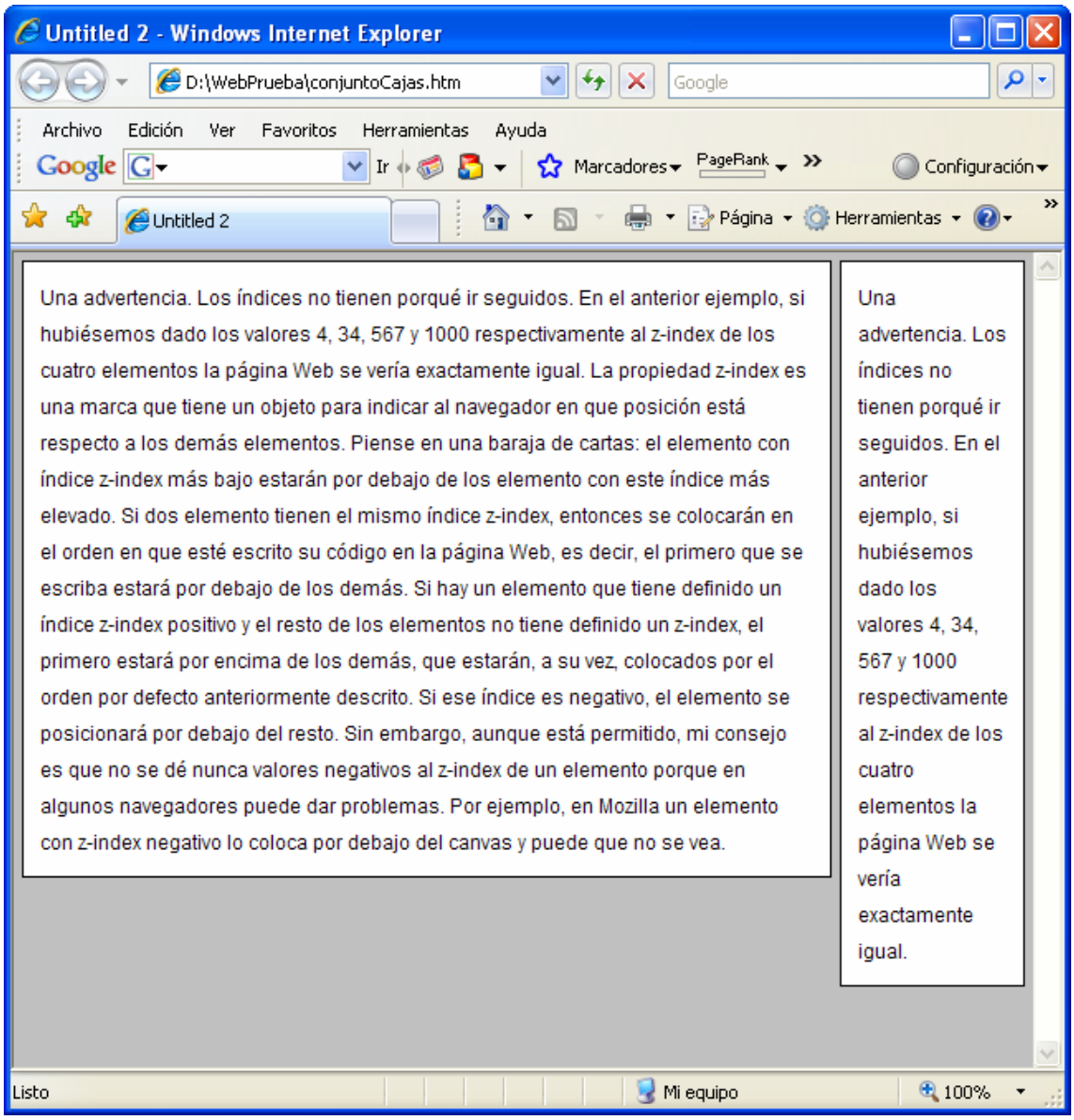
Hemos visto en un punto anterior que cada bloque –elemento `<div>`- ocupa siempre toda una fila. ¿Cómo es posible presentar dos bloques, en dos columnas?. La solución es sencilla: basta con que el primer bloque tenga un posicionamiento estático, con lo que ocupará toda la fila. Pero dejamos un margen derecho igual al ancho que deseamos para el otro bloque. Ahora posicionamos este segundo bloque con un posicionamiento absoluto, para que se ubique justo en el hueco dejado en el margen derecho. Ya tenemos nuestras dos columnas. Vamos a hacerlo:

Consideremos la anterior página web, pero para que quede un poco mejor voy a rellenar la primera de las dos cajas con más texto, aunque conservamos el esqueleto de la página:

Este ejemplo está recogido en el video Trabajar con juntos de cajas. Posicionamiento II de 10m 35seg de duración. Se explica con detenimiento cómo implementar una página con dos columnas, una de anchura fija, pensada para, por ejemplo ubicar un menú y la otra, de anchura variable, que en general contiene texto con información diversa. Este video tutorial es continuación del video anterior.

Consideremos la hoja de estilos siguiente:

```
body {
	margin: 0px;
	padding: 0px;
	font-family: Arial, Helvetica, sans-serif;
	font-size: 12px;
	line-height: 22px;
	color: black;
	background-color: silver;
}
#caja1 {
	padding: 10px;
	margin: 5px 122px 5px 5px;
	background-color: white;
	border: 1px solid black;
}
#caja2 {
	position:absolute;
	top:0px;
	right:0px;
	padding: 10px;
	margin: 5px;
	background-color: white;
	border: 1px solid black;
	width: 90px;
}
```



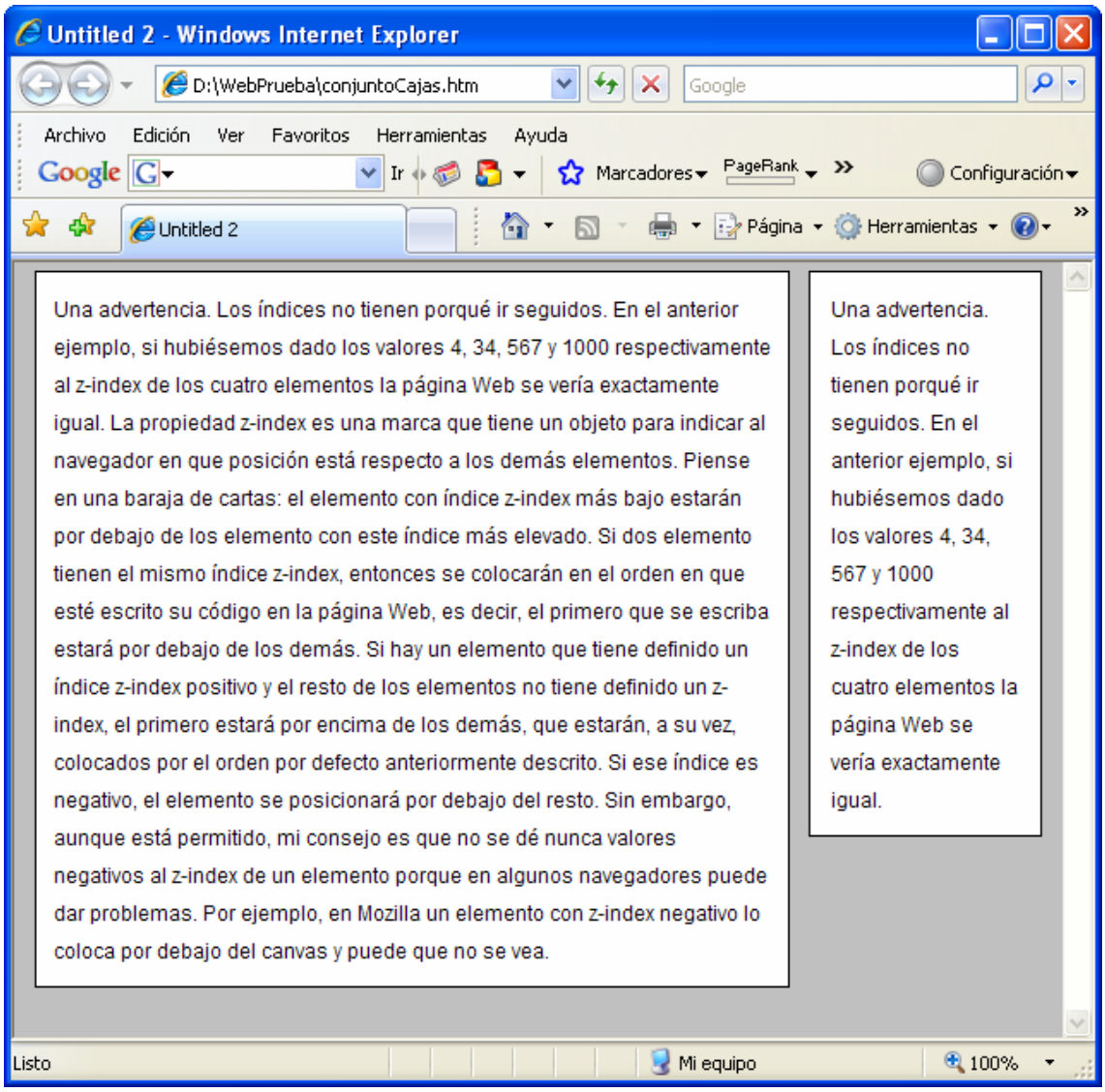
La `caja1` se sitúa a la izquierda porque se le ha dado un posicionamiento `static` –por defecto- y la anchura debería ser toda la línea pero al darle `margin-left: 122px` deja un hueco en la parte derecha de esa anchura. Ahora colocamos la `caja2` con un posicionamiento `absolute` –si fuese `static`, se situaría en la siguiente línea- en la esquina superior derecha (`top:0px; right:0px;`)- que como tiene un `margin` de `5px`,tendrá una anchura de `122px-5px-5px=112px`. Se resta dos veces porque tiene que guardar un margen por la derecha y otro por la izquierda.

Consideremos los sigiuentes cambios en nuestra hoja de estilos:

```
body {
      margin: 0px;
      padding: 0px;
      font-family: Arial, Helvetica, sans-serif;
      font-size: 12px;
      line-height: 22px;
      color: black;
      background-color: silver;
}
```

```
#caja1 {
	padding: 10px;
	margin: 5px 26% 5px 2% ;
	background-color: white;
	border: 1px solid black;
}
#caja2 {
	position:absolute;
	top:0px;
	right:0px;
	padding: 10px 5px 10px 5px;
	margin: 5px 2% 5px 1%;
	background-color: white;
	border: 1px solid black;
	width: 18%
}
```

La página se ve así:



En primer lugar, la `caja1` tiene un margen derecho de 26% del ancho de la página, para dejar espacio donde situar la otra columna. La `caja2` tiene un posicionamiento absoluto, se referencia igual que antes, en la esquina superior derecha.

5.4.3 Tres columnas: dos menús y el cuerpo del documento

Vamos a hora a poner tres columnas. Esto es común en la web, cuando se desea tener dos menús laterales y un cuerpo del documento en el centro.

En el video tutorial Cómo trabajar con conjuntos de cajas. Posicionamiento III de 3m 14 seg. de duración se explica cómo crear una página con tres columnas. Dos de ellas serán laterales y con un ancho fijo y posicionamiento absoluto, ubicadas en las partes derecha e izquierda de la página con la idea de que pudiesen ser cajas de menús. La tercera columna está situada en el centro y lleva en su interior un bloque de texto. La anchura de esta columna es variable y el texto fluye con la anchura de la columna. Este video tutorial es continuación del video anterior.

La página sería:

```
<body>
      <div id="cajaIzquierda">Una advertencia. Los índices no tienen
      porqué ir seguidos. En el anterior ejemplo, si hubiésemos dado
      los valores 4, 34, 567 y 1000 respectivamente al z-index de los
      cuatro elementos la página Web se vería exactamente igual.</div>

      <div id="cajaCentro">Una advertencia. Los índices no tienen
      porqué ir seguidos. En el anterior ejemplo, si hubiésemos dado
      los valores 4, 34, 567 y 1000 respectivamente al z-index de los
      cuatro elementos la página Web se vería exactamente igual.
      La propiedad z-index es una marca que tiene un objeto para
      indicar al navegador en que posición está respecto a los demás
      elementos. Piense en una baraja de cartas: el elemento con
      índice z-index más bajo estarán por debajo de los elemento
      con este índice más elevado. Si dos elemento tienen el mismo
      índice z-index, entonces se colocarán en el orden en que esté
      escrito su código en la página Web, es decir, el primero que se
      escriba estará por debajo de los demás. Si hay un elemento
      que tiene definido un índice z-index positivo y el resto de
      los elementos no tiene definido un z-index, el primero estará
      por encima de los demás, que estarán, a su vez, colocados por
      el orden por defecto anteriormente descrito. Si ese índice es
      negativo, el elemento se posicionará por debajo del resto.
      Sin embargo, aunque está permitido, mi consejo es que no se
      dé nunca valores negativos al z-index de un elemento porque en
      algunos navegadores puede dar problemas. Por ejemplo, en Mozilla
      un elemento con z-index negativo lo coloca por debajo del
      canvas y puede que no se vea.</div>

      <div id="cajaDerecha">Una advertencia. Los índices no tienen
      porqué ir seguidos. En el anterior ejemplo, si hubiésemos dado
      los valores 4, 34, 567 y 1000 respectivamente al z-index de los
      cuatro elementos la página Web se vería exactamente igual.</div>

</body>
```

La hoja de estilos es:

```
body {
      margin: 0px;
```

```
      padding: 0px;
      font-family: Arial, Helvetica, sans-serif;
      font-size: 12px;
      line-height: 22px;
      color: black;
      background-color: silver;
}
#cajaIzquierda {
      position:absolute;
      top:0px;
      left:0px;
      padding: 10px;
      margin: 5px;
      background-color: white;
      border: 1px solid black;
      width:90px;
}
#cajaCentro {
      padding: 10px;
      margin: 5px 122px 5px 122px;
      background-color: white;
      border: 1px solid black;

}

#cajaDerecha {
      position:absolute;
      top:0px;
      right:0px;
      padding: 10px;
      margin: 5px;
      background-color: white;
      border: 1px solid black;
      width:90px;
}
```

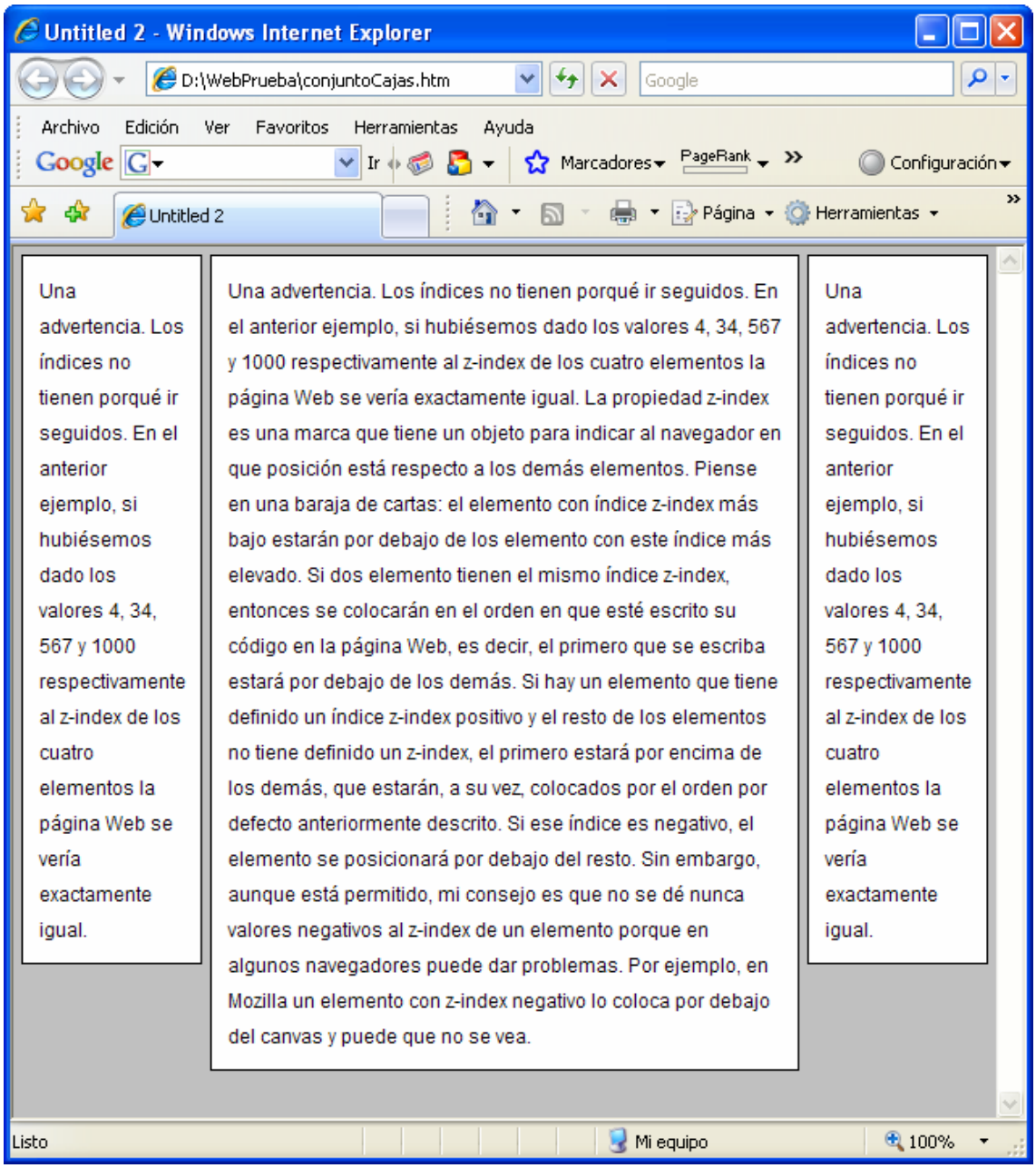
Para situar las cajas hacemos lo siguiente:

`cajaCentro`. Tiene un posicionamiento `static` –por defecto-. Dejamos un margen en los lados izquierdo y derecho de `122px` para que puedan ponerse los dos menús. Esta caja ocupará toda la línea. Sólo pueden ubicarse aquí los objetos que tengan un posicionamiento absoluto. Eso es lo que hacemos con las otras dos cajas.

`cajaIzquierda`. Con un posicionamiento `absolute` la situamos en la esquina superior izquierda, poniendo las propiedades `top` y `left` a `0px`. Le damos una anchura de `90px`.

`cajaDerecha`. La situamos con un posicionamiento absolute en la esquina superior derecha, poniendo las propiedades `top` y `right` a `0px`, y le damos una anchura de `90px`.

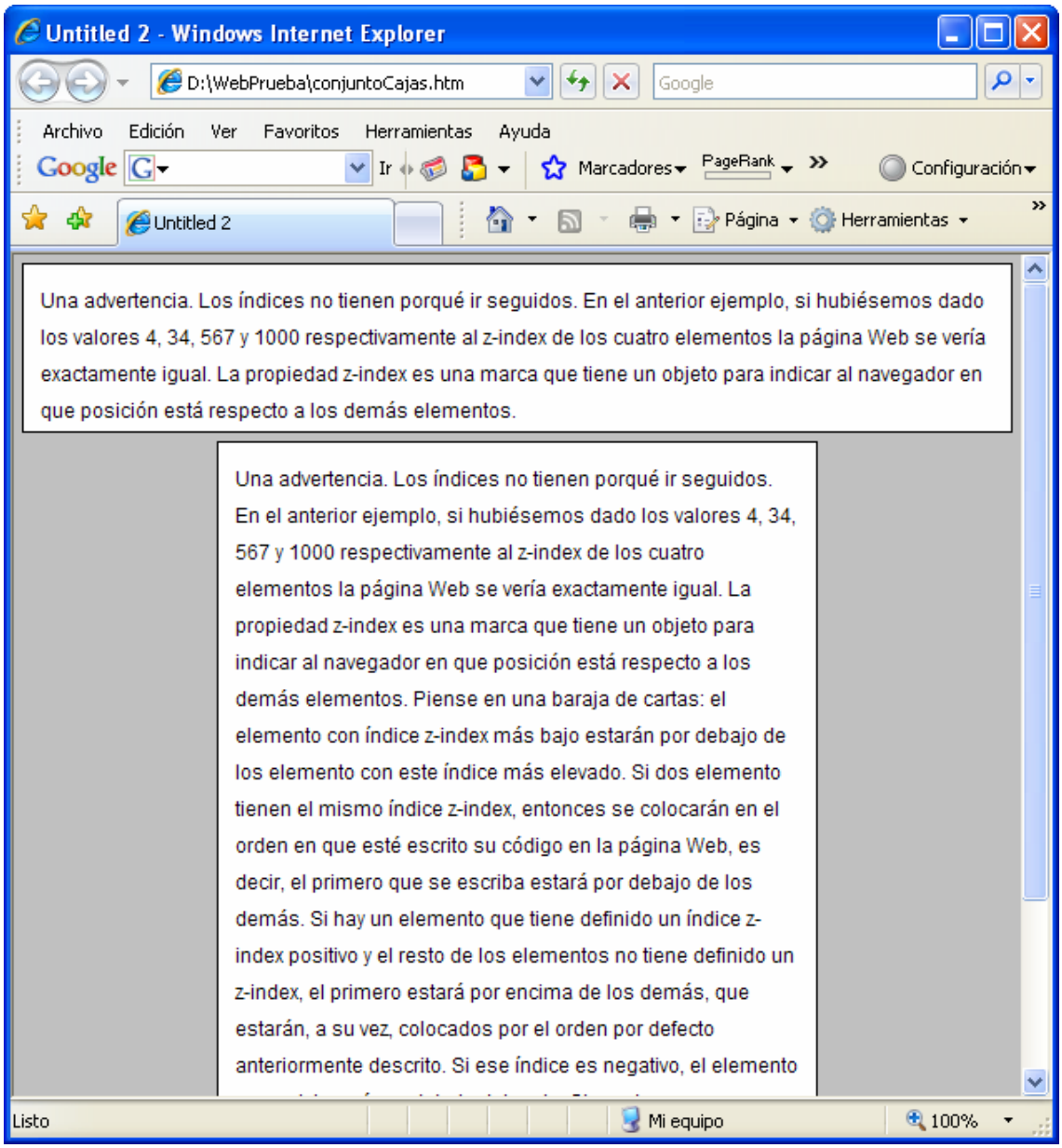
La página se ve así:



5.4.4 Tres columnas y una fila encima de ellas

Ahora vamos a colocar una fila por encima de las dos anteriores.

En primer lugar colocamos las dos cajas centrales. Para ello, pensaremos en que la caja superior y la central tendrán un posicionamiento `static`, normal, es decir, flotando, como hemos hecho en el ejemplo de las dos filas. La primera se sitúa en la primera fila ocupando todo el ancho y la `cajaCentro` en la segunda fila reservando espacio en los laterales de `122px` para colocar las otras dos cajas con posicionamiento `absolute`. A la caja `cajaSuperior` le daremos una altura fija (`height:80px`), porque así podremos posicionar los menús debajo. Estas dos cajas formarán una T como en la figura:



Los estilos serán:

```
#cajaCentro {
	padding: 10px;
	margin: 5px 122px 5px 122px;
	background-color: white;
	border: 1px solid black;

}
#cajaSuperior{
	height:80px;
	margin:5px;
	padding: 10px;
	background-color: white;
	border: 1px solid black;

}
```

A las cajas laterales les variamos la propiedad `top` y la ajustamos a `110 px` para que no se solapen con la caja superior –hay que tener en cuenta el `margin` (dos veces), el `padding`(dos veces) y `height` de la `cajaSuperiro` para definir `top` de las cajas laterales.

Nuestra página sería:

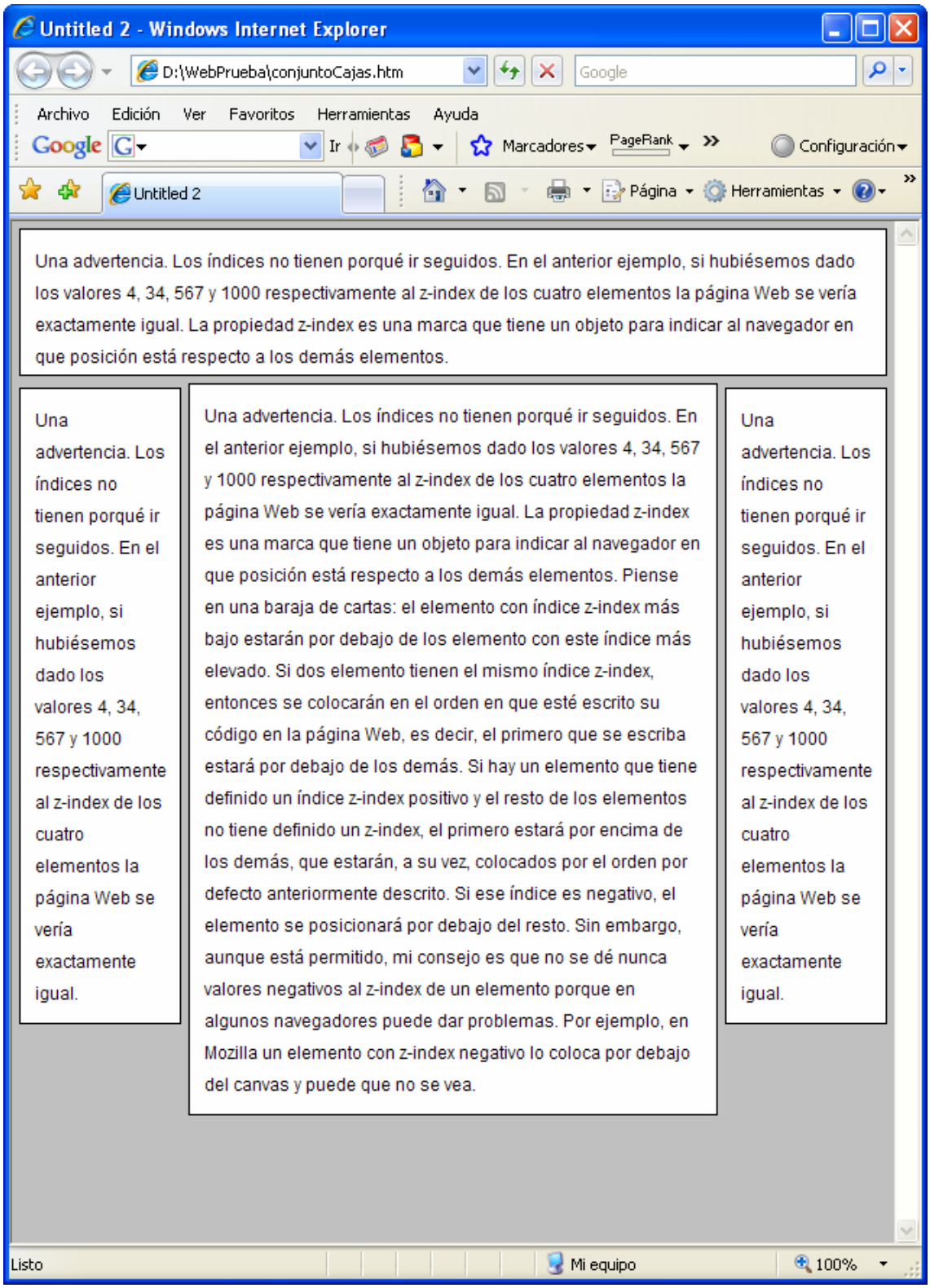
```
<body>
	<div id="cajaSuperior">Una advertencia. Los índices no tienen porqué ir seguidos. En el anterior ejemplo, si hubiésemos dado los valores 4, 34, 567 y 1000 respectivamente al z-index de los cuatro elementos la página Web se vería exactamente igual.La propiedad z-index es una marca que tiene un objeto para indicar al navegador en que posición está respecto a los demás elementos. </div>
	<div id="cajaIzquierda">Una advertencia. Los índices no tienen porqué ir seguidos. En el anterior ejemplo, si hubiésemos dado los valores 4, 34, 567 y 1000 respectivamente al z-index de los cuatro elementos la página Web se vería exactamente igual.</div>

	<div id="cajaCentro">Una advertencia. Los índices no tienen porqué ir seguidos. En el anterior ejemplo, si hubiésemos dado los valores 4, 34, 567 y 1000 respectivamente al z-index de los cuatro elementos la página Web se vería exactamente igual. La propiedad z-index es una marca que tiene un objeto para indicar al navegador en que posición está respecto a los demás elementos. Piense en una baraja de cartas: el elemento con índice z-index más bajo estarán por debajo de los elemento con este índice más elevado. Si dos elemento tienen el mismo índice z-index, entonces se colocarán en el orden en que esté escrito su código en la página Web, es decir, el primero que se escriba estará por debajo de los demás. Si hay un elemento que tiene definido un índice z-index positivo y el resto de los elementos no tiene definido un z-index, el primero estará por encima de los demás, que estarán, a su vez, colocados por el orden por defecto anteriormente descrito. Si ese índice es negativo, el elemento se posicionará por debajo del resto. Sin embargo, aunque está permitido, mi consejo es que no se dé nunca valores negativos al z-index de un elemento porque en algunos navegadores puede dar problemas. Por ejemplo, en Mozilla un elemento con z-index negativo lo coloca por debajo del canvas y puede que no se vea.</div>
	<div id="cajaDerecha">Una advertencia. Los índices no tienen porqué ir seguidos. En el anterior ejemplo, si hubiésemos dado los valores 4, 34, 567 y 1000 respectivamente al z-index de los cuatro elementos la página Web se vería exactamente igual.</div>

</body>
```

Y la hoja de estilos sería:

```
body {
	margin: 0px;
	padding: 0px;
	font-family: Arial, Helvetica, sans-serif;
	font-size: 12px;
	line-height: 22px;
	color: black;
	background-color: silver;
```

```
}
#cajaIzquierda {
      position:absolute;
      top:110px;
      left:0px;
      padding: 10px;
      margin: 5px;
      background-color: white;
      border: 1px solid black;
      width:90px;
}
#cajaCentro {
      padding: 10px;
      margin: 5px 122px 5px 122px;
      background-color: white;
      border: 1px solid black;

}

#cajaDerecha {
      position:absolute;
      top:110px;
      right:0px;
      padding: 10px;
      margin: 5px;
      background-color: white;
      border: 1px solid black;
      width:90px;
}
#cajaSuperior{
      height:80px;
      margin:5px;
      padding: 10px;
      background-color: white;
      border: 1px solid black;

}
```

La representación es:



5.4.5 Tres columnas y una fila por encima y otra por debajo

Ahora podríamos pensar en hacer lo mismo con una caja inferior. Esto no es posible porque las distancias no son fijas y se solaparían. No podríamos utilizar un

posicionamiento absoluto para ello, ni tampoco situarla desde abajo, es decir, configurando la propiedad `bottom`. Lo que sí podríamos hacer es colocar la caja inferior debajo de la caja del centro.

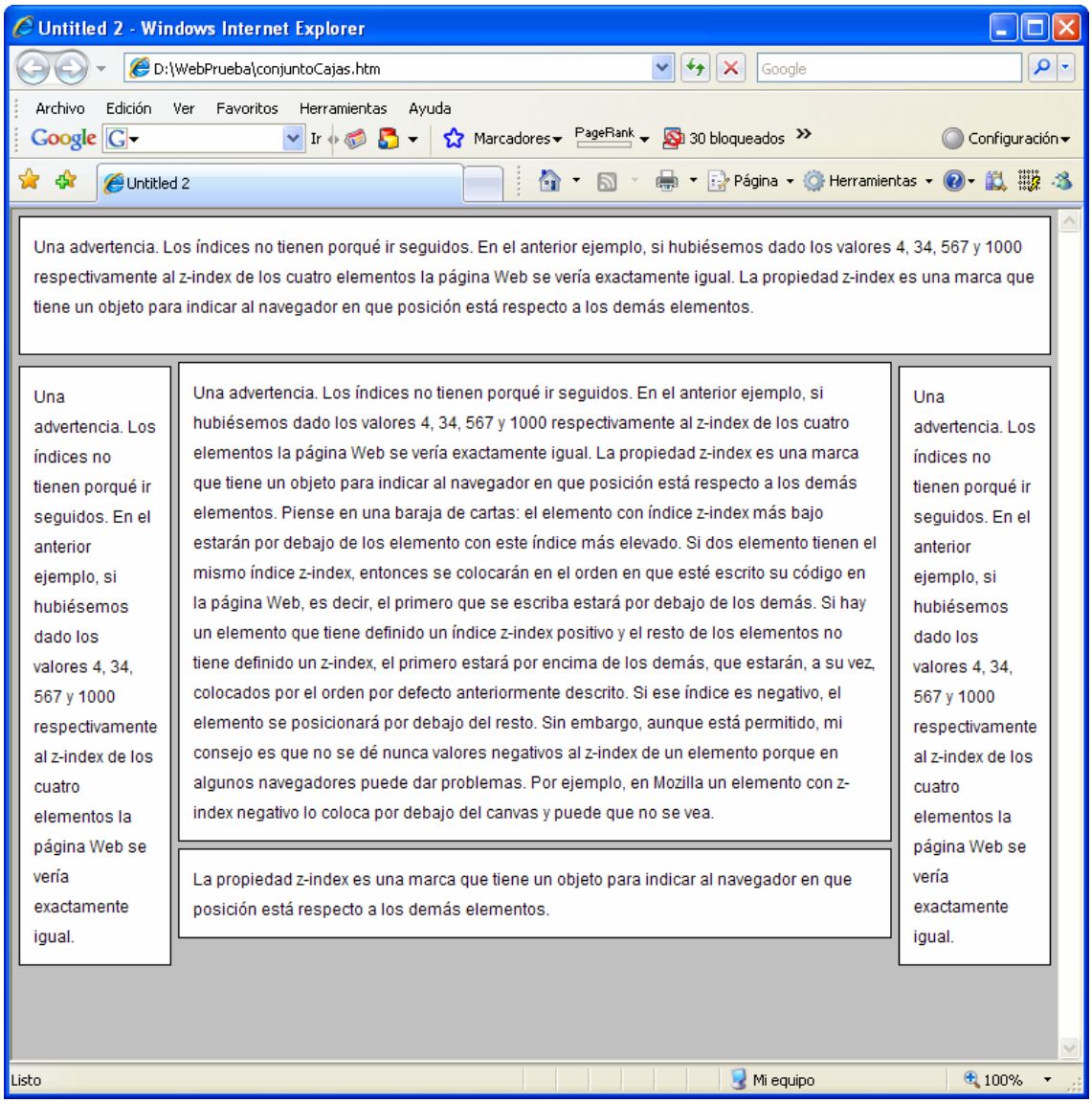
Añadiríamos en la página otra caja:

```
<div id="cajaInferior">La propiedad z-index es una marca que tiene un objeto para indicar al navegador en que posición está respecto a los demás elementos. </div>
```

Y en la hoja de estilos el identificador:

```
#cajaInferior{
	padding: 10px;
	margin: 5px 122px 5px 122px;
	background-color: white;
	border: 1px solid black;
}
```

Esto se ve así:



5.5 ANIDAR CAJAS

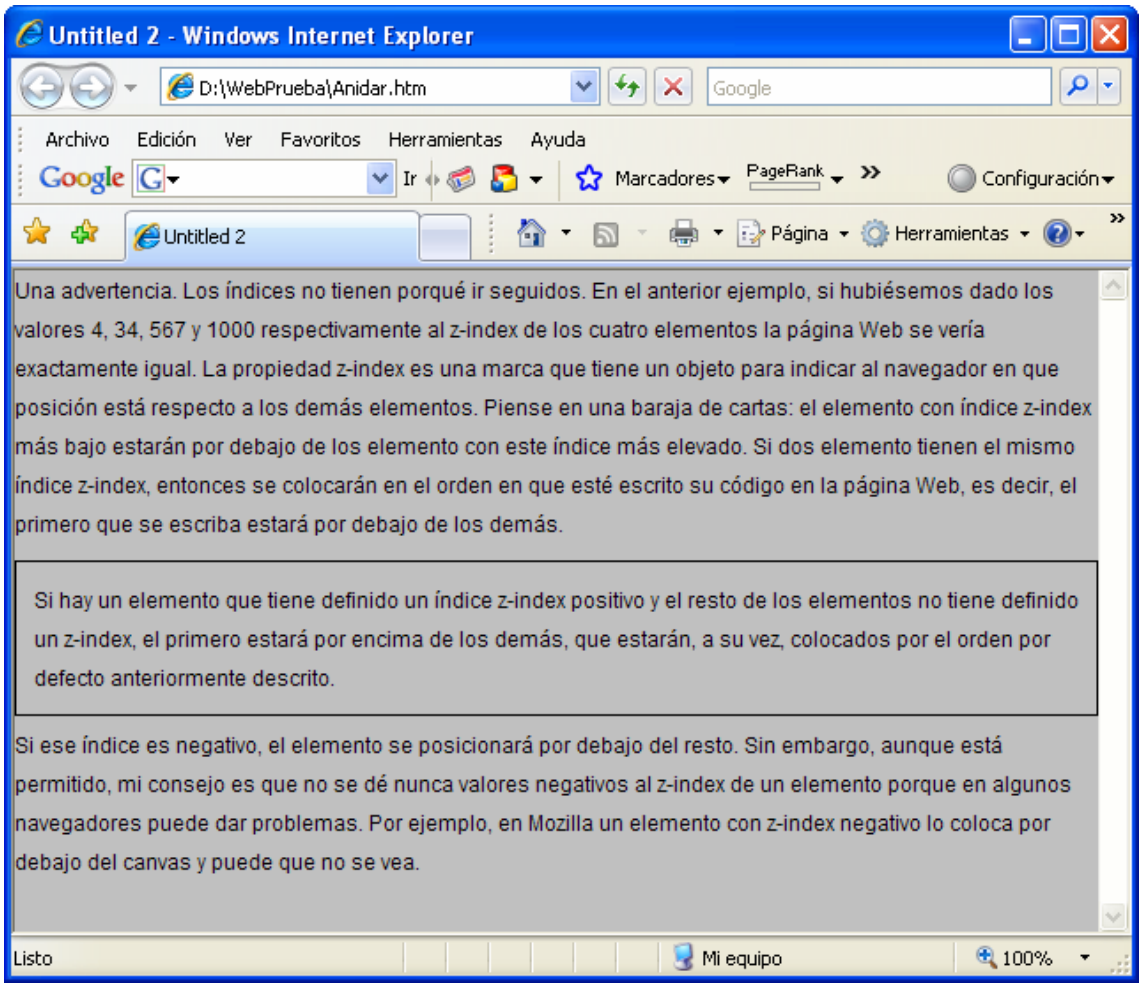
Consideremos la página web siguiente:

```
<body>
      <div id="cajaPrincipal">Una advertencia. Los índices no tienen
      porqué ir seguidos. En el anterior ejemplo, si hubiésemos dado
      los valores 4, 34, 567 y 1000 respectivamente al z-index de los
      cuatro elementos la página Web se vería exactamente igual.La
      propiedad z-index es una marca que tiene un objeto para indicar
      al navegador en que posición está respecto a los demás
      elementos. Piense en una     baraja de cartas: el elemento con
      índice z-index más bajo estarán por debajo de los elemento con
      este índice más elevado. Si dos elemento tienen el mismo índice
      z-index, entonces se colocarán en el orden en que esté escrito
      su código en la página Web, es decir, el primero que se escriba
      estará por debajo de los demás.
            <div id="cajaAnidada">Si hay un elemento que tiene definido
            un índice z-index positivo y el resto de los elementos no
            tiene definido un z-index, el primero estará por encima de
            los demás, que estarán, a su vez, colocados por el orden
            por defecto anteriormente descrito.
            </div>
      Si ese índice es negativo, el elemento se posicionará por debajo
      del resto. Sin embargo, aunque está permitido, mi consejo es que
      no se dé nunca valores negativos al z-index de un elemento
      porque en algunos navegadores puede dar problemas. Por ejemplo,
      en Mozilla un elemento con z-index negativo lo coloca por debajo
      del canvas y puede que no se vea.
      </div>
</body>
```

Si damos un estilo al elemento `body` y definimos un estilo para el elemento que va a estar anidado –que tenga un borde y un poco más de margen-

```
body {
      margin: 0px;
      padding: 0px;
      font-family: Arial, Helvetica, sans-serif;
      font-size: 12px;
      line-height: 22px;
      color:black;
      background-color: silver;
}
#cajaAnidada{
      padding: 10px;
      margin: 10px 0px 5px 0px;
      border: 1px solid #000000;
}
```

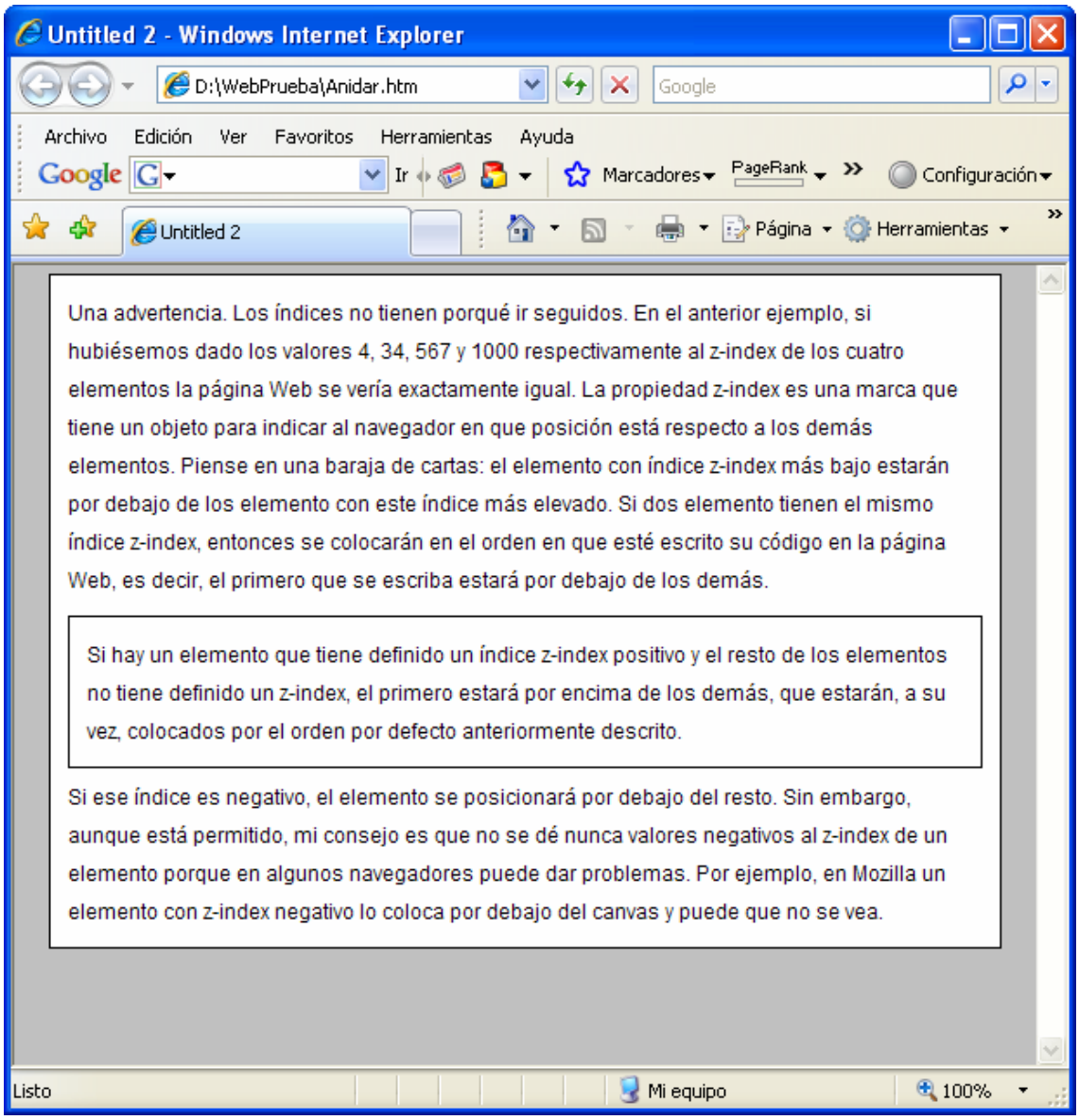
Se ve así:



Vamos a darle un estilo a la caja contenedora, `cajaPrincipal`

```
body {
	margin: 0px;
	padding: 0px;
	font-family: Arial, Helvetica, sans-serif;
	font-size: 12px;
	line-height: 22px;
	color:black;
	background-color: silver;
}
#cajaAnidada{
	padding: 10px;
	margin: 10px 0px 5px 0px;
	border: 1px solid black;
}
#cajaPrincipal {
	padding: 10px;
	margin: 5px 20px 5px 20px;
	background-color: white;
	border: 1px solid black;
}
```

Se ve así:



Las cajas anidadas se utilizan mucho porque se pueden aprovechar todos los mecanismos de la herencia para trabajar los estilos de las cajas “internas” o “hijas”. Por ejemplo, se pueden utilizar las cajas anidadas para definir menús en la parte inferior de la pantalla.