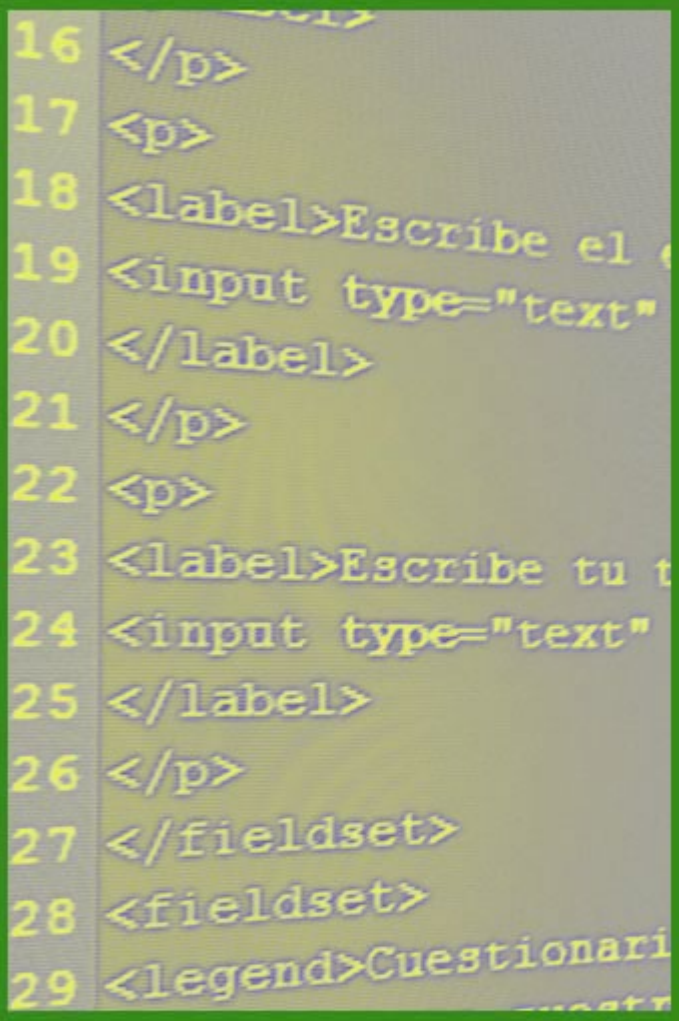
mosaic
comunicación interactiva y tecnologías multimedia

Graduado en Multimedia

Pere Barnola Augé




# Introducción a la Creación de Páginas Web

UOC

http://mosaic.uoc.edu

**A la hora de crear páginas web, el respeto a los estándares y el uso correcto de XHTML y CSS es imprescindible. En este recurso se ofrece una introducción al tema desgranando uno por uno los elementos básicos, siguiendo un sencillo caso práctico al que se van añadiendo los diferentes elementos y pasos a seguir que pueden moldear la estructura y la presentación del contenido hasta llegar a la publicación.**

**Autor: Pere Barnola Augé** es ingeniero superior multimedia por la Universitat Ramon Llull. Posteriormente colaboró con la University of Calgary (Canadá), donde publicó varios artículos sobre métodos de clasificación automática de textos y tratamiento del lenguaje. Ha trabajado en el departamento de proyectos europeos de la Fundació Catalana per a la Recerca, gestionando y desarrollando la parte tecnológica de los proyectos, una tarea que también ha desarrollado en Whads-Accent. Actualmente lidera el área de interactivos en **Identitat**, empresa de la que es fundador.

Primera edición: septiembre 2008

# Índice

# Introducción

## Breve historia del World Wide Web y la hipermedia

Comenzaremos con el concepto básico: **hipermedia**.

**La hipermedia es el conjunto de procedimientos o métodos que agrupan texto, vídeo, audio u otros tipos de información con referencias cruzadas entre los contenidos, con la principal característica de que el usuario puede interaccionar con ellas.**

### Ejemplo

Imaginad un documento en que se habla del Everest, y que en un punto de este documento aparece el nombre de "Edmund Hillary", que fue el primer escalador que alcanzó su cima. En un documento convencional, lo más probable es que tuviéramos un asterisco, o un número que hiciera referencia a una anotación al pie de página donde se explicarían brevemente las proezas de dicha persona. Pues bien, la hipermedia nos permitiría que el nombre fuera un enlace, y que, cuando el usuario hiciera clic (interaccionara), accediéramos a otro documento con toda la información de Edmund Hillary, con la opción de encontrar más referencias en ese documento, entre ellas, una al documento del Everest, o incluso imágenes, vídeos..., con lo que se podría construir una gran red de contenidos con referencias entrecruzadas.

El ejemplo más claro de hipermedia es el **World Wide Web**, que no es nada más que un sistema de documentos de hipermedia enlazados y accesibles por Internet.

Actualmente, con un navegador web (Explorer, Firefox, etc.), un usuario visualiza páginas web con diferentes tipos de contenidos (texto, imágenes, vídeos) y puede navegar por ellos utilizando los hipervínculos.

### Un poco de historia

La idea inicial de web se remonta a 1945, cuando el director de la Oficina de Desarrollo e Investigación Científica (Estados Unidos), el doctor Vannevar Bush, escribió el artículo "As We May Think", en el cual expresaba su preocupación por la gran cantidad de información que existía y se estaba generando, y los sistemas poco eficientes para ordenarla y encontrarla. Así, basándose en la tecnología de aquellos tiempos, describió el Memex, que imaginaba como un suplemento a su memoria, y que permitiría a cualquier persona guardar su información en microfilms para poder consultarla posteriormente de forma rápida, y lo que es más importante, creó vínculos entre unos documentos y los otros, de manera que durante la lectura de un documento se recordara al lector cuáles eran los que contenían información relacionada. Era una visión de lo que pasaría cuarenta y cinco años más adelante.

El término hipertexto fue utilizado por primera vez por Ted Nloson en 1965, quien ideó un modelo para la interconexión de documentos electrónicos. Antes, sin embargo, habían surgido otros proyectos como el NLS (oNLine System), de Douglas Engelbart, un entorno de trabajo por ordenador, con teclado, pantalla, ratón e impresora, con posibilidad de teleconferencia y correo electrónico mediante una red de ordenadores para la comunicación rápida entre los profesionales.

El World Wide Web fue inventado en 1989 por un informático del CERN (Organización Europea de Investigación Nuclear) llamado Tim Berners-Lee. Era un sistema de hipertexto para compartir información basado en Internet, concebido originalmente para servir como herramienta de comunicación entre los científicos nucleares del CERN. A finales de 1990, el primer navegador de la historia, el WorldWideWeb, ya tenía forma.

A principios de 1993 había en torno a cincuenta servidores (ordenadores que alojaban webs). Había dos tipos de browsers o navegadores para cualquier plataforma, pero muy limitados y muy poco atractivos. En febrero, se lanzó la primera versión alfa del

navegador Mosaic for X, desarrollado en el NCSA (National Center for Supercomputing Applications). Funcionaba en X Windows, que era una plataforma popular entre la comunidad científica. En abril, el tráfico del WWW era el 0,1% del total de Internet. El CERN declaraba el WWW como tecnología de acceso gratuito. En septiembre, ya había versiones de mosaico para PC y Macintosh. El tráfico llegaba al 1% de todo el tráfico de Internet y había más de 500 servidores. Es el comienzo del crecimiento explosivo de la web. A finales de 1994, ya había más de 10.000 servidores y 10 millones de usuarios. Y en 1997, más de 650.000 servidores.

Hoy en día, la web es algo cotidiano para una gran parte de los más de 600 millones de usuarios de Internet que hay en todo el mundo. Sus utilidades son variadas, y su impacto en la economía mundial es más que apreciable. No sólo existen documentos de texto, sino que también hay imágenes, vídeos, música; además, se puede comprar cualquier producto de todo el mundo, hacer reservas, etc.

Cada uno de nosotros puede ir tejiendo una parte de esta gran red...

## Los estándares

Cuando el volumen en Internet empieza a crecer, se hace necesario una estandarización que vuelva a situar el intercambio de información accesible como eje principal, y establezca las bases del siguiente paso en la evolución de la web; una web basada en un código semántico que aporte más significado a la información, para facilitar su localización y recuperación.

Los estándares que harán que eso sea posible son desarrollados por un consorcio integrado por diferentes organizaciones de diferentes países, conocido como el **World Wide Web Consortium (W3C).**

Este consorcio fue creado a mediados de los años noventa por Tim Berners-Lee.

Su principal propósito es que la web evolucione y desarrolle todo su potencial de forma ordenada, y sin perder de vista su concepto inicial de intercambio de información para que sea accesible, interoperable e independiente de tecnologías propietarias. Los estándares se desarrollan en un proceso abierto en que cualquier empresa puede participar, pero nunca ser propietaria.

Finalmente, depende de nosotros respetar el principio de universalidad o no. Cada vez que alguien crea una página de manera que no es accesible para todo el mundo, o que sólo es visible para los que utilizan un determinado navegador, sea cual sea, está contribuyendo a hacer que la web sea un sitio más fragmentado y peor.

La web tendría que ser universal y para todo el mundo, pero si no seguimos una serie de normas, no todos podrán tener acceso a ella.

**Ejemplo**

Un ejemplo muy claro es el de los invidentes, que navegan con un sintetizador de voz, que toma el texto de una web y lo lee en voz alta. Por lo tanto, en el caso de las imágenes, por ejemplo, si éstas no contienen una buena descripción de lo que se ve, habrá gente que no podrá acceder a ese contenido.

El esfuerzo del W3C ha dado pie a más de **70 tecnologías**. Partieron del HTML para llegar al modelo "actual", donde nos encontramos nuevas tecnologías para modelizar hacia dónde está yendo la web. Nuevas tecnologías para:

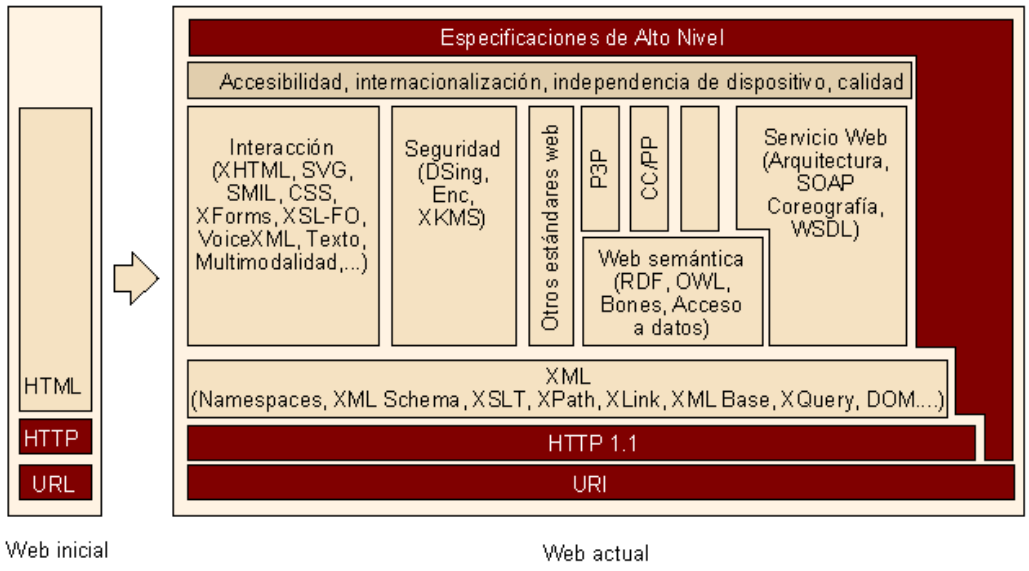
- web semántica: RDF, OWL
- servicios web: SOAP y WSDL
- gráficos y multimedia (SVG, SMIL)
- diálogos de voz (VoiceXML)

- formularios interactivos (XForms)
- documentos de texto (XHTML, MathML)
- presentación de contenidos (CSS)
- etc.

La mayoría son aplicaciones del XML, pero también hay tecnologías no basadas en XML (WebCGM, PNG).

Se ha de decir que, mientras que algunas de estas tecnologías ya están bastante maduras, todavía hay otras en pleno desarrollo.

Esquematización de la evolución de los estándares web: del HTML a la web actual




**Más información sobre estándares web**

Podréis encontrar más información sobre estándares web en **Web Standards Project** y **World Wide Web Consortium W3C**.

## HTML, XML y XHTML

**HTML**

HTML es el acrónimo inglés de *hypertext markup language* ('lenguaje de marcaje de hipertexto').

HTML es el lenguaje de marcaje predominante para la creación de páginas web. Se utiliza para describir la estructura y el contenido en forma de texto, así como para complementar el texto con objetos como imágenes. El HTML se escribe en forma de etiquetas entre corchetes angulares (<,>). El HTML también puede describir, hasta cierto punto, la apariencia de un documento, y puede incluir scripts (por ejemplo, Javascript), lo cual puede afectar al comportamiento de navegadores web y otros procesadores de HTML.

Por convención, los archivos en formato HTML utilizan la extensión .htm o .html.

El HTML se desarrolló como un subconjunto del SGML (un lenguaje de marcas más complejo) y es el lenguaje de marcas que ha hecho posible la web tal como la conocemos actualmente.

El HTML 4.01 es la última versión de HTML recomendada por el W3C.

Como ya hemos visto, el HTML se definió dentro del marco de SGML y fue de lejos la aplicación más conocida de este estándar. De todos modos, los navegadores nunca han puesto muchas exigencias al código HTML que interpretan, y las páginas web pueden llegar a ser muy caóticas si no cumplen la sintaxis. Sin embargo, hay que decir que no todos los navegadores siguen los estándares, lo cual significa que una misma web se puede visualizar de manera diferente según qué navegador la interprete.

**XML**

XML es el acrónimo inglés de *extensible markup language* (lenguaje extensible de marcas).

XML es un metalenguaje extensible de etiquetas desarrollado por el World Wide Web Consortium (W3C). Es una simplificación y adaptación del SGML y permite definir la gramática de lenguajes específicos (de la misma manera que HTML es un lenguaje definido por SGML). Por lo tanto, el XML no es realmente un lenguaje en particular, sino una manera de definir lenguajes para diferentes necesidades.

El XML no ha nacido sólo para su aplicación en Internet, sino que se propone como un estándar para el intercambio de información entre diferentes plataformas. Se puede utilizar en bases de datos, editores de texto, hojas de cálculo y casi cualquier cosa imaginable.

Es extensible, lo que quiere decir que, una vez diseñado un lenguaje y puesto en producción, se puede extender añadiendo nuevas etiquetas, de manera que los antiguos consumidores de la vieja versión puedan seguir entendiendo el nuevo formato.

Si un tercero decide utilizar un documento creado en XML, es muy fácil entender su estructura y procesarlo de forma automática, pues mejora la compatibilidad entre aplicaciones.

La tecnología XML busca dar solución al problema de expresar información estructurada de la manera más abstracta y reutilizable posible.

**XHTML**

XHTML es el acrónimo inglés de *extensible hypertext markup language* (lenguaje extensible de marcaje de hipertexto).

XHTML es el lenguaje de marcaje pensado para sustituir el HTML como estándar para las páginas web. XHTML es la versión XML de HTML, y por lo tanto tiene, básicamente, las mismas funcionalidades, aunque cumple las especificaciones, más estrictas, del XML.

Su objetivo es avanzar en el proyecto del World Wide Web Consortium de conseguir una web semántica, donde la información y la forma de presentarla estén claramente separadas. En este sentido, el XHTML serviría únicamente para transmitir la información que contiene un documento y se dejaría para las hojas de estilos su aspecto y diseño en diferentes medios (ordenadores, PDA, móviles, impresoras).

El XHTML es el sucesor del HTML. Por ello, muchos lo consideran la "versión actual" del HTML, aunque es una recomendación aparte y al mismo tiempo paralela. El W3C continúa recomendando el uso de XHTML 1.1, XHTML 1.0, o HTML 4.01 para publicar en la web.

Los cambios de HTML en la primera generación de XHTML (es decir, XHTML 1.x) son menores, ya que, principalmente, están destinados a conseguir la conformidad con XML.

El cambio más importante es el requisito de que el documento esté bien formado y que todas las etiquetas estén explícitamente cerradas, tal como se requiere en XML.

Como las etiquetas en XML distinguen entre mayúsculas y minúsculas, la recomendación XHTML ha definido todos los nombres de etiqueta en minúsculas. En XHTML, los valores de los atributos tendrán que cerrarse entre comillas (siempre comillas "dobles").

**XHTML 1.0**

Para ver todas las diferencias entre los lenguajes de marcaje, podéis consultar de forma detallada la recomendación XHTML 1.0 del W3C.

## La web semántica, la accesibilidad y la separación de diseño y contenido

En la actualidad, el World Wide Web se basa principalmente en documentos escritos en HTML, un lenguaje de marcas que sirve principalmente para crear hipertexto en Internet. El lenguaje HTML es válido para adecuar el aspecto visual de un documento e incluir objetos multimedia en el texto (imágenes, esquemas de diálogo). Sin embargo, ofrece pocas posibilidades para categorizar y definir los elementos que configuran el texto más allá de las típicas funciones estructurales.

La web semántica se ocupa de resolver estas deficiencias. Para conseguirlo, dispone de tecnologías de descripción de los contenidos que se combinan para aportar descripciones explícitas de los recursos de la web (XML, XHTML, etc.).

Son informaciones adicionales que describen el contenido, el significado, y la relación de los datos. El objetivo es mejorar Internet ampliando la interoperabilidad entre los sistemas informáticos y automatizar al máximo los trabajos de gestión de información.

De esta forma, el contenido queda descrito como los datos de una base de datos accesible por web, o las etiquetas incluidas en el documento. Estas etiquetas permiten que los gestores de contenidos interpreten y realicen procesos inteligentes de captura y tratamiento de la información.

**Lo que intentaremos, por lo tanto, es estructurar y marcar el documento de la forma más clarificadora posible. Esto consiste, como veremos más adelante, en separar totalmente el contenido (XHTML) de la forma en que éste se muestra (CSS), para dar el máximo de información y etiquetar de la forma más descriptiva posible nuestro documento.**

De alguna manera, sería como decir "eso es una cabecera", "eso es una lista de ingredientes", "ésta es la etiqueta y éste es el valor"...

El hecho de que nuestro contenido esté mejor descrito hará a la vez que éste sea más accesible para la mayoría de usuarios de Internet. Al final, no se trata más que de llamar y marcar las cosas por lo que son.

# Contenido y estructura de un documento XHTML

El XHTML consiste en diferentes componentes vitales. Veremos a continuación sus elementos, atributos, tipos de fecha y la declaración de tipo de documento.

Para hacerlo más comprensible, iremos construyendo un documento XHTML de forma paralela, a modo de ejemplo, en cada uno de los elementos que explicaremos.

El lenguaje XHTML se puede crear y modificar con cualquier editor de texto básico, como puede ser el Bloc de Notas de Windows, o cualquier otro editor que admita texto sin formato GNU Emacs, Microsoft Wordpad, TextPad, Vim, Note pad++...

**Programas WYSIWYG**

Hay también otros programas para la realización de sitios web o edición de código XHTML, como por ejemplo el Dreamweaver de Adobe, el Expression Web de Microsoft o el CompoZer. A estos programas se los conoce como WYSIWYG o *what you see is what you get* ('lo que ves es lo que obtienes'). Esto significa que son editores en los cuales se ve el resultado de lo que se está editando en tiempo real a medida que se va desarrollando el documento. Ahora bien, esto no significa una manera diferente de realizar webs, sino una manera más sencilla, ya que estos programas, además de tener la opción de trabajar con la vista preliminar, tienen su propia sección XHTML, la cual va generando el código a medida que se va escribiendo.

Para empezar y tener más claro el concepto, nosotros utilizaremos un editor de texto simple, en nuestro caso, el Bloc de Notas de Windows.

## 1. Elementos principales

Antes de ver los elementos que componen un XHTML diremos, como introducción, que un archivo XHTML está formado por elementos y contenido. Los elementos delimitan el contenido y le dan información adicional, que será interpretada por el navegador. Un elemento está compuesto, normalmente, por dos etiquetas. Si el elemento es de tipo vacío sólo será necesaria una, como veremos más adelante.

Aquí tenemos un ejemplo:

```
<p>este texto quiero que sea un párrafo</p>
```

El elemento p está definido por dos etiquetas:  y .

Estas etiquetas están colocadas al inicio y al final de la frase que queremos delimitar como párrafo. La etiqueta  estará al inicio, mientras que la etiqueta  estará al final. Es decir, el signo / dentro de la etiqueta quiere decir que cerramos el elemento.

No todos los elementos se tienen que cerrar de la manera que acabamos de ver. Por ejemplo, cuando queramos introducir una imagen, lo haremos de la manera siguiente:

```
<img src="imgs/imagen1.jpg"/>
```

A esto lo llamaremos, como ya hemos adelantado, elementos vacíos, ya que no tienen contenido dentro de las etiquetas. De hecho, sería lo mismo escribir

```
<img src="imgs/imagen1.jpg"></img>
```

Una vez hemos visto cómo funcionan los elementos HTML, otro aspecto importante que hay que tener en cuenta es que los elementos HTML se pueden "imbricar" los unos con los otros de la manera siguiente:

```
<div> <p>Este texto es un párrafo dentro de una capa</p></div>
```

Aquí vemos cómo el elemento p está dentro del elemento ; por lo tanto, el texto estará afectado por las dos etiquetas. A la hora de cerrar los elementos, se tiene que tener una precaución especial, ya que el primer elemento en ser cerrado será el último elemento que hemos abierto.

Otro aspecto importante que hay que conocer de los elementos es que éstos pueden tener atributos. Los atributos especifican características particulares del elemento. Los atributos se componen del nombre del atributo seguido de un signo = y entre comillas el valor del atributo.

```
<a href="http://www.example.com">Link en un web</a>
```

En este ejemplo, el atributo href tiene como valor "http://www.example.com" y especifica cuál es la dirección a la que queramos enlazar.

**Separación de atributos en documentos CSS**

Más adelante, cuando profundicemos en el estudio de las hojas de estilo, veremos que es mejor tener separados ciertos atributos de los elementos en documentos CSS, tanto para la comprensión global del documento como para conseguir, al mismo tiempo, una mejor valoración de nuestra web por parte de los motores de búsqueda (Google, Yahoo).

Una vez hemos visto cómo funciona el sistema de etiquetado del XHTML, veremos cuáles son sus elementos principales.

## 1.1. Declaración XML

El primer elemento que escribiremos en nuestro documento XHTML es el de declaración de XML. Veremos un ejemplo y lo comentaremos.

Comenzaremos, pues, a construir nuestro documento. Abriremos nuestro editor de texto y escribiremos la línea siguiente.

```
<?xml version="1.0" encoding="UTF-8"?>
```

Guardaremos el documento que acabamos de crear con el nombre de "index.html". Para comprobar cómo interpreta nuestro código un navegador, es decir, cómo se visualiza en un navegador, tendremos que ir hasta el archivo "index.html", hacer clic con el botón derecho y, de las opciones que nos salgan, escoger un navegador (Internet Explorer o Firefox son los más habituales). En caso de que no nos saliera un navegador para escoger, lo podremos hacer de la manera siguiente: abriendo primero el navegador y, después, arrastrándolo adentro de nuestro archivo "index.html".

La declaración XML es un tipo especial de elemento. Como excepción, no cerraremos el elemento de la forma usual, y sirve para indicar al navegador qué versión de XML queremos utilizar. En el atributo "encoding" especificaremos la codificación de caracteres que necesitaremos en nuestro documento. En principio, el valor "UTF-8" sería suficiente para la mayoría de los casos.

**Enlace recomendado**

No profundizaremos en el asunto de las codificaciones, ya que puede ser muy extenso y, en principio, se escapa de los objetivos de este curso. Para conocer más sobre codificaciones, podéis ir a la siguiente dirección: **http://es.wikipedia.org/wiki/Codificaci%C3%B3n_de_caracteres**

## 1.2. Declaración DTD

Con este elemento, se indica el documento público que contiene las reglas de sintaxis y gramática con las cuales se determina si el documento actual es válido o no con respecto a la versión del lenguaje indicada.

Mediante este documento, las aplicaciones de validación verifican que todos los elementos estén correctamente imbricados y que las etiquetas y atributos que contienen son válidos.

Hay distintos DTD (definición de tipo de documento) que establecen los diferentes grados con los cuales un documento XHTML tiene que ajustarse a las reglas de gramática y sintaxis del XML. En concreto, hay tres tipos de DTD que podemos especificar en nuestro documento. A continuación, veremos los más significativos, ya que hay uno que es sólo para cuando utilizamos *frames* y, por lo tanto, lo obviaremos.

Así pues, veremos qué forma tienen los más significativos, de menos a más restrictivos:

- **Transicional**. El DTD transicional permite el uso de algunos elementos y atributos antiguos del HTML que, actualmente, están en desuso.

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

- **Estricto**. Es el DTD más estricto: no soporta ninguna etiqueta antigua y el código tiene que seguir estrictamente los estándares, lo cual nos ayudará a tener nuestra web más estructurada.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
```

En nuestro caso, utilizaremos el estricto.

Si seguimos con nuestro archivo "index.html", el código quedaría de la forma siguiente:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
```

## 1.3. Elemento HTML

Éste es el elemento raíz de nuestro documento. El atributo xmlns indica el espacio nominal para el XHTML. Los espacios nominales quieren decir el conjunto de etiquetas y atributos que forman el vocabulario del XHTML. Los atributos lang y xml:lang especificarán el idioma principal del documento.

No olvidemos que este elemento sí que lo tenemos que cerrar correctamente, al contrario de los dos anteriores.

Aquí podemos ver un ejemplo:

```
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="es"
lang="es"></html>
```

donde "es" es el código ISO del español.

En nuestro ejemplo, añadiremos otra línea y nos quedará así:

```
<?xml version="1.0" encoding="UTF-8"?>
```

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

**Más información sobre el elemento HTML**

Para más información podéis ir al enlace siguiente:

**http://www.w3.org/International/articles/language-tags/Overview.es.php**

## 1.4. Elemento HEAD

El elemento head contiene información sobre el documento actual, como el título, palabras clave que pueden ser de utilidad para motores de búsqueda y otros datos que no se consideran parte del contenido del documento. La información que contiene también se puede llamar meta-información. *Meta* significa 'información sobre'.

A continuación, tenemos un ejemplo con algunos de los meta más comunes:

```
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
<meta name="author" content="Pere Barnola" />
<meta name="description" content="La gastronomía en la Garrotxa" />
<meta name="keywords" content="gastronomia, garrotxa" />
</head>
```

Veamos diferentes elementos meta.

Estos elementos sirven para definir información adicional en nuestro documento, como autor, descripción del contenido de la página y palabras clave.

Si lo añadimos a nuestro ejemplo, nos quedará lo siguiente:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="es" lang="es">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
<meta name="author" content="Pere Barnola" />
<meta name="description" content="La gastronomía en la Garrotxa" />
<meta name="keywords" content="gastronomia, garrotxa" />
</head>
</html>
```

## 1.5. Elemento TITLE

El elemento title es de gran importancia, pues sirve para identificar los contenidos de un documento. A causa de que los usuarios a menudo consultan documentos fuera de contexto, tendríamos que proporcionar títulos ricos en contexto.

Así, en lugar de utilizar un título como "introducción", que no proporciona mucha información del contexto, tendríamos que cambiarlo por un título del estilo "Introducción a la gastronomía de la Garrotxa".

Veamos un ejemplo:

```
<title>Introducción a la gastronomía de la Garrotxa</title>
```

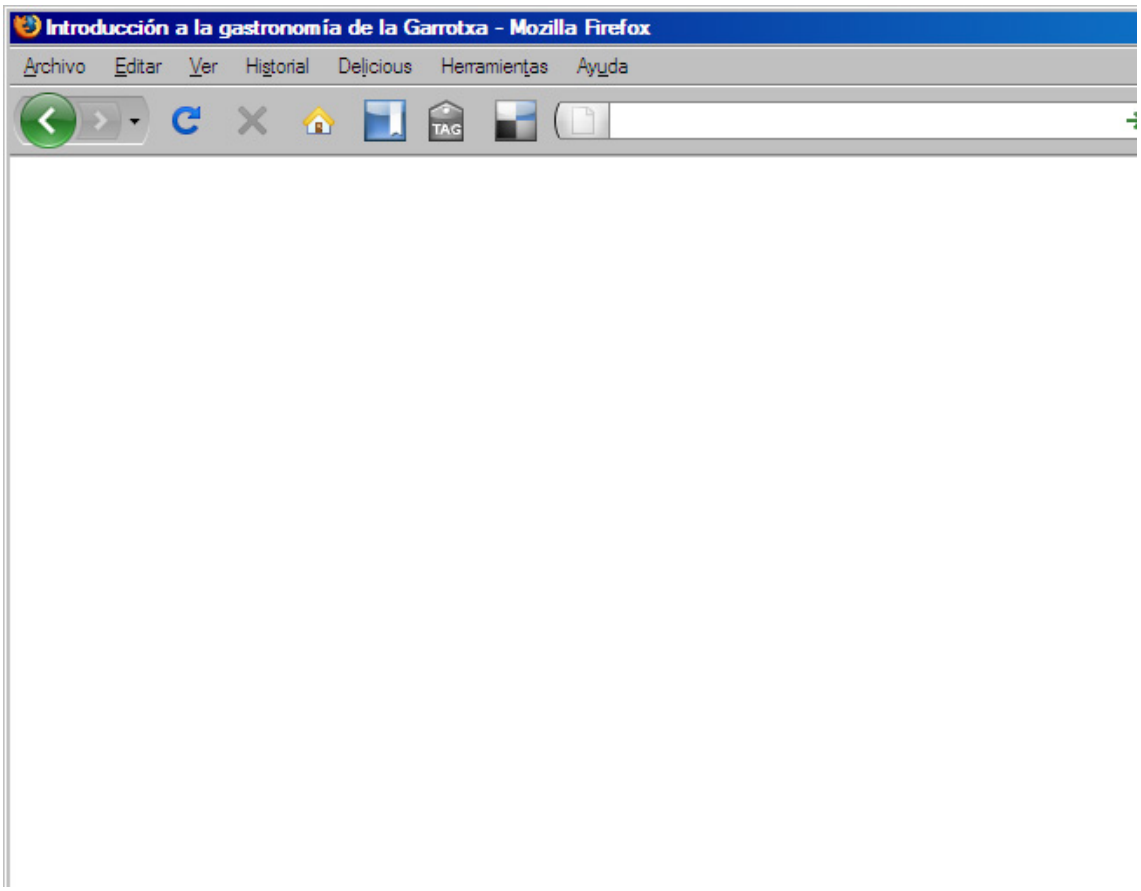
Además, buscadores como Google o Yahoo dan más prioridad al title a la hora de dar importancia a las webs en sus resultados que a otros elementos.

Hay que tener en cuenta que el elemento title va dentro del elemento head que hemos visto antes.

Añadimos este elemento a nuestro documento:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="es" lang="es">
<head>
<title>Introducción a la gastronomía de la Garrotxa</title>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
<meta name="author" content="Pere Barnola" />
<meta name="description" content="La gastronomía en la Garrotxa" />
<meta name="keywords" content="gastronomia, garrotxa" />
</head>
</html>
```

Si probamos el index.html en un navegador, tendríamos que ver una cosa parecida a lo siguiente:




Como apreciamos en la imagen, la página aparece en blanco porque todavía no tiene ningún contenido, pero podemos observar cómo, en la parte superior del navegador (con fondo de color azul), sale nuestro título. ¡Vamos por buen camino!

## 1.6. Elemento BODY

Si el elemento head contenía toda la información sobre el documento, el elemento body contiene todo el contenido. Nos tenemos que imaginar este elemento como una hoja en blanco sobre la cual aparecerá todo el contenido: texto, imágenes, colores, gráficos...

Más adelante, veremos todos los elementos que pueden estar en el body.

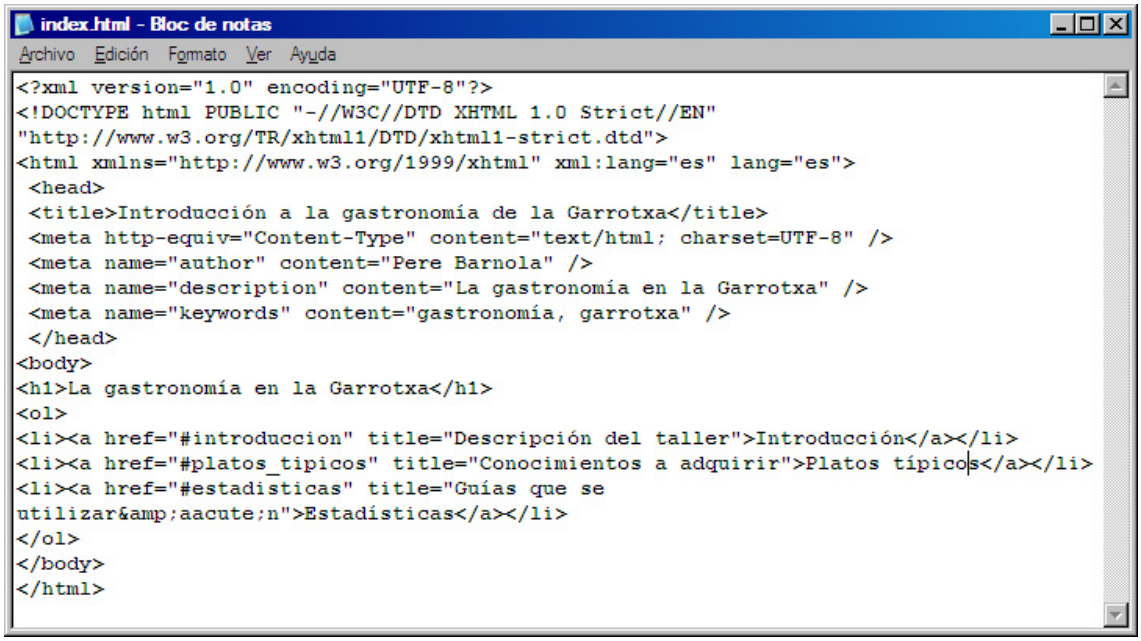
```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="es" lang="es">
<head>
<title>Introducción a la gastronomía de la Garrotxa</title>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
<meta name="author" content="Pere Barnola" />
<meta name="description" content="La gastronomía en la Garrotxa" />
<meta name="keywords" content="gastronomia, garrotxa" />
</head>
<body>
</body>
```

```
</html>
```

## 1.7. Un documento XHTML básico

A lo largo de los puntos que hemos estado viendo, hemos configurado un documento XHTML. Este documento sería un documento XHTML básico, pero sin ningún tipo de contenido.

Añadiremos ahora una pequeña parte de contenido para adelantar lo que veremos en los puntos siguientes:




Disponéis del código en el archivo **ejemplo01.html**

En primer lugar, podemos ver los elementos xml y DOCTYPE, necesarios, como hemos visto, para determinar cuáles serán las características esenciales de nuestro documento.

Seguidamente, el elemento html que abre y cierra toda la estructura de elementos. Dentro de este elemento encontramos dos más: head y body. Recordemos que el elemento head nos ofrece información sobre el documento, como el título o su descripción, y el elemento body es el que contendrá todo el contenido de nuestro documento.

Adelantándonos un poco a lo que veremos más adelante, dentro del elemento body vemos otros elementos que nos permitirán etiquetar nuestro contenido.

Si visualizamos el archivo en un navegador, tendríamos que ver algo parecido a lo siguiente:


Lo primero que vemos es el elemento h1 que equivale al encabezamiento de tipo 1, seguido de un elemento ol que equivale a una lista ordenada, elementos que veremos acto seguido.

## 2. Etiquetas y elementos

Veremos en este punto los elementos básicos para etiquetar nuestro contenido XHTML.

### 2.1. Encabezamientos

Cuando queramos que un texto sea un titular o texto destacado dentro del contenido, utilizaremos los encabezamientos.

Los encabezamientos se pueden generar mediante 6 etiquetas diferentes h1, h2, h3, h4, h5 y h6 que tienen una jerarquía de más a menos importancia. Los encabezamientos se tienen que utilizar de forma jerárquica y sin saltarnos ningún nivel. Es decir, después del h1, vendrá el h2 y así consecutivamente, sin que podamos pasar por ejemplo del h1 al h6 de forma directa.

Un ejemplo de uso correcto de encabezamientos sería:

```
<h1> Encabezamiento con elemento h1</h1>
<h2> Encabezamiento con elemento h2</h2>
```

El ejemplo siguiente sería incorrecto:

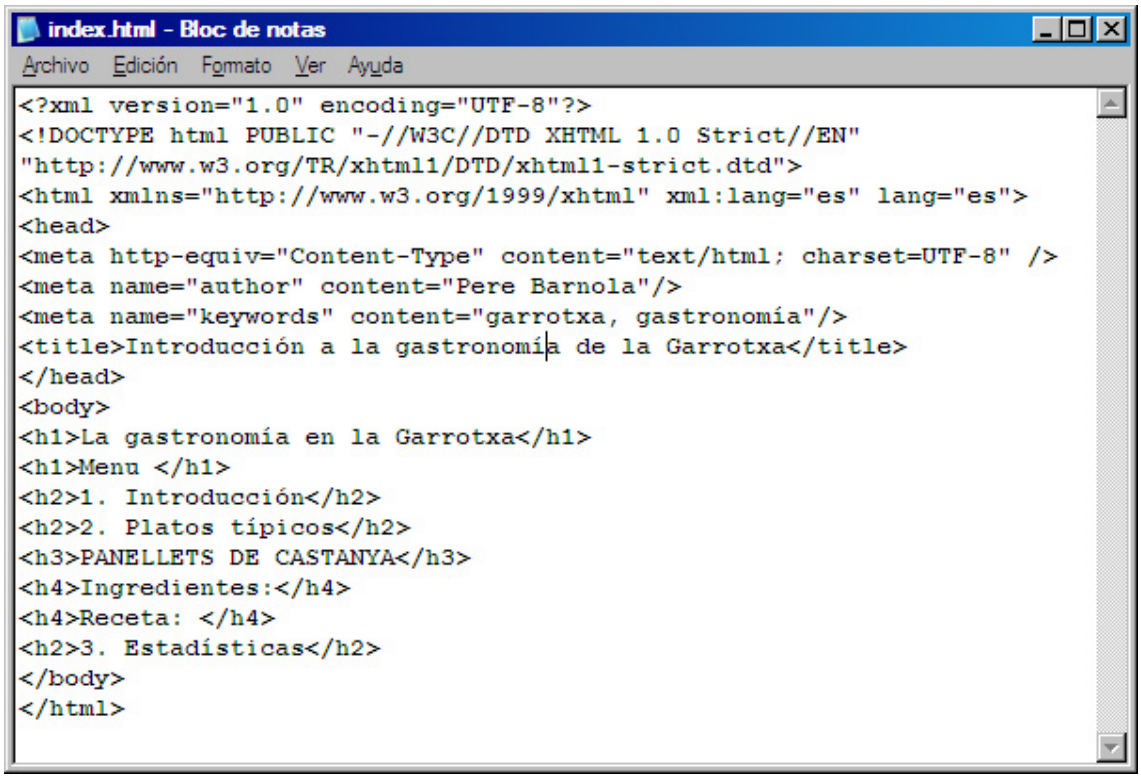
```
<h1> Encabezamiento con elemento h1</h1>
<h6> Encabezamiento con elemento h6</h6>
```

Por defecto, los encabezamientos de mayor importancia tienen un cuerpo de letra mayor, aunque, como veremos en el capítulo dedicado a las hojas de estilo, podremos definir un estilo y un tipo de letra específico para cada encabezamiento.

Otro punto muy interesante, con vistas al buen posicionamiento de nuestra página web en los buscadores más utilizados (Google, Yahoo) es que éstos dan más importancia a un texto etiquetado como encabezamiento que a otro de otro tipo.

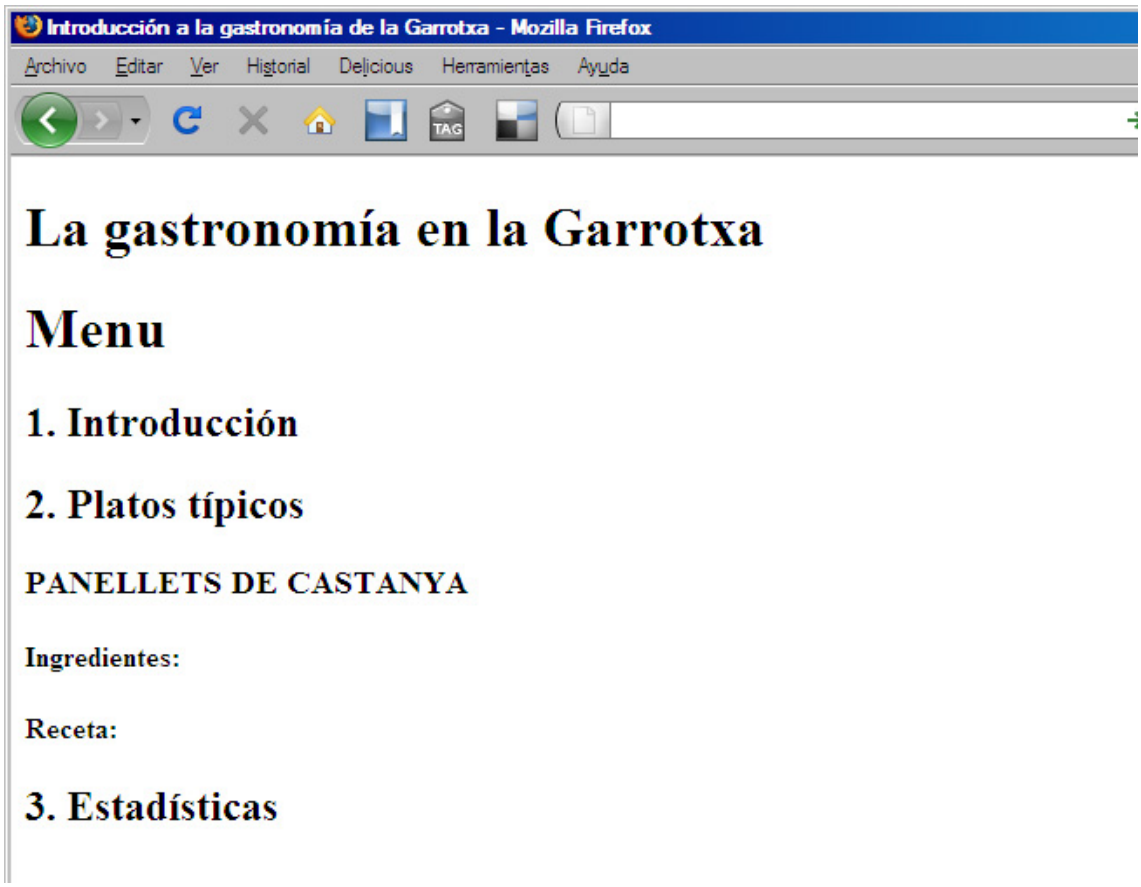
Hay que decir, además, que los encabezamientos permiten saber a los navegadores cuándo empieza y cuándo acaba una nueva sección, lo cual facilita la experiencia de usuario.

Siguiendo nuestro ejemplo, incluiremos los elementos de encabezamiento. El código que nos quedaría sería el siguiente:




Disponéis del código en el archivo **ejemplo02.html**

Si lo visualizamos en un navegador, nos quedará así:




Podemos apreciar cómo, según los tipos de encabezamiento escogido, el tipo de letra es mayor o más pequeño.

## 2.2. Párrafos

Los párrafos son grupos de texto delimitados por las etiquetas  y . Cada párrafo queda delimitado visualmente por un salto de línea. Haciendo el paralelismo, sería lo mismo que un párrafo de un artículo o un libro.

Dentro de un elemento de párrafo no podremos poner otro elemento de tipo párrafo, siempre que estemos utilizando XHTML de tipo "strict". Es decir, no los podremos "imbricar".

**Ejemplo de código de párrafo**

```
<p>Este texto es un párrafo. Básicamente lo utilizaremos para delimitar grupos de texto</p>
```

Incluiremos elementos de párrafo en nuestro documento y nos quedará así:

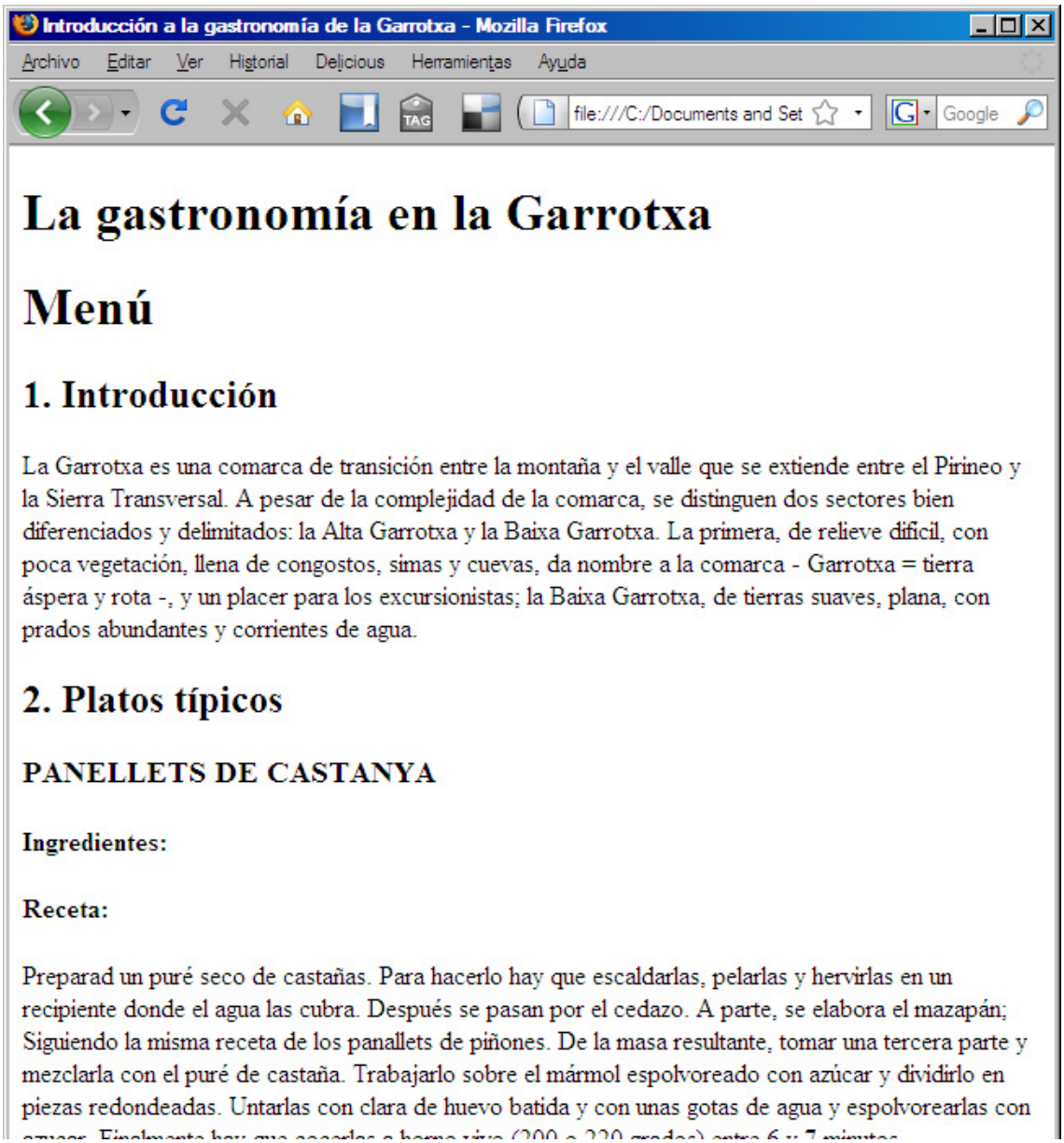
index.html - Bloc de notas

Archivo Edición Formato Ver Ayuda

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="es" lang="es">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
<meta name="author" content="Pere Barnola"/>
<meta name="keywords" content="garrotxa, gastronomía"/>
<title>Introducción a la gastronomía de la Garrotxa</title>
</head>
<body>
<h1>La gastronomía en la Garrotxa</h1>
<h1>Menú</h1>
<h2>1. Introducción</h2>
<p>La Garrotxa es una comarca de transición entre la montaña y el valle que
se extiende entre el Pirineo y la Sierra Transversal. A pesar de la
complejidad de la comarca, se distinguen dos sectores bien diferenciados y
delimitados: la Alta Garrotxa y la Baixa Garrotxa. La primera, de relieve
difícil, con poca vegetación, llena de congostos, simas y cuevas, da nombre a
la comarca - Garrotxa = tierra áspera y rota -, y un placer para los
excursionistas; la Baixa Garrotxa, de tierras suaves, plana, con prados
abundantes y corrientes de agua.</p>
<h2>2. Platos típicos</h2>
<h3>PANELLETS DE CASTANYA</h3>
<h4>Ingredientes:</h4>
<h4>Receta: </h4>
<p>Preparad un puré seco de castañas. Para hacerlo hay que escaldarlas,
pelarlas y hervirlas en un recipiente donde el agua las cubra. Después se
pasan por el cedazo. A parte, se elabora el mazapán; Siguiendo la misma
receta de los panallets de piñones.
De la masa resultante, tomar una tercera parte y mezclarla con el puré de
castaña. Trabajarlo sobre el mármol espolvoreado con azúcar y dividirlo en
piezas redondeadas. Untarlas con clara de huevo batida y con unas gotas de
agua y espolvorearlas con azucar. Finalmente hay que cocerlas a horno vivo
(200 o 220 grados) entre 6 y 7 minutos.</p>
<h2>3. Estadísticas</h2>
<p>Documento conforme W3C <a href="http://validator.w3.org/"
title="Código revisado con el validador del W3C">XHTML 1.0</a>|
<a href="http://jigsaw.w3.org/css-validator/" title="Hojas de
estilo revisadas con el validador del W3C">CSS 2.0</a>|<a
href="http://www.tawdis.net/taw3/online" title="Nivel de
adecuación doble A, conforme a las directrices del Web
Accessibility Initiative">WAI AA</a></p>
</body>
</html>
```

Disponéis del código en el archivo **ejemplo03.html**

Vemos cómo se visualizará este código en el navegador:




## 2.3. Enlaces

Uno de los elementos básicos a la hora de crear páginas web son los enlaces. Los enlaces son los elementos que nos permiten navegar por la web, es decir, ir a otras páginas de una misma web, o a páginas de webs externas. También podemos enlazar a diferentes partes de un mismo documento.

El elemento utilizado para crear vínculos es a. Aquí tenemos un ejemplo:

```
<a href="http://www.webexterna.com">Enlace a una web externa</a>
```

El valor del atributo href nos indica hacia dónde apuntará nuestro enlace. En este caso, estamos enlazando a una página web externa.

También es posible enlazar a otras páginas dentro de nuestra web. En este caso, el código sería así:

```
<a href="href="index.html">Enlace interno en el índice de nuestra web</a>
```

En este caso, estamos creando un enlace en el documento index.html dentro de nuestra estructura de archivos.

El valor del atributo href ha de ser la ruta del documento al que quiero enlazar.

También es posible enlazar a una parte concreta de nuestro documento. Para eso, necesitaremos dos valores. Uno para indicar la parte enlazable y el otro para indicar la posición a la cual queremos ir.

Así pues, en nuestro código tenemos que tener, por una parte, marcado con el elemento a, la parte enlazable:

```
<a href="#menu">Ir al menú</a>
```

Vemos cómo el atributo href comienza por el signo #. Eso quiere decir que estamos enlazando a una parte concreta dentro de nuestro documento. Concretamente, nos moveremos hacia la parte del documento que contenga un elemento con el atributo id igual al que está especificado después del #.

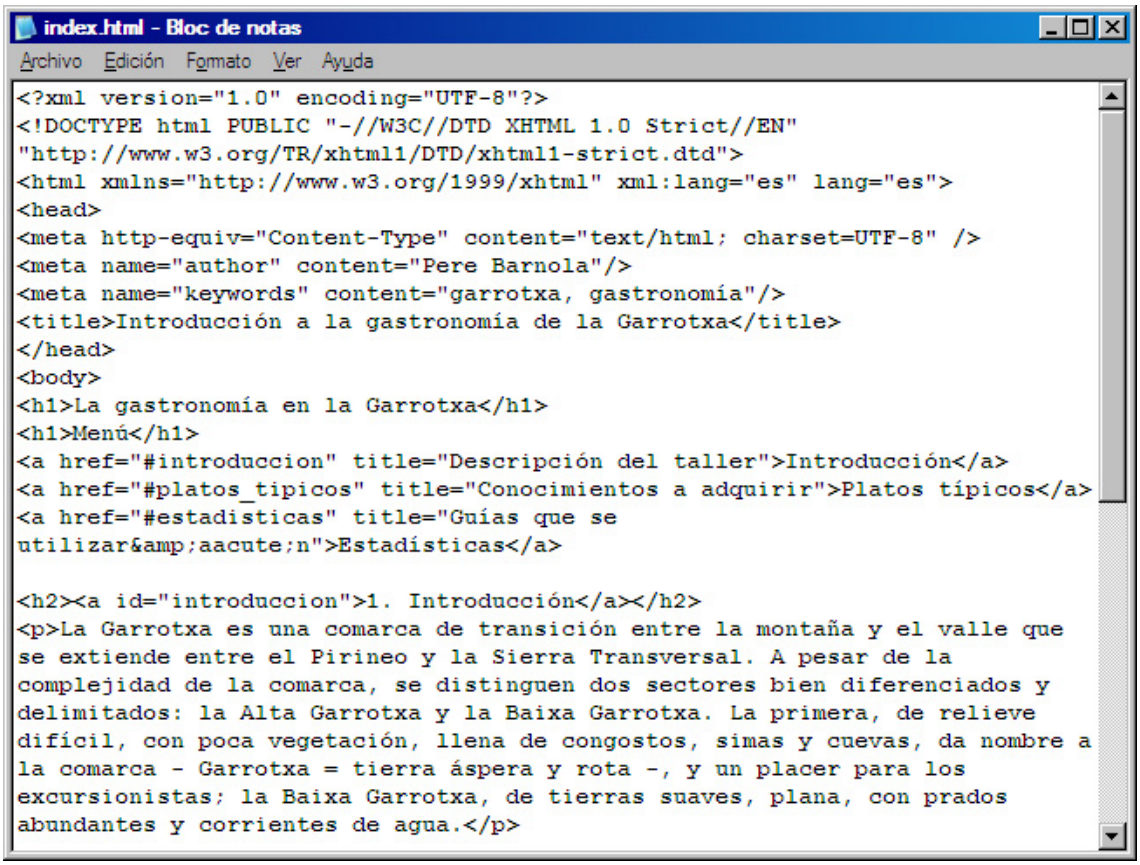
Por tanto, para que el enlace funcione, nos tendríamos que asegurar de tener un elemento dentro del documento con el identificador.

Del estilo:

```
<a id="menu" >Menú de nuestra web</a>
```

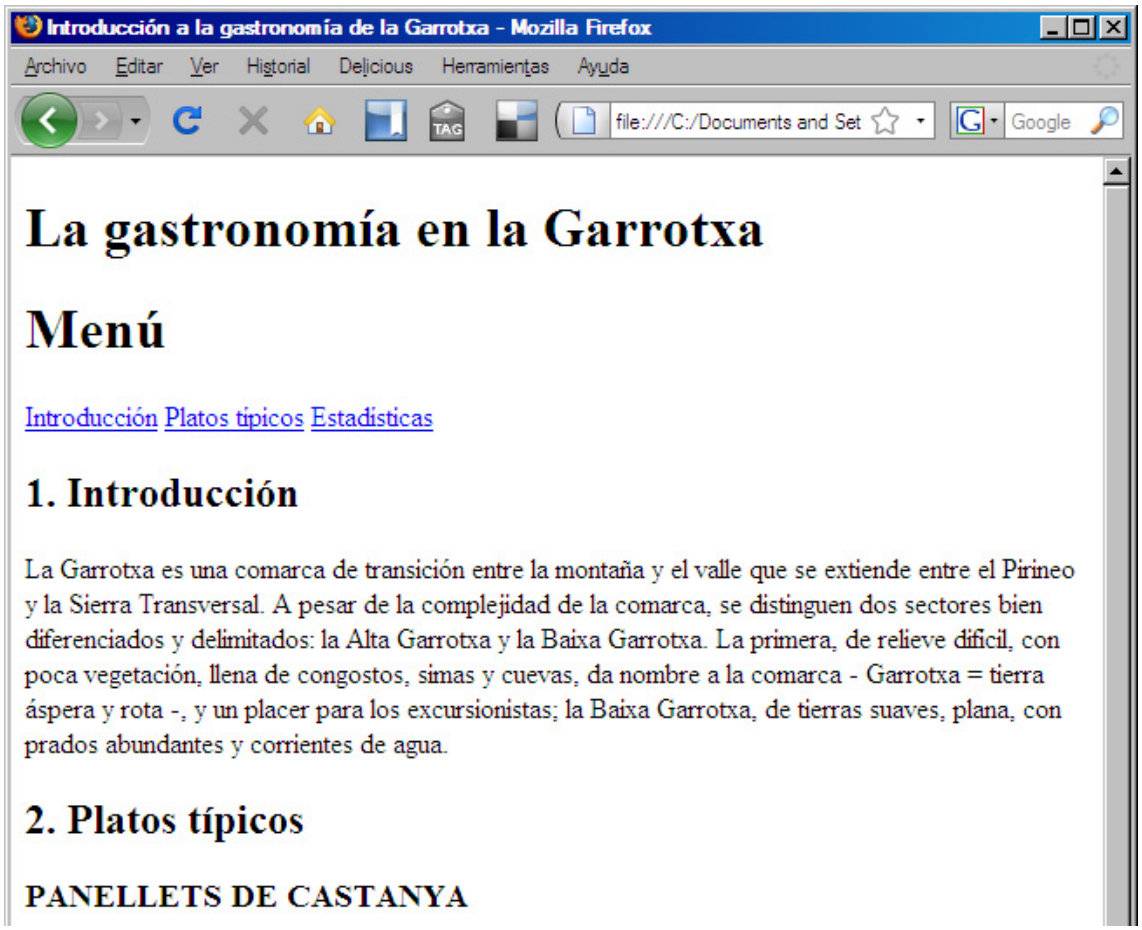
Vemos cómo el ID es aquél al que hacíamos referencia en el enlace. Por lo tanto, cuando hacemos clic en el enlace, nos dirigiremos a la parte del documento especificada.

Si incluimos enlaces en nuestro documento de ejemplo, el código quedaría así:




Disponéis del código en el archivo **ejemplo04.html**

Vemos el impacto de este código en el navegador.




**Ejemplo**

Como podemos apreciar por la imagen, los enlaces se diferencian de otros elementos porque, a la hora de visualizar el documento, éstos salen subrayados y de color azul. Si hacemos clic con el ratón, veremos cómo los enlaces internos nos redireccionan a la parte de contenido marcada, y si hacemos clic en los externos, vamos a la página especificada.

## 2.4. Imágenes

Introducir imágenes en nuestro documento XHTM puede ayudarnos a explicar más fácilmente nuestra información.

Para incluir imágenes en nuestro documento XHTML, utilizaremos el elemento img. Será necesario especificar una serie de atributos. Lo más importante es el atributo src que sirve para indicar la dirección o lugar en el cual se encuentra la imagen que queremos incluir en el documento.

Así pues, la sintaxis quedará de la forma siguiente:

```
<img src="foto_garrotxa.gif" alt="imagen del cráter de Santa Margalida"
width="376" height="226" />
```

**Formatos GIF y JPG**

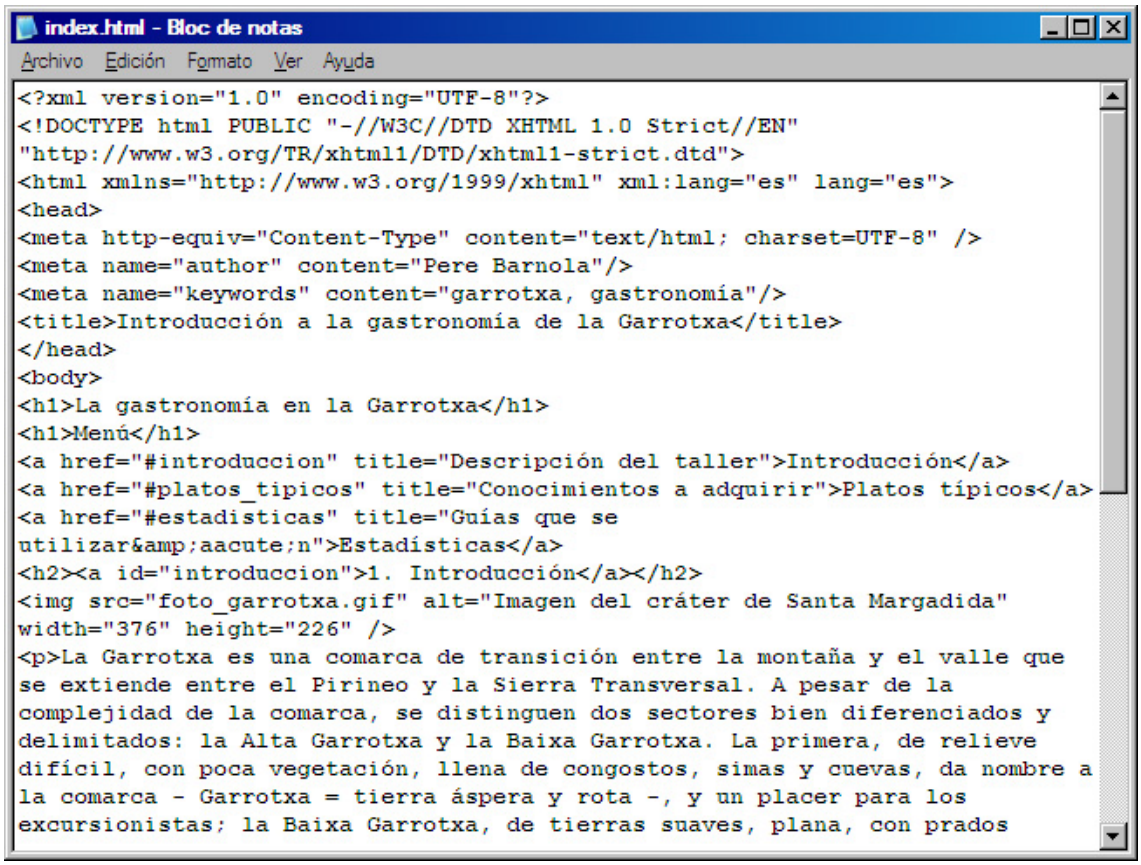
Hay que indicar que los formatos de imagen reconocidos universalmente por todos los navegadores son el GIF y el JPG. La utilización de otros tipos de imágenes en nuestro documento podría comportar que, a la hora de visualizarlas en el navegador, éstas no aparezcan ni se muestren de forma correcta. De todos modos, la mayoría de

navegadores ya aceptan formatos de imágenes con mejores prestaciones, como podrían ser el PNG y el SVG.

Otro atributo importante del elemento img es el alt. Este atributo sirve para indicar una descripción alternativa en la imagen para usuarios que no puedan visualizarla. Teniendo en cuenta que hay usuarios que tienen deshabilitada la opción de visualización de imágenes mientras navegan, o simplemente no las pueden visualizar (como colectivos de discapacidades visuales), este atributo es importante para hacer que nuestro documento sea más accesible.

Los atributos width y height no son obligatorios, y si no los especificamos, se cogen por defecto los valores de la imagen original.

Incluiremos una imagen en nuestro documento a modo de ejemplo:




Disponéis del código en el archivo **ejemplo05.html**

Vemos cómo se visualizará la imagen en nuestro navegador:


## 2.5. Listas

Una lista es una serie de elementos estructurados que nos permite organizar mejor nuestro texto. Hay tres tipos de listas: numeradas, sin numerar y de definición.

### 2.5.1. Listas numeradas

Las listas numeradas representarán sus elementos en el orden que ocupan en la lista. Para construir una lista numerada, necesitaremos el elemento ol (ordered list), que marcará el inicio y el final de una lista ordenada. Cada elemento de la lista estará marcado por el elemento li (list item).

Aquí vemos un ejemplo:

```
<ol>
<li><a href="#introduccion" >Introducción</a></li>
<li><a href="#platos_tipicos" >Platos típicos</a></li>
<li><a href="#estadisticas" >Estadísticas</a></li>
</ol>
```

Añadiremos una lista numerada en nuestro documento para ver el efecto:

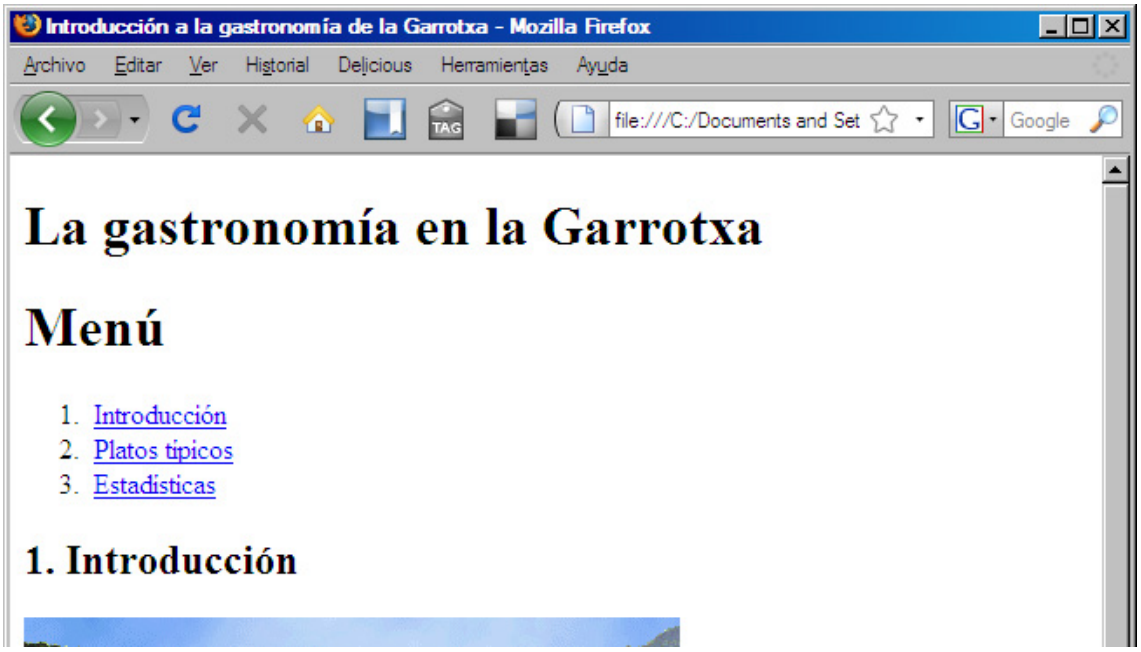
index.html - Bloc de notas

Archivo Edición Formato Ver Ayuda

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="es" lang="es">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
<meta name="author" content="Pere Barnola"/>
<meta name="keywords" content="garrotxa, gastronomía"/>
<title>Introducción a la gastronomía de la Garrotxa</title>
</head>
<body>
<h1>La gastronomía en la Garrotxa</h1>
<h1>Menú</h1>
<ol>
<li><a href="#introduccion" title="Descripción del
taller">Introducción</a></li>
<li><a href="#platos_tipicos" title="Conocimientos a adquirir">Platos
típicos</a></li>
<li><a href="#estadisticas" title="Guías que se
utilizar&amp;aacute;n">Estadísticas</a></li>
</ol>
<h2><a id="introduccion">1. Introducción</a></h2>
```

Disponéis del código en el archivo **ejemplo06.html**

Veamos cómo visualiza el navegador este nuevo elemento:




### 2.5.2. Listas sin numerar

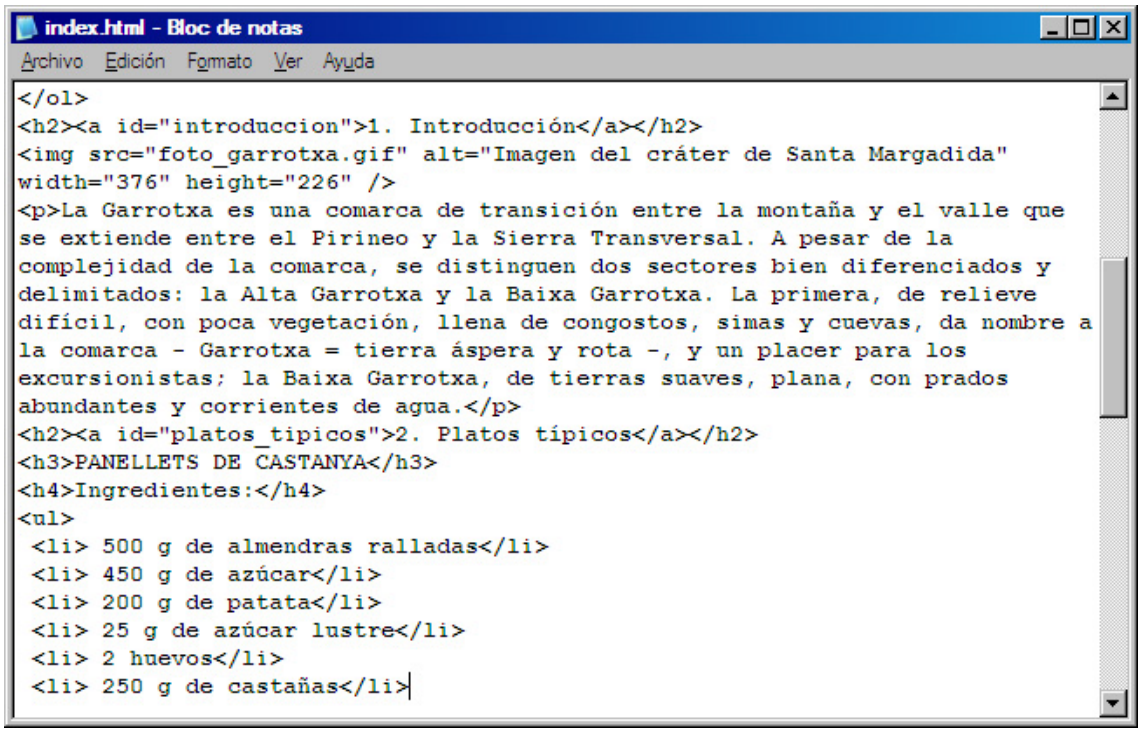
Las listas sin numerar, como indica el nombre, son también listas de elementos. La diferencia con las listas numeradas es que, en lugar de marcar los elementos con su orden dentro de la lista, los elementos se marcan con "puntos" (típicamente bolas negras).

El elemento para construir la lista sin numerar será ul (unordered list).

Un ejemplo de lista sin enumerar sería el siguiente:

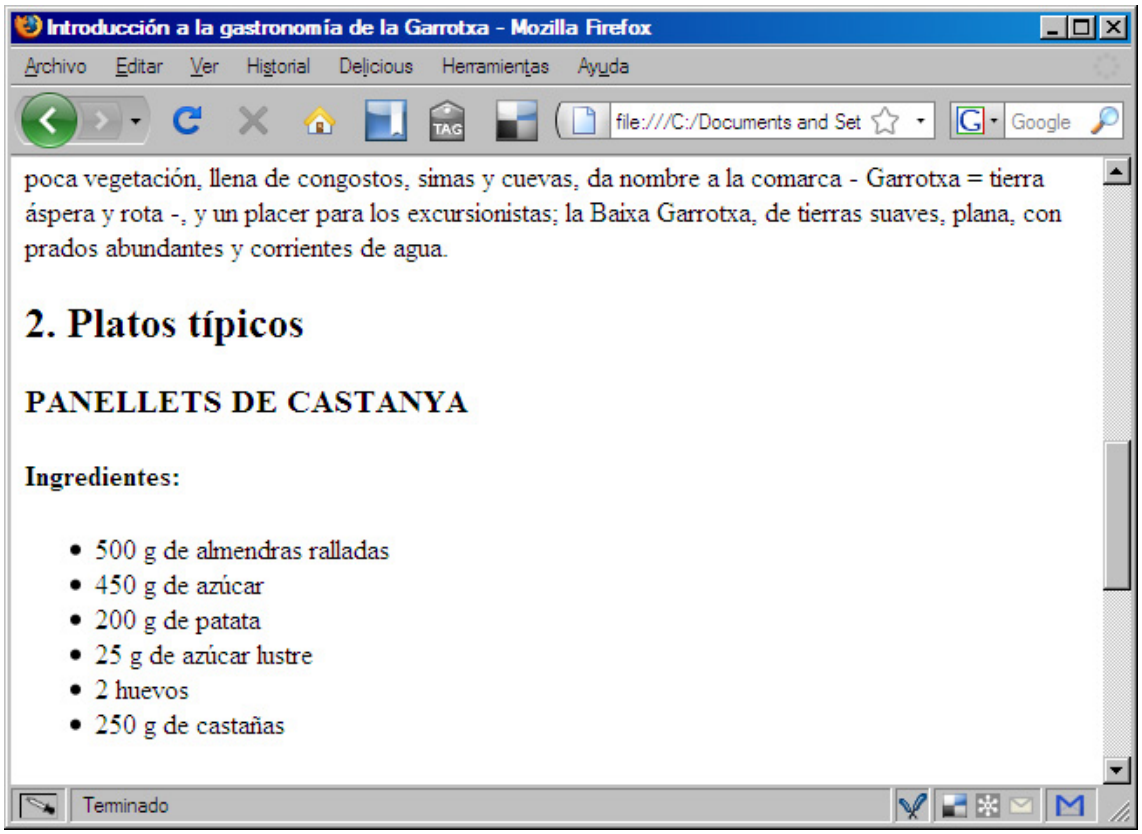
```
<ul>
<li> 500 gr de almendras ralladas</li>
<li> 450 gr de azúcar</li>
<li> 200 gr de patata</li>
<li> 25 gr de azúcar lustre</li>
<li> 2 huevos</li>
<li> 250 gr de castañas</li>
</ul>
```

Incluimos ahora una lista sin numerar en nuestro código:




Disponéis del código en el archivo **ejemplo07.html**

En el navegador se visualizaría así:




### 2.5.3. Listas de definición (listas diccionario)

Las listas de definición se utilizan para formatear un conjunto de palabras con sus correspondientes descripciones. Básicamente, son para tratar contenido de tipo diccionario (parejas de palabra, descripción). Estos tipos de listas se engloban con las etiquetas <dl> y </dl> (definition list). Mientras que para la palabra a describir se utiliza el elemento dt (definition-list term), para la definición dd (definition-list definition).

Con un ejemplo lo veremos más claro:

```
<dl>
    <dt>Perro</dt>
    <dd>animal de compañía</dd>
    <dt>Coche</dt>
    <dd>vehículo de autolocomoción</dd>
    </dl>
```

En este caso, no lo incluimos en el documento de ejemplo, porque su funcionamiento es similar al de las listas no numeradas. Pero si queréis, podéis hacer vosotros mismos la prueba incluyendo el código de ejemplo en el XHTML.

## 2.6. Tablas

Las tablas nos permiten representar y ordenar cualquier elemento de nuestro contenido en diferentes filas y columnas, de manera tal que grandes cantidades de información se puedan representar de forma rápida y fácil. Es decir, las tenemos que utilizar para marcar "contenido tabular".

**Sobre el uso de las tablas**

> Un uso bastante extendido, años atrás, consistía en maquetar el contenido de la web mediante tablas, ya que permitía estructurar y situar fácilmente todos los elementos de la web. El uso de las tablas tiene que estar limitado a su función de representar grandes cantidades de información tabular de una manera ordenada. Utilizar las tablas para "maquetar" o "estructurar" nuestra web es del todo desaconsejable, ya que se hace bastante complicado realizar modificaciones posteriores en nuestro diseño de la página. No estamos dando información semántica del contenido, y además introducen mucho código dentro del contenido, lo cual hará que para los buscadores el contenido real tenga menos importancia dentro del documento.

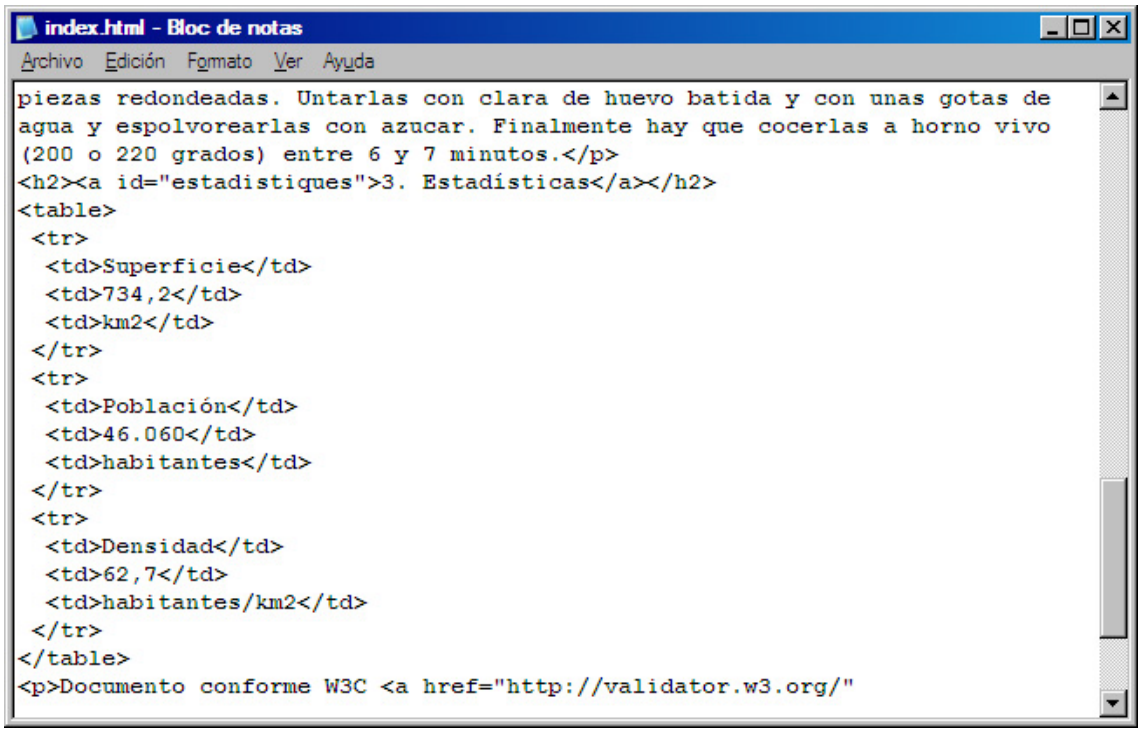
El contenido de una tabla tiene que estar dentro de las etiquetas <table> y </table>.

Cada fila de la tabla se indica mediante las etiquetas <tr> y </tr>. Las etiquetas <td> y </td> sirven para indicar las celdas individuales dentro de cada fila. Las columnas se calcularán automáticamente según el número de celdas que haya en cada fila.

Un código de una tabla podría ser como el que sigue:

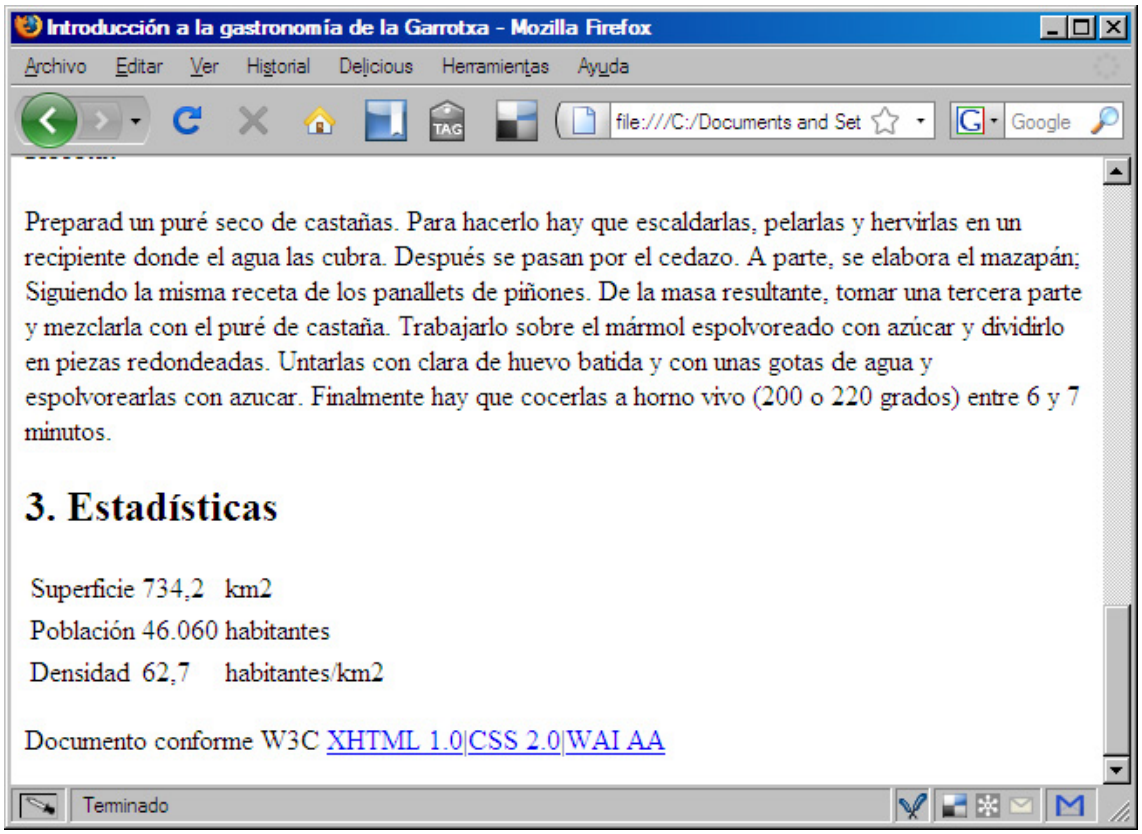
```
<table>
    <tr>
    <td>Superficie</td>
    <td>734,2</td>
    <td>km2</td>
    </tr>
    <tr>
    <td>Población</td>
    <td>46.060</td>
    <td>habitantes</td>
    </tr>
    <tr>
    <td>Densidad</td>
    <td>62,7</td>
    <td>habitantes/km2</td>
    </tr>
    </table>
```

Añadiremos una tabla en nuestro documento de ejemplo:




Disponéis del código en el archivo **ejemplo08.html**

Veríamos la tabla de esta manera:




## 2.7. Capas

Las capas sirven para delimitar un fragmento de documento que puede contener en el interior cualquier otro elemento, incluyendo otras capas. Una de las aplicaciones más útiles de las capas es la de dividir y estructurar nuestro contenido en sus partes principales, como cabecera del documento, menú, contenido y pie del documento. De esta manera, podremos estructurar de forma sencilla el documento y nos será mucho más fácil poder dar estilo a cada una de las partes de forma independiente, como veremos más adelante.

La imagen siguiente ilustra una posible estructuración del contenido con capas:


Las capas se representan con el elemento div. Aquí podemos ver un ejemplo muy sencillo:

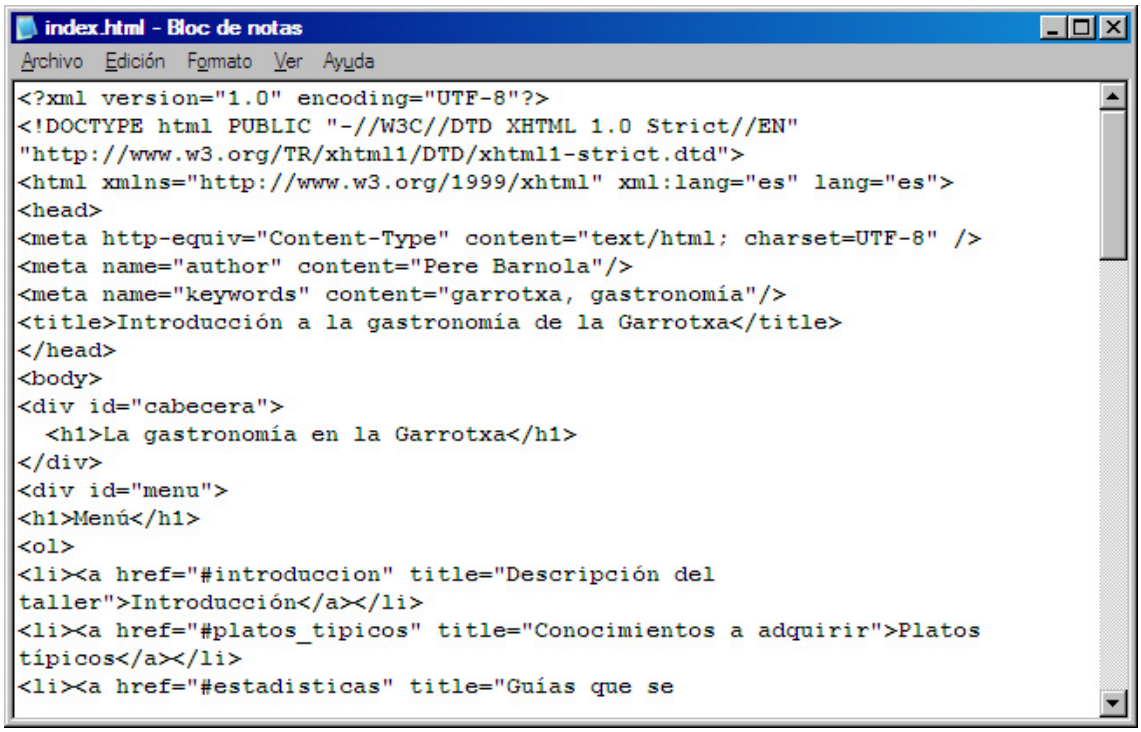
```
<div>
<h2> Encabezamiento de tipo 2</h2>
<p>Texto descriptivo</p>
</div>
```

Para identificar la capa y darle un cierto tratamiento visual, tendremos que relacionarla con un identificador único mediante el atributo "id". Es aconsejable utilizar ID para describir el contenido que engloba la capa, con el fin de facilitar la comprensión posterior de la estructura.

Siguiendo nuestro ejemplo anterior:

```
<div id="cabecera">
<h2>Encabezamiento de tipo 2</h2>
<p>Texto descriptivo</p>
</div>
```

Si estructuramos nuestro documento con capas, el código quedaría de la forma siguiente:




Disponéis del código en el archivo **ejemplo09.html**

**Tratamiento visual de capas**

La incorporación en nuestro código de las capas no tiene, hoy por hoy, ninguna repercusión visual. Más adelante, con el estudio de los CSS, veremos cómo dar tratamiento visual a las capas y otros elementos.

## 2.8. Otros

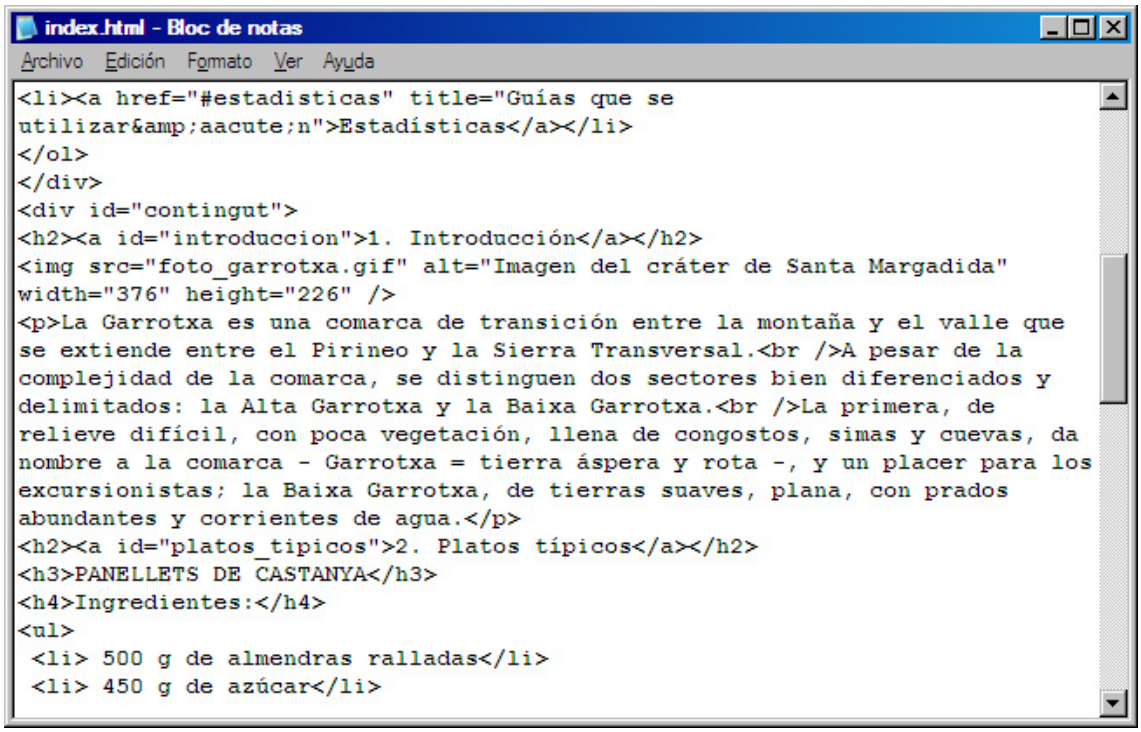
Acabamos de ver algunos de los elementos básicos para el marcaje de documentos en XHTML. A continuación, repasaremos otros elementos también bastante utilizados en XHTML.

### 2.8.1. Elemento BR

Este elemento sirve para forzar un salto de línea en nuestro documento.

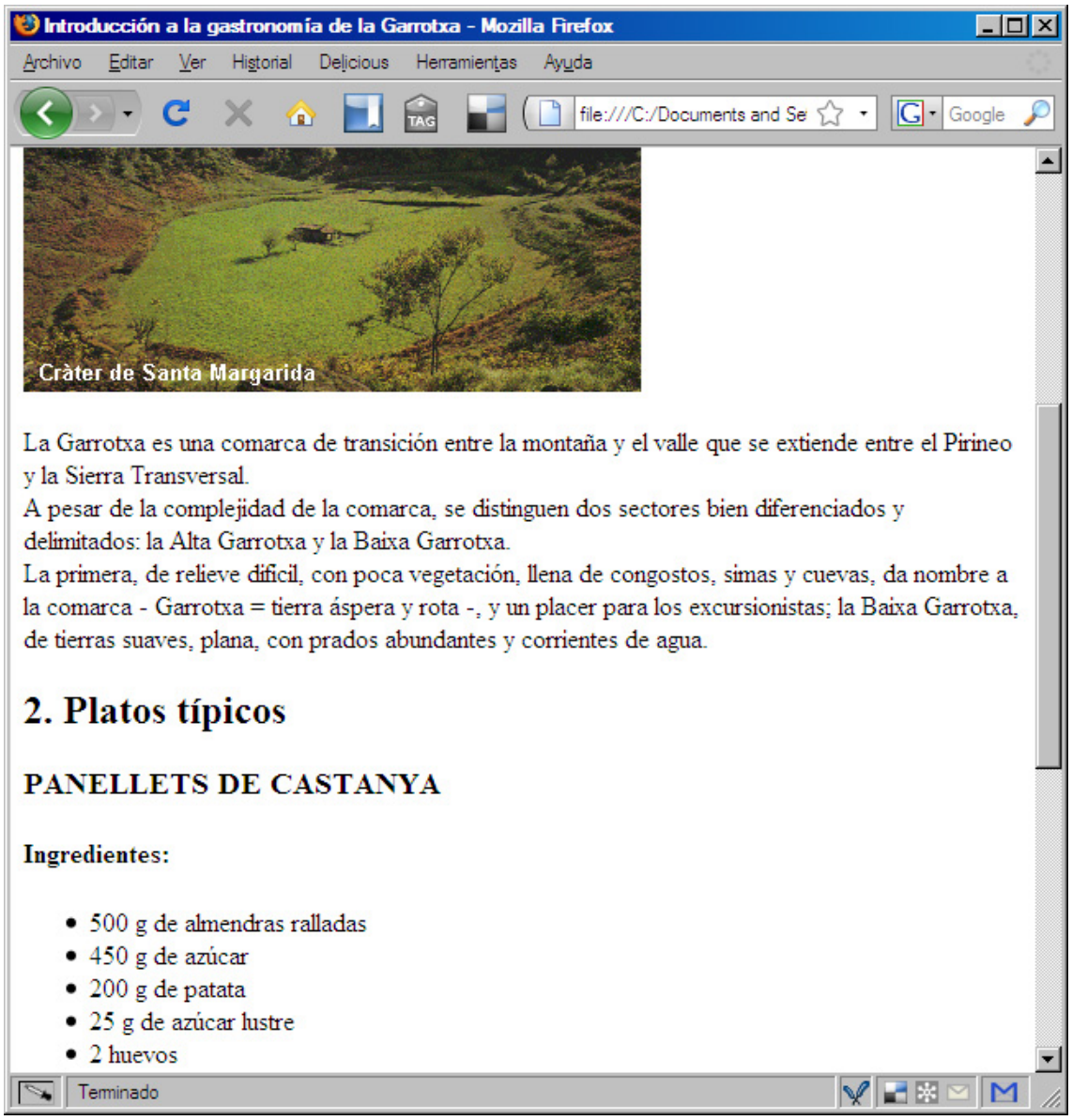
Se ha de ir con cuidado con el uso de este elemento, ya que muchas veces se utiliza de forma errónea como elemento de estilo para separar contenidos.

Incluiremos una etiqueta <br /> en nuestro documento:




Disponéis del código en el archivo **ejemplo10.html**

Aquí vemos cómo afectan a estos elementos en la visualización del documento:

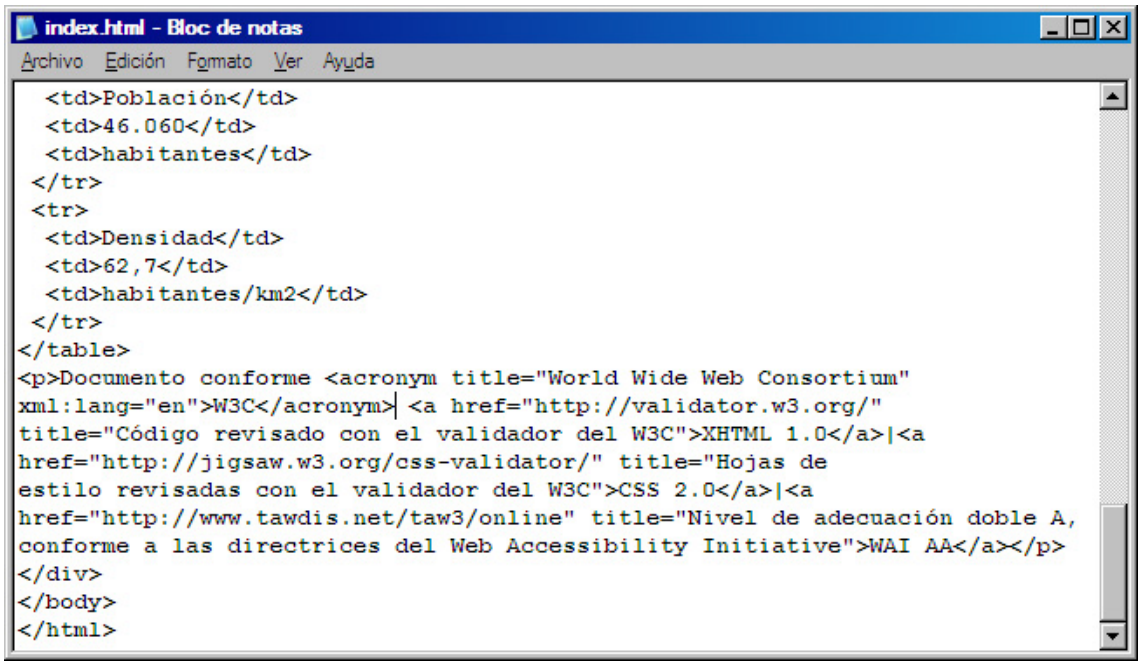



### 2.8.2. Elemento ACRONYM

Este elemento se utiliza para marcar acrónimos, es decir, palabras formadas por las sigles o partes de otras palabras. En el atributo title especificaremos el texto completo. Eso sí, no nos tenemos que olvidar de especificar el lenguaje del acrónimo si éste es diferente al del documento. Dentro de los tags especificaremos el acrónimo tal como podemos ver en el ejemplo siguiente.

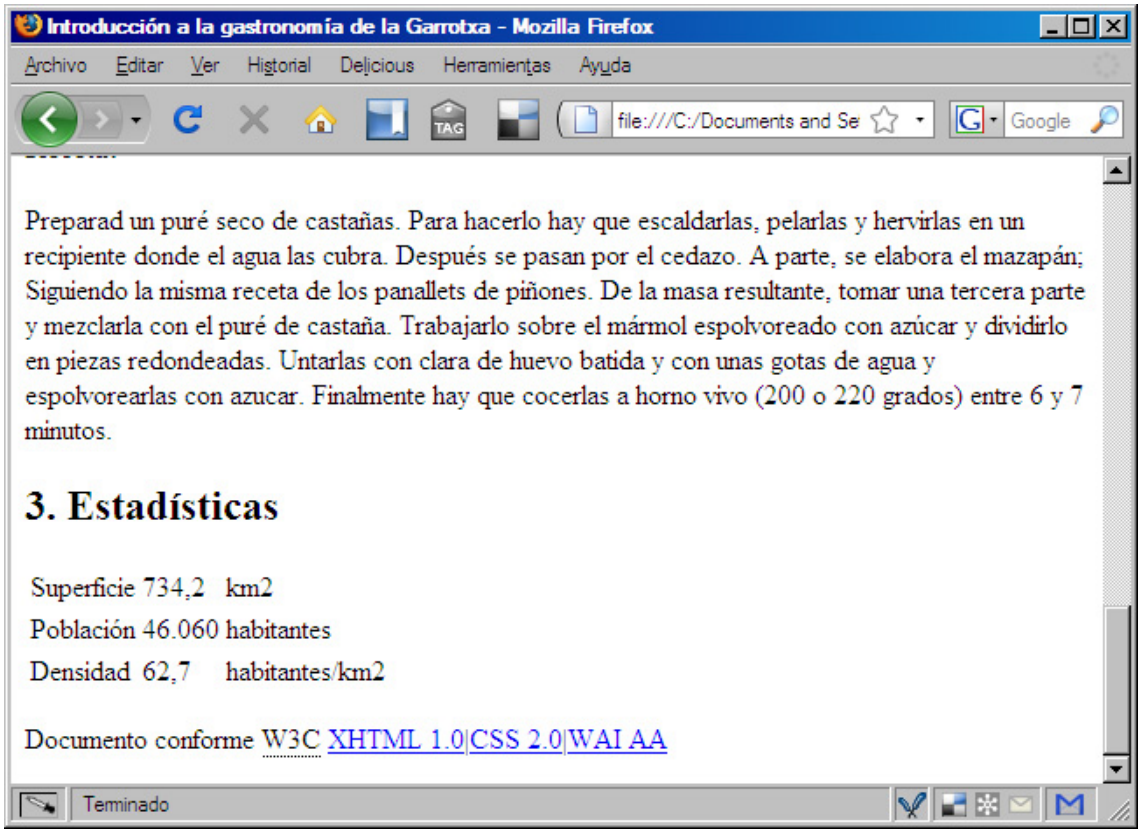
```
<acronym title="World Wide Web Consortium" xml:lang="en">W3C</acronym>
```

Vemos cómo quedará nuestro código de ejemplo con un elemento *acrónimo*:




Disponéis del código en el archivo **ejemplo11.html**

En la siguiente imagen, podemos ver el efecto del elemento *acronym* cuando tenemos el ratón sobre este elemento, en nuestro caso, el texto "W3C". Podemos apreciar también que este elemento destaca visualmente porque el texto aparece subrayado con puntos.




### 2.8.3. Elemento ADDRESS

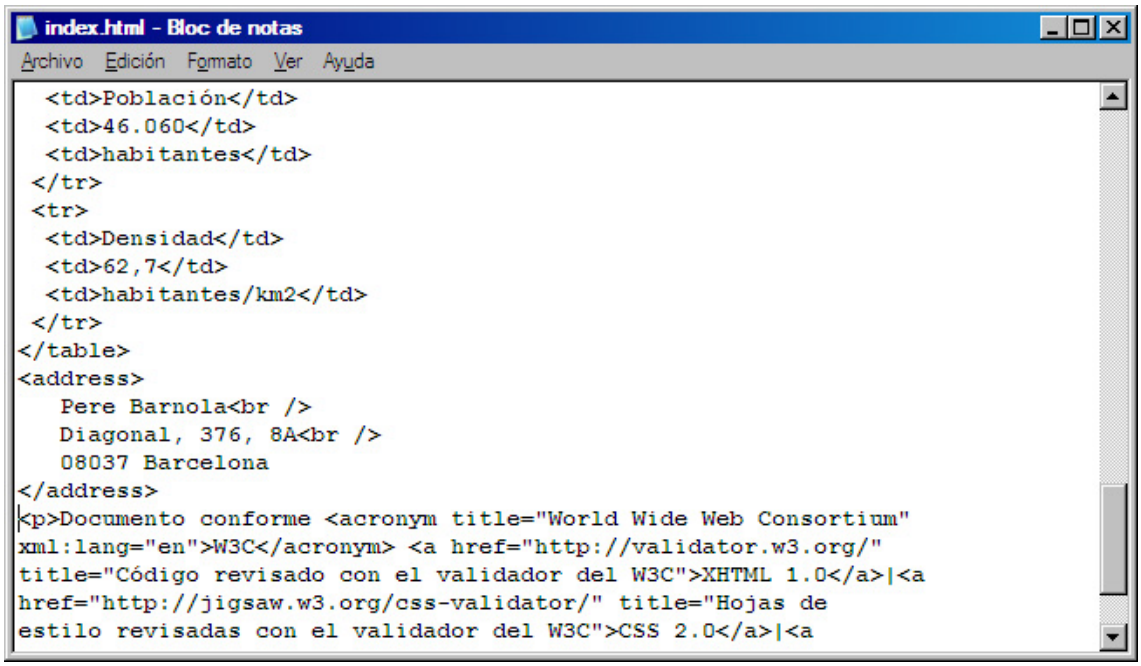
Este elemento nos sirve para marcar información de contacto, ya sea una dirección, firma o autor del documento.

La etiqueta <address> no se tendría que utilizar para marcar una dirección postal, a menos que ésta esté dentro de la parte de información de contacto.

**Visualización del elemento ADDRESS**

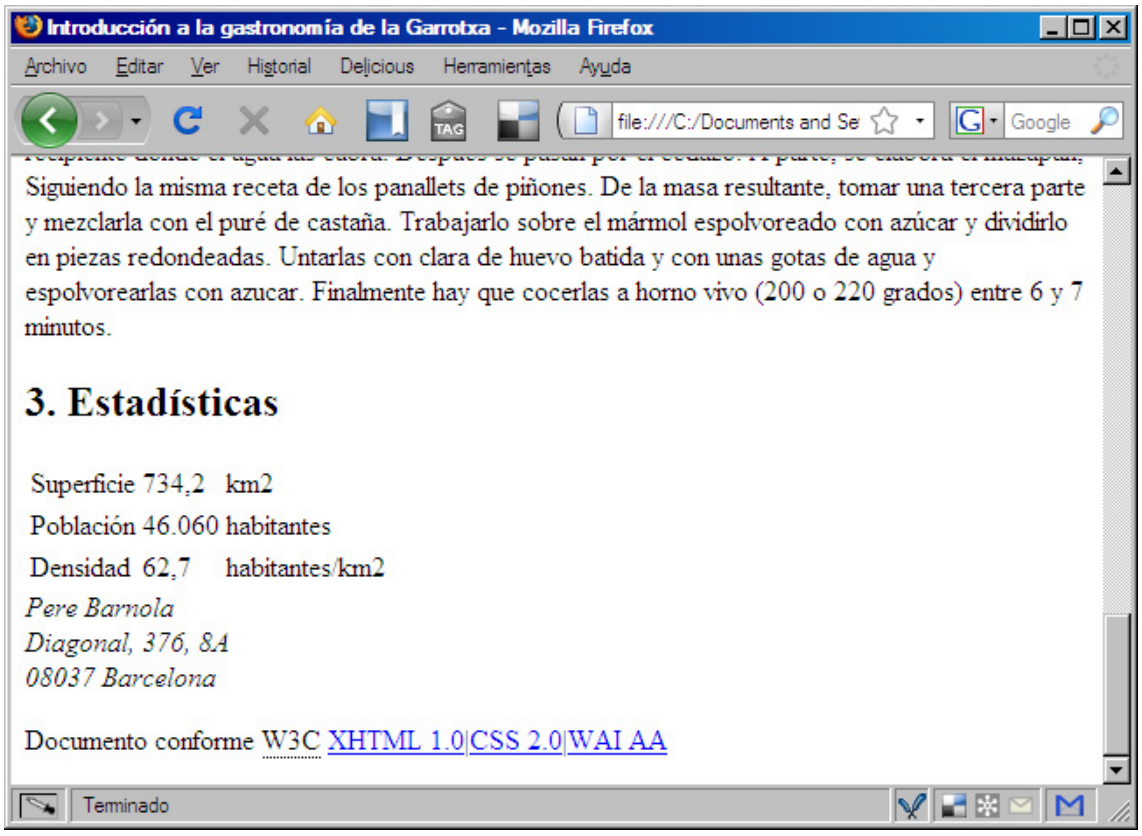
Normalmente, se visualiza en los navegadores en itálico. La mayoría de navegadores añadirán un salto de línea antes y después del elemento, pero si queremos saltos de línea dentro de los tags les tendremos que añadir nosotros.

Añadiremos este elemento en nuestro documento de ejemplo, indicando la dirección del autor del documento.




Disponéis del código en el archivo **ejemplo12.html**

En el navegador, podemos apreciar que el texto que figura dentro de las etiquetas del elemento <address> está tratado como texto itálico.



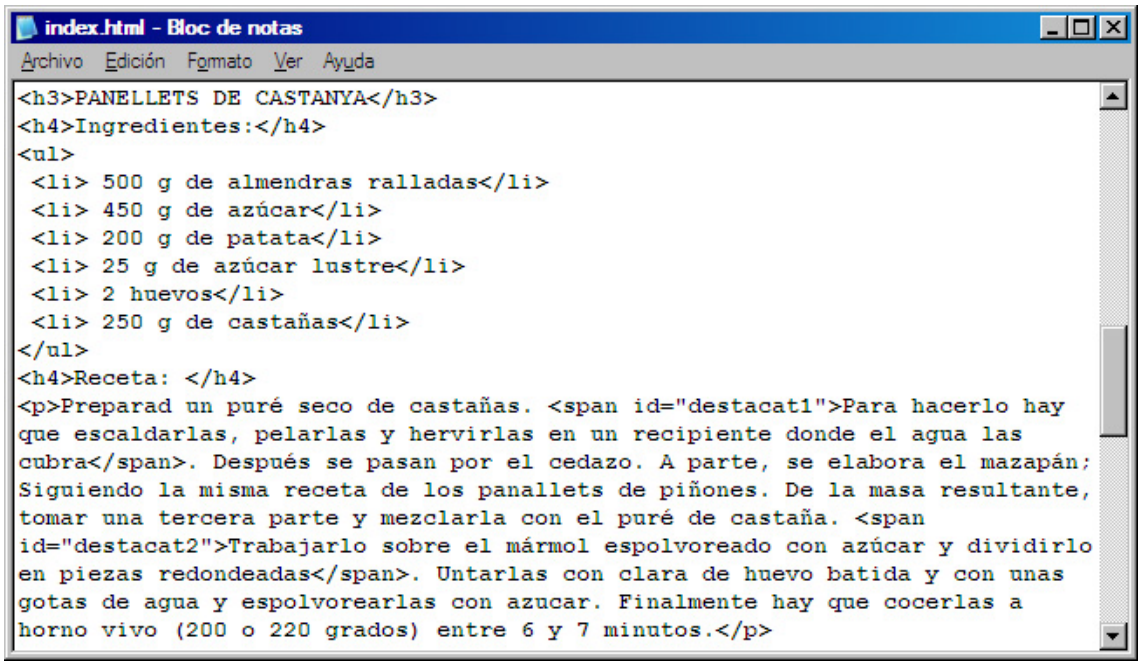

### 2.8.4. Elemento SPAN

El elemento span se utiliza para agrupar y estructurar partes específicas de texto de la misma manera que hacíamos con el div. La única diferencia es que el div es un elemento de tipo blog y el span es de tipo "inline". Eso quiere decir que el elemento span nos mantendrá su contenido en la misma línea a diferencia del div, que siempre nos forzará un salto de línea antes y después. Ya entraremos más adelante, cuando veamos la parte de estilos, en la utilidad de este tag.

Un ejemplo de uso del elemento sería el siguiente:

```
<p>Este es un texto normal en nuestro documento
<Span id="destacat">este otro texto tendrá una característica visual especial</span></p>
```

Pondremos unos cuantos elementos span en nuestro documento de ejemplo, aunque no tendrá ninguna repercusión visual de momento.




Disponéis del código en el archivo **ejemplo13.html**

**Lista completa**

Para una lista completa de todos los elementos para el marcaje de documentos en XHTML, podéis visitar el enlace siguiente: **http://www.w3.org/TR/REC-html40/index/elements.html**

## 2.9. Validaciones

Una vez tengamos nuestro documento XHTML con todo el contenido bien etiquetado y correctamente estructurado, convendría que nuestro código estuviera validado por los últimos estándares web.

**Tenemos tres maneras de validar nuestro código XHTML:**

**1) Señalando la dirección de nuestro documento a validar.**

**2) Señalando el nombre y la ruta del archivo dentro de nuestro ordenador.**

**3) Copiando el código y enganchándolo en el área de texto indicada en el web.**

Una vez hecho el análisis, el validador nos dará información sobre cualquier error detectado en nuestro código. Estas indicaciones que nos devuelva el validador siempre serán relativas a la versión del XHTML que hayamos especificado en el elemento DOCTYPE.

**Validador del W3C**

La principal herramienta para la validación de documentos XHTML es el validador del W3C, que podremos encontrar en: **http://validator.w3.org/**

Los errores más típicos son el cierre erróneo de etiquetas de elementos y los imbricados no permitidos de algunos elementos.

**Ventajas de la validación**

Las ventajas de seguir los estándares y tener la web completamente validada son muchos. Todavía hoy en día nos encontramos con que la mayoría de webs no siguen los estándares. Por lo tanto, tener validado nuestra web ya nos coloca en un paso por encima de muchas webs en este sentido. Además, si seguimos los últimos estándares y separamos el contenido de la manera en cómo éste se muestra, tenemos, por una parte, una estructuración de nuestra web mucho más óptima y, por otra parte, una mejor valoración de nuestra web por parte de buscadores como Google o Yahoo, lo cual supone que saldremos antes en las buscas, hecho que se traduce en más presencia en Internet.

## 3. Caso práctico: "nuestra primera web"

**El objetivo** de esta actividad será conocer y utilizar de forma correcta los elementos que hemos estado describiendo con anterioridad, y la importancia que aporta el significado semántico al contenido.

Se adjuntará un contenido sin marcar, al cual se tendrá que dar formato.

**Resultado final del caso práctico visto a lo largo del módulo**

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="es" lang="es">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
<meta name="author" content="Pere Barnola"/>
<meta name="keywords" content="garrotxa, gastronomía"/>
<title>Introducción a la gastronomía de la Garrotxa</title>
</head>
<body>
<div id="cabecera">
<h1>La gastronomía en la Garrotxa</h1>
</div>
<div id="menu">
<h1>Menú</h1>
<ol>
<li><a href="#introduccion" title="Descripción del
taller">Introducción</a></li>
<li><a href="#platos_tipicos" title="Conocimientos a adquirir">Platos
típicos</a></li>
<li><a href="#estadisticas" title="Guías que se
utilizan">Estadísticas</a></li>
</ol>
</div>
<div id="contenido">
<h2>1. Introducción<a name="introduccion"></a></h2>
<img src="foto_garrotxa.gif" alt="imagen del cráter de
```

```
Santa Margalida" width="376" height="226" />

<p>La Garrotxa una comarca de transicitre la montaña y la plana<a[plana|llanura]>, que se 'estentre el Pirineo y la Cordillera Transversal. A pesar de la complejidad de la comarca se distinguen dos sectores bien diferenciados y delimitados: el Alta Garrotxa y la Baja Garrotxa.

La primera, ferega y de relevo<a[relevo|relieve]> difl, con poca vegetaci llena de desfiladeros, simas y cuevas, dnom en la comarca -Garrotxa = tierra áspera y rotura-, y un paradper a los excursionistas; la Baja Garrotxa, de sierras suaves y depresiones volcques, mplanera, con prados abundantes y corrientes d'aigua</p>

<h2>2. Platos típicos<a name="platos_tipicos"></a></h2>

<h3>PANECILLOS DE CASTAÑA</h3>

<h4>Ingredientes:</h4>

<ul>

<li> 500 gr de almendras ralladas</li>

<li> 450 gr de azúcar</li>

<li> 200 gr de patatas</li>

<li> 25 gr de azúcar lustre</li>

<li> 2 huevos</li>

<li> 250 gr de castañas</li>

</ul>

<h4>Receta:</h4>

<p>Preparad un puré seco de castañas. Para hacerlo, hay que escaldarlas, pelarlas y hervirlas en un recipiente que las cubra con agua. Después, las pasáis por el cedazo. Aparte, elaborad el mazapán; siguiendo la misma receta de los panecillos de piñones. De la masa resultante, tomad una 1/3 parte y mezcladla con el puré de castaña. Trabajadlo sobre el mármol empolvado con azúcar y divididlo en piezas redondeadas. Untadlas con clara de huevo batida con unas gotas de agua y empolvadlas con azúcar. Finalmente, hay que cocerlas en el horno (200 o 220 grados) entre 6 o 7 minutos.</p>

<h2>3. Estadísticas<a name="estadisticas"></a></h2>

<table>

<tr>

<td>Superficie</td>

<td>734,2</td>

<td>km2</td>

</tr>

<tr>

<td>Población</td>

<td>46.060</td>

<td>habitantes</td>

</tr>

<tr>

<td>Densidad</td>

<td>62,7</td>

<td>habitantes/km2</td>

</tr>
```

```
</table>
<address>
Pere Barnola<br /> Diagonal, 376, 8A<br />
08037 Barcelona
</address>
<p>Documento conforme <acronym
title="World Wide Web Consortium" xml:lang="en">W3C</acronym>
<a href="http://validator.w3.org/"
title="Código revisado con el validador del W3C">XHTML 1.0</a>|
<a href="http://jigsaw.w3.org/css-validator/"
title="Hojas de estilo revisadas con el validador del W3C">CSS 2.0</a>|
<a href="http://www.tawdis.net/taw3/online"
title="Nivel de adecuación doble A, conforme a las directrices de la Web
Accessibility Initiative">WAI AA</a></p>
</div>
</body>
</html>
```

### Comentario sobre el caso práctico

Como vemos en esta posible solución que hemos planteado, tenemos, en primer lugar, la definición del XML y del DOCTYPE, necesarios para poder construir un XHTML válido. Después nos encontramos con el elemento html que contiene al mismo tiempo los dos elementos esenciales para la construcción de nuestro documento, el head y el body. Tal como hemos aprendido, el head contiene información relativa al documento como el title y los elementos meta que sirven para especificar al autor del documento o las palabras clave que lo definen.

En el body nos encontramos con el contenido de nuestro documento. Este contenido lo hemos estructurado en distintos elementos div, en nuestro caso con tres: uno para la cabecera, otro para el menú y, finalmente, otro para el contenido.

Vemos cómo el título de cada sección de nuestro contenido lo hemos etiquetado con encabezamientos. Eso, aparte de diferenciar cada sección de forma visual, hará que tengamos una mejor valoración por parte de los buscadores web.

Finalmente, hay que comentar que la utilización de los elementos está relacionada con su finalidad. Es decir, cuando tenemos un texto explicativo utilizamos el elemento p, cuando tenemos, como en nuestro caso, una receta donde tenemos que enumerar los ingredientes, utilizamos el elemento ul, o cuando tenemos información susceptible de ser ordenada en filas y columnas, como las estadísticas de densidad de la Garrotxa, utilizamos tablas.

**Finalmente, comprobaremos que nuestro código cumple todos los requisitos para ser una web perfectamente validada por los últimos estándares. Si el resultado de la validación es una cosa como: "This Page Is Valid XHTML 1.0 Strict", ya estaremos preparados para avanzar.**

# La presentación

Como ya hemos ido diciendo en los capítulos anteriores, una de las claves para construir un documento web de forma correcta es separar el contenido de como éste se presenta. En el contenido, sólo tendrá que haber las etiquetas que describen el contenido, pues no requiere ninguna etiqueta de representación o de estilos para transmitir completamente su mensaje. El hecho de que la presentación vaya por otra parte nos facilitará modificar la visualización del contenido en función del medio en que lo queramos mostrar. Por lo tanto, nos dará mucha flexibilidad al mismo tiempo de modificar el "look and feel" de una web. Por ejemplo, si el contenido se tiene que ver en un teléfono móvil, es lógico que modifiquemos el diseño para mejorar la legibilidad y usabilidad en este entorno, aunque el contenido sea el mismo.

La tecnología que nos permitirá dar estilo, con cierta facilidad, a nuestro documento web serán las **CSS**. En los próximos capítulos las veremos en profundidad.

**CSS Zen Garden**

En la web siguiente, podemos ver cómo un mismo contenido se puede visualizar de manera completamente diferente sólo con el hecho de modificar el estilo: **http://www.csszengarden.com**

## 1. Las hojas de estilo (CSS)

Las hojas de estilo en cascada (*cascading style sheets*) son un mecanismo simple que nos describe cómo se muestra un documento en la pantalla del ordenador, o cómo se imprime, o incluso cómo se pronuncia la información del documento en un dispositivo de lectura.

Con las CSS, tendremos un control total sobre el estilo y el formato de nuestros documentos. Cualquier cambio realizado en el estilo establecido para un elemento afectará a todas las páginas vinculadas a la CSS en las cuales aparezca este elemento.

## 2. Cómo se puede dar estilo a un documento XHTML

La CSS funciona con declaraciones sobre el estilo de uno o más elementos. Por lo tanto, las hojas de estilo están compuestas por una o más declaraciones referidas a elementos de un XHTML. Una declaración (regla) tiene dos partes: una propiedad y el valor de la propiedad.

**Ejemplo**

p {font-size: 10em;} en este caso p es el selector y {font-size: 10em;} la declaración.

El selector es el que nos hace de enlace entre el documento y el estilo, especificando qué elementos se verán afectados por la declaración. Y la declaración es la parte que nos dice cómo se ven afectados los elementos. En el ejemplo, estaríamos indicando que todos elementos p se verán afectados por la declaración que establece que la propiedad *font-size* tendrá el valor 10em para todos los elementos *p* del documento o documentos que estén vinculados a la hoja de estilo.

A continuación, veremos las formas en que se puede hacer esta vinculación y todas las propiedades que podemos tener en una declaración.

### 2.1. La vinculación CSS-XHTML

Hay diferentes maneras de dar estilo a un documento mediante CSS, aunque hay algunas que rompen con la premisa de separar el contenido de la presentación.

La mejor manera de hacerlo, y la más recomendable, es utilizando una hoja de estilo externo que se vinculará al documento mediante el elemento <link>, el cual tiene que estar situado dentro de la sección <head>. En este caso, la hoja de estilo estará en un archivo con extensión ".css" totalmente separado del "xhtml". Lo cual nos permitirá que, al modificar el archivo .css, se modifiquen los estilos de todos los documentos donde está vinculado.

Lo primero será crear un archivo donde colocaremos nuestros estilos. Este fichero lo iremos construyendo a medida que expliquemos las propiedades principales y más utilizadas de css (el resto de propiedades las dejaremos como referencia). Lo guardaremos" como "estilos.css" en el mismo lugar donde tenemos el "index.html" que creamos en el módulo 1. Una vez creado, abriremos nuestro documento XHTML y añadiremos el siguiente código a la cabecera.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="es" lang="es">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
<meta name="author" content="Pere Barnola"/>
<meta name="keywords" content="garrotxa, gastronomía"/>
<title>Introducción a la gastronomía de la Garrotxa</title>
<link href="estilos.css" rel="stylesheet" type="text/css" />
</head>
<body>
```

De esta manera ya tendremos realizada la vinculación entre el archivo "estilos.css", que es donde tendremos todos los estilos de nuestro documento, y el archivo "index.html", que es donde teníamos el contenido.

Como hemos dicho, hay otras formas de dar estilos a un documento, aunque rompen ligera o completamente la premisa de separar el contenido de los estilos.

Se puede utilizar el elemento <style> en el interior del documento al cual se quiere dar estilo, y que generalmente se situará dentro de la sección <head>. Se opta por hacerlo de esta manera cuando los estilos se utilizan sólo en el documento en concreto.

Aquí tenemos un ejemplo:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="es" lang="es">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
<meta name="author" content="Pere Barnola"/>
<meta name="keywords" content="garrotxa, gastronomía"/>
<title>Introducción a la gastronomía de la Garrotxa</title>
<Style type="text/css">
body {
padding-left: 11em;
font-family: Georgia, "Times New Roman", Serif;
color: red;
```

```
background-color: #d8da3d;
}
h1 {
font-family: Helvetica, Geneva, Arial, Sans-serif;
}
</Style>
</head>
<body>
```

También se podrían utilizar los estilos directamente sobre aquellos elementos que lo permiten por medio del atributo <style> dins del <body>. Sin embargo, este tipo de definición sí que rompe totalmente con las ventajas que nos ofrece el hecho de no mezclar el contenido con la presentación.

## 2.2. Selectores y declaraciones

Como hemos avanzado en la primera parte, los CSS constan de reglas o declaraciones que relacionamos con los elementos de nuestro documento XHTML mediante los selectores.

**Ejemplo**

En este ejemplo, p{color: red;}, el selector p, le dice al navegador la parte del documento que se verá afectado por la regla.

Los selectores pueden aparecer individualmente o agrupados, separándolos con comas:

Por tanto, esto

```
p, h1, h2 {
color: red;
}
```

sería lo mismo que:

```
p {color: red;}
h1 {color: red;}
h2 {color: red;}
```

La propiedad que en el caso del ejemplo anterior sería "color" especifica qué aspecto cambiaremos. En el ejemplo, cambiaríamos el color. Las propiedades que se quieren modificar para un mismo selector se pueden agrupar separándolas mediante un punto y coma.

```
p {text-align:center; color:red;}
```

Normalmente, describiremos una propiedad por línea de la manera siguiente:

```
h1 {
padding-left: 11em;
font-family: Georgia, "Times New Roman", Times, Serif;
color: red;
background-color: #d8da3d;
}
```

El valor, representado a la derecha de los dos puntos (: ), establece el valor de la propiedad. Es importante recordar que si el valor está formado por más de una palabra, se tiene que poner entre comillas.

```
p {font-family: "sans serif";}
```

**Hay diferentes tipos de selectores.**

**Los selectores de elemento o etiqueta, que hacen referencia directa a elementos o etiquetas que nos encontramos en el documento XHTML y son los que hemos visto en los ejemplos anteriores.**

**Los selectores de id, que hacen referencia al identificador de una sección o elemento dentro de nuestro XHTML.**

**#id {la declaración }**

Nos permite aplicar estilo a una parte única de nuestro documento identificada por un atributo id, cuyo valor será el nombre del selector.

Hemos de tener en cuenta que el ID será un identificador único para un elemento dentro del documento, por lo que los estilos sólo podrán afectar a una única parte del documento, al contrario de los selectores de clase, que veremos más adelante.

Por ejemplo, si en nuestro documento XHTML tenemos un elemento

```
<div id="capsa"></div>
```

Para referirnos a él desde la hoja de estilos, escribiríamos lo siguiente dentro del documento CSS:

```
#capsa {font-family: "sans serif"; }
```

Y estos estilos sólo afectarían al contenido dentro del elemento que tiene el id=caja.

Los **selectores de clase** son los que se relacionan con los elementos o partes de nuestro XHTML, que tienen como valor del atributo clase el nombre del selector.

```
.clase1 {la declaración }
```

A diferencia de los selectores de id, los selectores de clase se pueden relacionar con diferentes elementos o partes de nuestro documento XHTML que estén identificados con la "clase" en la cual hemos definido los estilos.

Por ejemplo, si tenemos el siguiente código en el XHTML:

```
<p class="vermell">
Hola este texto es rojo. Pero la siguiente palabra es
<span class="amarilla">amarilla</span>
</p>
```

Para definir las clases "rojo" y "amarilla", que después podremos reutilizar en otras partes del documento, haríamos lo siguiente en nuestro documento CSS:

```
. rojo {color: #FF0000;}
. amarilla {color: #FFFF00; }
```

También tenemos los selectores contextuales, que se relacionan con elementos o partes de nuestro XHTML para el tipo de elemento y el contexto que se especifica en el selector.

Serán del tipo:

```
Contexto elemento {la declaración};
```

Cuando decimos contexto de un elemento nos referimos a los antecedentes del elemento, es decir, a los padres.

Por ejemplo, si tenemos el código siguiente:

```
<div id="cap">
<ul>
<li>elemento 1 lista</li>
<li>elemento 2 lista</li>
```

```
</ul>
</div>
```

El contexto del elemento 1 de la lista sería el siguiente "#cap ul li", que resumiendo querría decir "los li, dentro de ul, dentro de elemento con id cap". Por lo tanto, si en nuestra hoja de estilos tuviéramos lo siguiente:

```
#cap ul li {font-family: "sans serif";}
```

Estos estilos sólo se aplicarían a los elementos que cumplieran este contexto.

Hay que tener en cuenta que, para un elemento, siempre mandarán los selectores contextuales sobre tipos de selectores más genéricos.

También están los **selectores de atributo**, que se relacionan con los elementos que contienen un atributo con el valor que se especifica en el selector cuando se aplican los estilos. De este tipo selector no hablaremos mucho, ya que no lo soportan la mayoría de navegadores. Sólo pondremos la forma por si os lo encontráis alguna vez.

Son del estilo:

```
Selector[atributo="valor"] {la declaración };
```

Quiere decir que los estilos sólo se aplicarán al elemento dentro del documento que contenga el atributo que se especifica igual al valor del atributo que especificamos en el selector.

Como última cosa antes de empezar a ver las diferentes propiedades que podemos modificar para aplicar estilos en los documentos, cabe decir que tenemos que considerar que los estilos se heredan. ¿Qué quiere decir eso? Que algunos elementos heredan los estilos de su padre, a menos que se especifique lo contrario.

**Ejemplo**

Por ejemplo, si defino un tipo de letra para el elemento <body>, por defecto, si no especifico lo contrario, todos los elementos de mi documento dentro del *body*, tomarán ese tipo de letra.

## 2.3. Propiedades de texto: tipografía

Comencemos a ver las propiedades que podemos modificar en un selector que se aplique a texto: tipografía, color, fuente, decoración, alineación, espacios, etc.

De los elementos de texto, podemos modificar dos tipos de propiedades: las que se refieren a la **tipografía**, y las que se refieren al **espaciado**.

Las propiedades que podemos variar para la tipografía son las siguientes:

### 2.3.1. Color

Esta propiedad nos permite especificar el color del texto.

Valor: el valor puede ser el color en rgb (rgb (100%, 50%, 0%), rgb (255, 128, 0)), el color en hexadecimal (#ffa500, #008000, #800080), o uno de los 17 nombres de colores predefinidos (aqua, black, blue, fuchsia, gray, green, lime, maroon, navy, olive, orange, purple, red, silver, teal, white, yellow).

La propiedad color se hereda, o lo que es lo mismo, si no se especifica, toma el valor del padre.

Se puede aplicar a todos los elementos y lo soportan todos los navegadores.

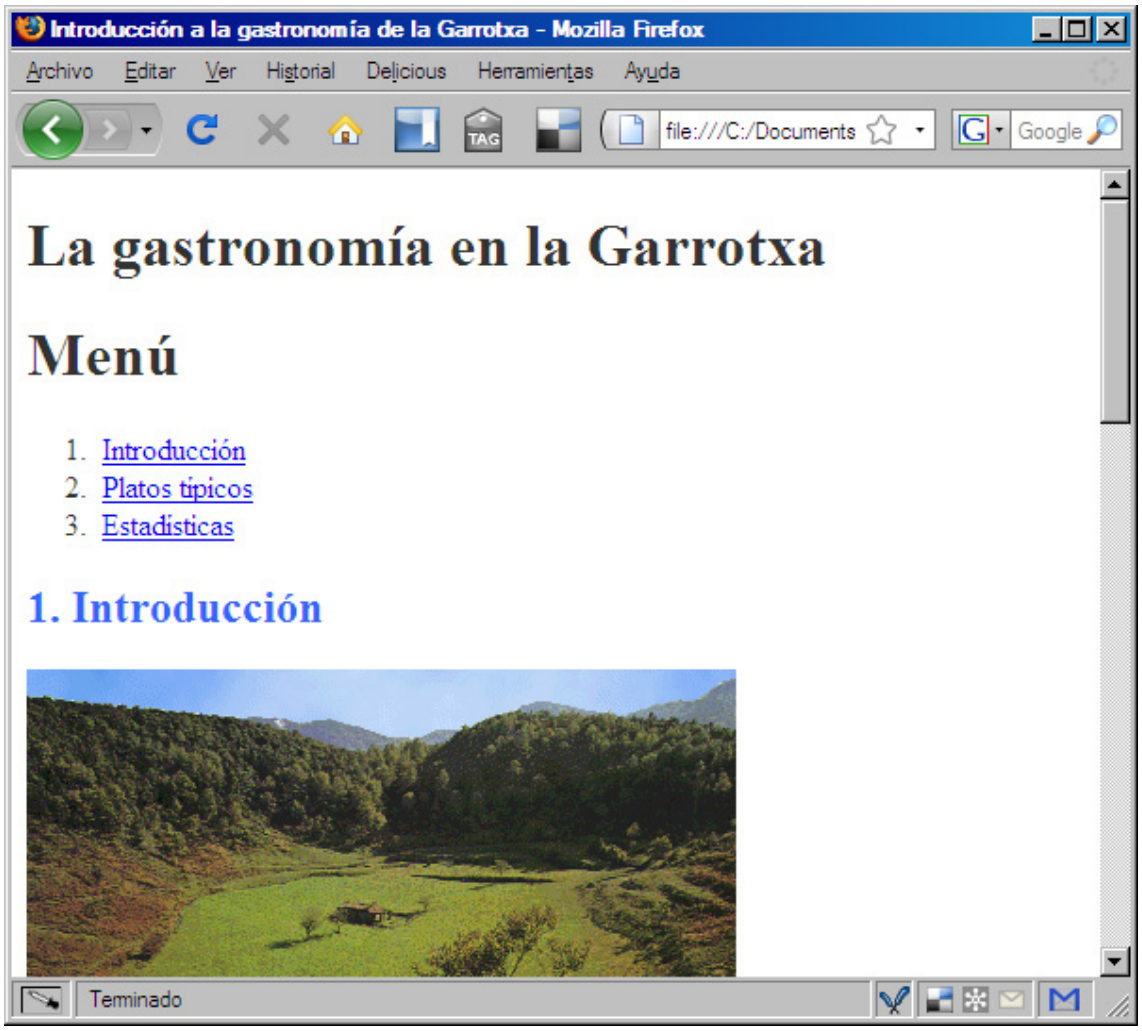
Seguidamente, algunos ejemplos:

```
h2{ color: lime;}
p{ color: #ff0000; }
a{ color: rgb(255,255,0); }
```

Siguiendo con nuestro ejemplo, abrimos el archivo "estilos.css" y añadimos este código:

```
body {
color: #333333;
}

h2 {
color:#3366FF;
}
```

Vemos el efecto sobre la página que estamos construyendo. Si abrimos ahora el archivo "index.html" con nuestro navegador, veremos de qué manera afectan a estos estilos que acabamos de incorporar en el fichero estilos.css:




Los textos que están con encabezamientos de nivel 2, como "1. Introducción", se ven afectados por el estilo:

```
h2 {
color:#3366FF;
}
```

Veamos, pues, cómo el texto sale de color azul.

### 2.3.2. Font-family

Es una lista de nombres de familias de fuentes en orden de prioridad. En los ordenadores, no siempre tenemos todas las fuentes instaladas. Existen las llamadas "fuentes de sistema", que en principio las tiene todo el mundo. En esta propiedad se hará caso a la primera fuente de la lista que exista en el ordenador, donde se está visualizando el documento. Por ejemplo, si se especifica font-family: Arial, Verdana; y resulta que, en mi ordenador, no tengo Arial pero sí Verdana, lo vería en Verdana.

El valor de la propiedad, como hemos dicho, es una lista de familias de fuentes separadas por coma ",".

También es una propiedad que se hereda, se puede aplicar a todos los elementos y la soportan todos los navegadores principales.

Un ejemplo sería el siguiente:

```
p{
font-family: Arial, Helvetica, Sans-serif;
}
```

Seguimos con nuestro ejemplo "estilos.css". Añadiremos ahora este código:

```
body {
font-family: Georgia,"Times New Roman", Serif;
color: #333333;
}

h1 {
font-family: Helvetica, Geneva, Arial, Sans-serif;
}

h2 {
color:#3366FF;
}
```

Visualizamos el archivo "index.html" en un navegador y vemos los resultados:


### 2.3.3. Font-size

Esta propiedad nos permite establecer la medida de la fuente a utilizar. Su valor se puede especificar con una "medida", un "porcentaje" o alguna de las medidas absolutas o relativas predefinidas.

Lo que explicaremos ahora sobre las diferentes maneras de asignar las medidas se puede aplicar también a otras propiedades que requieran como valor una medida.

Una "medida" será un número decimal seguido de una unidad. Las **unidades absolutas** son las siguientes:

mm: (milímetro)

cm: (centímetro, 1 cm = 10 mm) in: (pulgada (inch), 1in = 2,54 cm) pt: (punto, 1 pt = 1/72 in)

pc: (pica, 1 pc = 12 pt)

Las **unidades relativas** son aquellas que, a pesar de la redundancia, son relativas a una medida del documento. En general, a la medida de la fuente. Estas unidades son:

em: (1 em = medida de la fuente actual)

ex: (1 ex = altura de la letra 'x' en la fuente actual)

Si la medida que especificamos es de 2 em, y la medida de la fuente actual es 10 px, la medida que hemos especificado será de 20 px. El hecho de especificar las medidas en unidades relativas es especialmente útil en el caso de hacer diseños que se adapten a la medida de la pantalla del usuario.

**La unidad más utilizada, sin embargo, es el px (píxel).**

También podemos especificar las medidas de forma relativa en porcentajes. En el caso de las medidas predefinidas, también las tenemos de dos tipos:

1) Las **absolutas**,

**Medidas absolutas**

Las medidas absolutas con su correspondencia con los encabezamientos y el factor de escala:

| Medida | Factor de escala | Encabezamiento |
|---|---|---|
| xx-large | 2 | h1 |
| x-large | 1.5 | h2 |
| Large | 1.2 | h3 |
| Medium | 1 | h4 |
| Small | 0.89 | h5 |
| x-small | 0.75 | |
| xx-small | 0.6 | h6 |

2) Las **relativas** en las cuales podemos especificar "larger", que aumentará la fuente actual un paso y "smaller" que la decrementará.

El valor por defecto de la propiedad *font-size* es el valor predefinido "medium". Es decir, si no especificamos nada, el valor será "medium". Es una propiedad que hereda, lo que quiere decir que toma la medida de su padre si no se especifica lo contrario. Se puede aplicar a todos los elementos y la soportan todos los navegadores principales.

Aquí podemos ver algunos ejemplos:

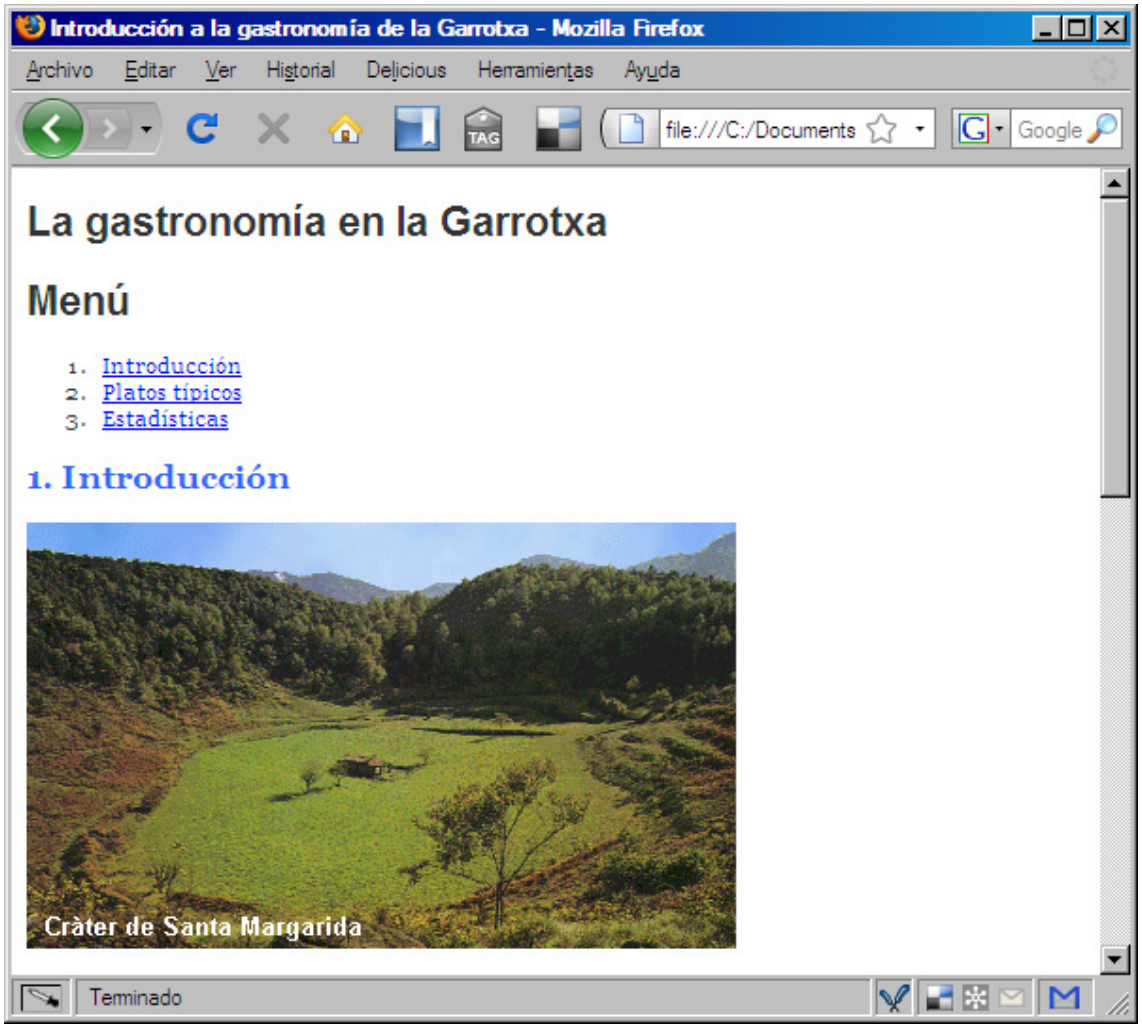
```
p{
font-size:10px;
}

.classe1{
font-size:x-large;
}

#peu{
font-size:smaller;
}
```

En nuestro ejemplo, añadimos el siguiente código en el archivo "estilos.css":

```
body {
font-family: Georgia,"Times New Roman", Serif;
color: #333333;
font-size: 11px;
}
```

```
h1 {
font-family: Helvetica, Geneva, Arial, Sans-serif;
}

h2 {
color:#3366FF;
}
```

Ahora veremos el impacto de éste cuando visualizamos nuestro archivo en el navegador:




Como vemos, la tipografía de las listas que hay dentro del "body" ha tomado la nueva medida, igual que lo ha hecho el resto de texto genérico.

### 2.3.4. Font-style

Especifica el estilo oblicuo o itálico dentro de la familia de fuente actual.

Los valores que puede tomar esta propiedad son: "normal", "italic", u "oblique".

El valor por defecto es "normal"; es una propiedad que hereda del padre el valor si no se especifica, es aplicable a todos los elementos y la soportan todos los navegadores.

```
.clase1{font-style:oblique;}


#cos{font-style:italic;}
```

### 2.3.5. Font-variant

Esta propiedad determina si la fuente se muestra en mayúsculas de tipo normal o pequeñas "small-caps". Éstas se muestran de forma que todas las letras de una determinada palabra están en mayúsculas con caracteres ligeramente mayores que las minúsculas.

Los valores que se pueden especificar son "normal" o "small-caps". El valor por defecto es "normal"; es una propiedad que hereda del padre el valor si no se especifica, es aplicable a todos los elementos y la soportan todos los navegadores.

### 2.3.6. Font-weight

Especifica el grueso de la fuente actual. Podemos definir que sea en negrita, normal o delgada.

Los valores que se pueden especificar son: "normal", "bold", "lighter", o uno de 9 valores numéricos (100, 200..., 900). El valor por defecto es "normal". Es una propiedad que hereda del padre el valor si no se especifica, es aplicable a todos los elementos y la soportan todos los navegadores.

```
p{ font-weight:bold; }
a{ font-weight:400: }
```

### 2.3.7. Text-decoration

Especifica si el texto aparece subrayado, con una línea superior, rayado, o con parpadeo.

Los valores que se pueden especificar son: "none" (ninguno), "underline" (subrayado), "overline"(línea superior), line-through" (rallado) o "blink" (parpadeo).

El valor por defecto es "none"; es una propiedad que no hereda del padre el valor si no se especifica, es aplicable a todos los elementos y la soportan todos los navegadores.

A continuación, podemos ver el ejemplo más típico con los enlaces. Los enlaces están, por defecto, subrayados y de color azul. Una cosa que no hemos dicho es que los enlaces pueden tener diferentes estados. El estado normal, el estado cuando tenemos el ratón encima (mouseover), o el estado de visitado, que quiere decir que ya le hemos hecho clic.

En el ejemplo siguiente, hacemos referencia al estado normal "a", y al estado de mouseover "a:hover". Por lo tanto, el enlace se verá sin subrayado en el estado normal, y cuando nos pongamos encima (mouseover) aparecerá la línea superior tal como estamos especificando.

```
a{ text-decoration:none; }


a:hover{ text-decoration:overline; }
```

### 2.3.8. Text-transform

Permite al texto transformarse en alguna de las cuatro propiedades: "capitalize" (la primera letra con mayúscula), "uppercase" (todas las letras con mayúsculas), "lowercase" (todas las letras con minúsculas), "none" (ningún efecto sobre el texto).

Los valores que se pueden especificar son: "capitalize", "uppercase", "lowercase", "none". El valor por defecto es "none"; es una propiedad que hereda del padre el valor si no se especifica lo contrario, es aplicable a todos los elementos y la soportan todos los navegadores.

```
p{ text-transform:lowercase; }
```

## 2.4. Propiedades de texto: espaciado

Las propiedades que podemos variar con respecto al espaciado de texto son las siguientes:

### 2.4.1. White-space

Especifica cómo se comportarán las tabulaciones, los saltos de línea y el espacio extra en blanco en el contenido del elemento especificado.

Los valores que se pueden especificar son: "normal","pre", "pre-wrap", "pre-line". El valor por defecto es "normal"; es una propiedad que hereda del padre el valor si no se especifica, es aplicable a elementos de blog y la soportan todos los navegadores, excepto las propiedades "pre-wrap" y "pre-line", que fue añadido en el CSS 2.1 y sólo la soportan ciertos navegadores.

```
p{ white-space:nowrap; }
```

### 2.4.2. Text-align

Especifica la alineación horizontal de las líneas de texto. Podremos alinear un texto a la izquierda o a la derecha, en el centro o justificado.

Los valores que se pueden especificar son: "left", "right", "center", "justify". El valor por defecto es "none", es una propiedad que hereda del padre el valor si no se especifica, es aplicable a todos los elementos de blog y la soportan todos los navegadores.

```
p{ text-align:justify; }
```

### 2.4.3. Text-indent

Especifica la imbricación de la primera línea de texto de un párrafo.

El valor es una medida. El valor por defecto es "0", es una propiedad que hereda del padre el valor si no se especifica, es aplicable a todos los elementos de blog, y la soportan todos los navegadores.

```
p{ text-indent:10px; }
```

### 2.4.4. Line-height

Especifica la distancia mínima entre líneas base de texto.

Los valores que se pueden especificar son un factor, una medida o bien un porcentaje. El valor por defecto depende del navegador, es una propiedad que hereda del padre, el valor si no se especifica es aplicable a todos los elementos, y la soportan todos los navegadores.

El factor es un número decimal (por ejemplo, 1.2) que multiplicará la medida de fuente actual para calcular la nueva medida.

```
p{ line-height:1.2; }
```

### 2.4.5. Word-spacing

Especifica la cantidad adicional de espacio entre palabras.

Los valores que se pueden especificar son "normal" o una medida. El valor por defecto es "normal"; es una propiedad que hereda del padre, el valor si no se especifica lo contrario es aplicable a todos los elementos y la soportan todos los navegadores.

```
p{ word-spacing:10; }
```

### 2.4.6. Letter-spacing

Especifica la cantidad adicional de espacio entre letras.

Los valores que se pueden especificar son "normal" o una medida. El valor por defecto es "normal", es una propiedad que hereda del padre, el valor si no se especifica, es aplicable a todos los elementos y la soportan todos los navegadores.

```
p{ letter-spacing:10; }
```

### 2.4.7. Vertical-align

Sirve para alterar la ubicación vertical de un elemento en línea, en relación a su elemento padre o a la línea del elemento.

El valor puede ser un porcentaje relativo a la altura del elemento, que elevará la línea de base del elemento en la cantidad especificada por encima de la línea de base del padre. No se permiten valores negativos.

El valor también puede ser una palabra clave. Las siguientes palabras clave afectan a la ubicación en relación con el elemento padre:

- **Baseline**: alinea líneas base del elemento y el padre;
- **Middle**: alinea el punto medio vertical del elemento con la línea de base más la mitad de la altura de la letra "x" del padre.
- **Sub**: subíndice.
- **Super**: superíndice.
- **Text-top**: alinea las partes inferiores del elemento y la fuente del elemento padre.
- **Text-bottom**: alinea las partes inferiores del elemento y la fuente del elemento padre.

El valor por defecto es "baseline"; es una propiedad que no hereda del padre, el valor si no se especifica es aplicable a todos los elementos de línea, y la soportan todos los navegadores.

```
img{ vertical-align:middle; }
```

### 2.4.8. Direction

Especifica la dirección de escritura. Ésta puede ser de izquierda a derecha ("ltr") o de derecha a izquierda ("rtl"). Se utiliza, básicamente, para los lenguajes que se escriben de derecha a izquierda como el árabe.

El valor por defecto es "ltr", es una propiedad que hereda del padre el valor si no se especifica, es aplicable a todos los elementos y la soportan todos los navegadores.

```
p{ direction:rtl; }
```

## 2.5. Propiedades de disposición: estructura de caja

En cuanto en las propiedades de disposición ("layout", estructura de cajas, altura, anchura, margen, "padding", fondo, etc.), las podemos dividir en tres tipos:

**1)** Las de "estructura de caja",

**2)** las de posicionamiento,

**3)** y las de fondo.

Las propiedades de "estructura de caja" son las siguientes:

### 2.5.1. Margin

Especifica las cuatro propiedades individuales del margen de una vez.

```
div {margin: 1em 2em 3em 4em }
```

Donde cada uno de los valores corresponde a los márgenes superior, derecho, inferior e izquierdo respectivamente en este orden. Si sólo tenemos un valor, se aplica a todos los lados. Si tenemos dos o tres valores, los valores restantes se toman del lado opuesto.

```
margin: 1em;
```

(todos los márgenes = 1em)

```
margin: 1em 2em;
```

(superior e inferior = 1em, derecho e izquierdo = 2em)

```
margin: 1em 2 m 3m;
```

(superior = 1em, derecho e izquierdo = 2em, inferior = 3em)

```
margin: 1em 2em 3em 4em;
```

(superior = 1em, derecho = 2em, inferior = 3em, izquierdo = 4em)

Los posibles valores son hasta cuatro valores que pueden ser una medida, un porcentaje o "auto". El valor por defecto es "0", es una propiedad que no hereda del padre el valor si no se especifica, es aplicable a todos los elementos y la soportan todos los navegadores.

### 2.5.2. Margin-top, margin-right, margin-bottom, margin-left

Son cuatro propiedades y especifican el espacio entre el elemento seleccionado y los elementos que están al lado. Es decir, los márgenes, igual que hacía la propiedad "margin", pero especificando cada uno por separado.

Los valores que se pueden especificar son: un número, un porcentaje o bien "auto". El valor por defecto es "0", es una propiedad que no hereda del padre el valor si no se especifica, es aplicable a todos los elementos y la soportan todos los navegadores.

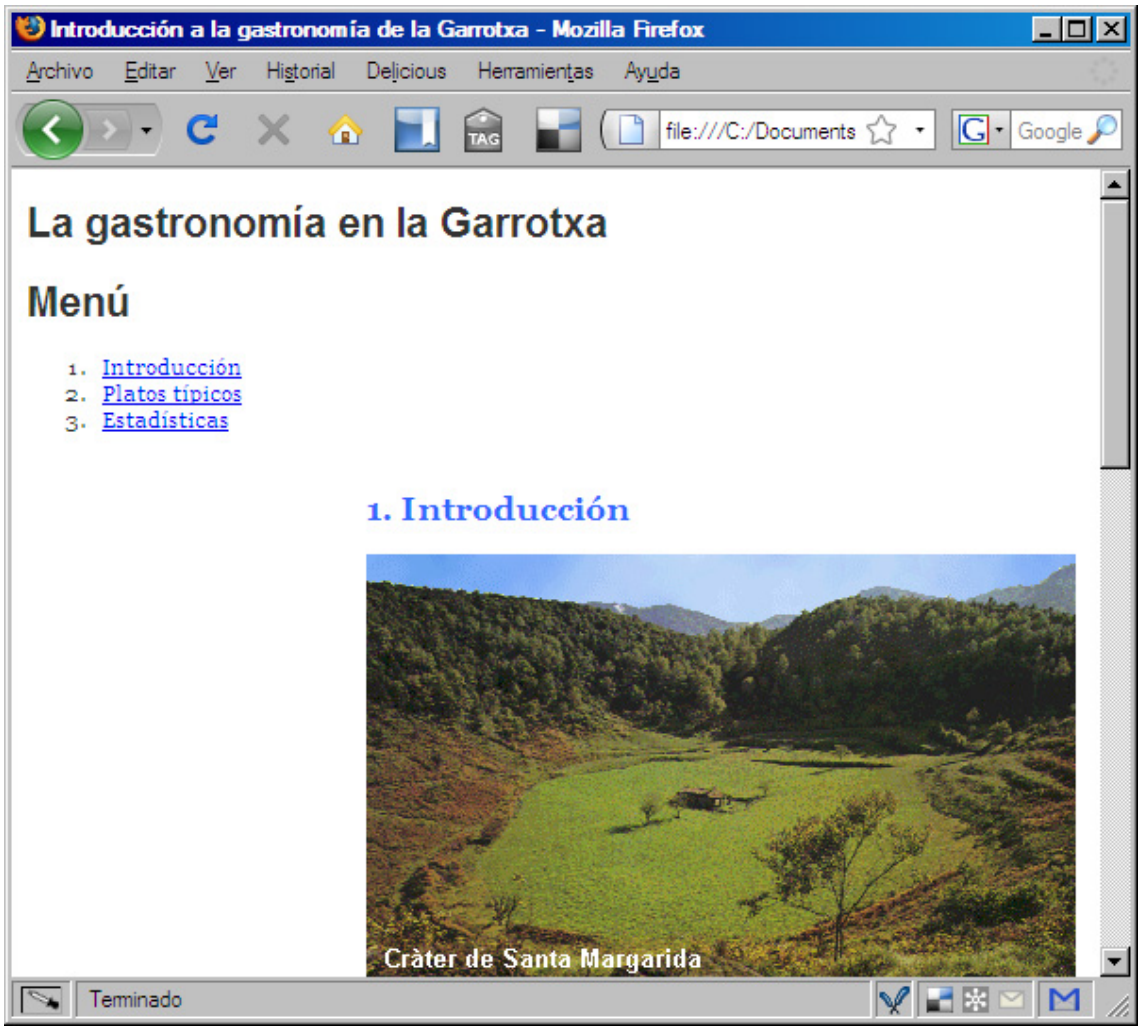
```
p{
margin-bottom:10px;
margin-left:auto;
margin-right:auto;
margin-top:20px;
}
```

Seguiremos ahora con nuestro archivo de ejemplo. Abriremos el archivo "estilos.css" y añadiremos el código siguiente:

```
body {
font-family: Georgia,"Times New Roman", Serif;
color: #333333;
font-size: 11px;
}

h1 {
font-family: Helvetica, Geneva, Arial, Sans-serif;
}
```

```
h2 {
color:#3366FF;
}

#contenido {
margin: 30px 10px 0px 32%;
}

#contenido address {
margin-top:10px;
}
```

Vemos la visualización en el navegador:

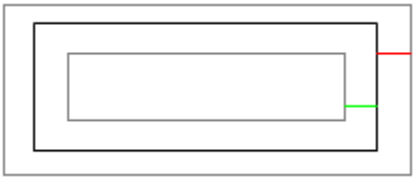



Podemos observar que todo el contenido dentro de la capa "content" se ha desplazado a la derecha la cantidad que le hemos especificado.

### 2.5.3. Padding

Para que nos entendamos, el "padding" es el "relleno". Si tenemos una caja que tiene contenido, el "margin" (margen), como hemos visto, sería el espacio que esta caja mantendrá en la parte de fuera enfrente de contenidos vecinos. Es decir, si especificamos 10px, nuestra caja estará separada del contenido 10px (dejamos 10px de margen). En cambio, el "padding" (relleno) es el espacio que tendrá que respetar el contenido de dentro de la caja con los límites de la caja.

En el siguiente esquema, lo veremos más claro. La línea verde sería el "padding" y la línea roja el "margin".



Después de esta nota, seguimos con el "padding". El "padding" nos especifica las cuatro propiedades de espacio entre el borde y el contenido del texto de una vez. Se aplica siguiendo el mismo mecanismo que la propiedad "margin".

Los posibles valores son hasta cuatro valores, que pueden ser una medida o un porcentaje. El valor por defecto es "0". Es una propiedad que no hereda del padre el valor si no se especifica, es aplicable a elementos y la soportan todos los navegadores.

### 2.5.4. Padding-top, padding-right, padding-bottom, padding-left

Especifica la cantidad de espacio que hay que insertar entre el contenido de un elemento y su margen o borde.

Los valores que se pueden especificar son: un número o un porcentaje. El valor por defecto es "0". Es una propiedad que no hereda del padre el valor si no se especifica, es aplicable a elementos y la soportan todos los navegadores.

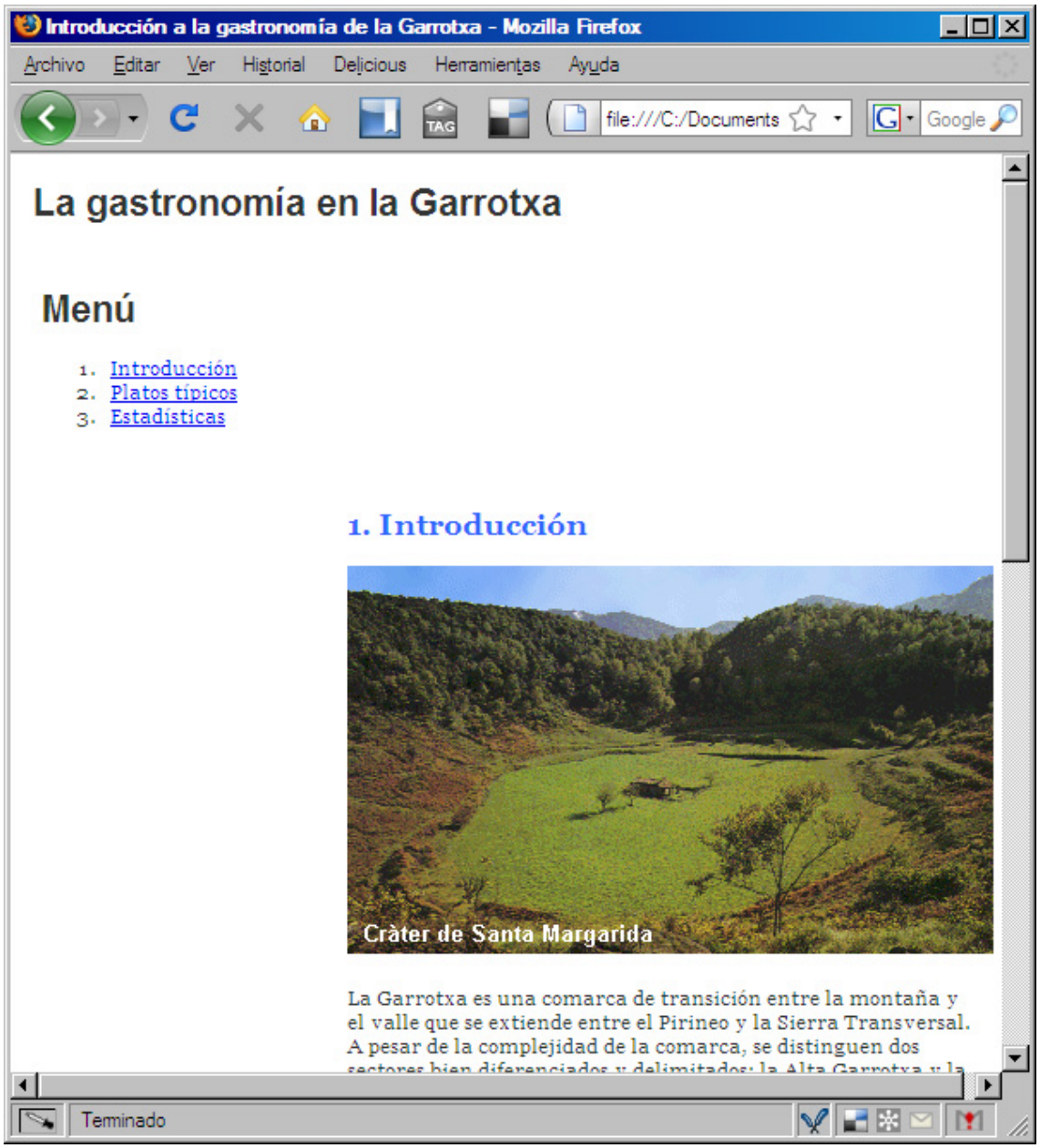
```
p{
padding-bottom:10px;
padding-left:auto;
padding-right:auto;
padding-top:20px;
}
```

Añadiremos algunos "paddings"en nuestro archivo "estilos.css":

```
body {
padding-left: 5px;
font-family: Georgia,"Times New Roman", Serif;
color: #333333;
font-size: 11px;
}

h1 {
font-family: Helvetica, Geneva, Arial, Sans-serif;
}

h2 {
```

```
color:#3366FF;
}

#cabecera {
padding-left:5px;
}


#menu {
padding-top:5px; padding-bottom:5px; padding-left: 5px;
}

#contenido {
margin: 30px 10px 0px 32%;
padding-left:5px;

}

#contenido address {
margin-top:10px;
}


#contenido img {
padding-right:10px;
padding-bottom:5px;
}
```

Y la visualización del archivo "index.html" en el navegador será así:




### 2.5.5. Border, border-top, border-right, border-bottom, border-left

Cada una especifica la anchura, el color y el estilo de cada uno de los bordes, a excepción del "border" que lo aplica a todos los lados en general.

Los valores que se pueden especificar están separados por un espacio: anchura (en píxeles o en porcentaje), el color (en RGB) y estilo. Son propiedades que no heredan el valor si no se especifica, son aplicables elementos y las soportan todos los navegadores.

Los valores que podemos especificar en el estilo del borde son:

- **None**: ningún estilo.
- **Dotted**: línea de puntos.
- **Dashed**: línea de barras.
- **Solid**: línea sólida.
- **Double**: doble grueso de línea.

- **Groove**, **ridge**, **inset** y **outset**: diferentes estilos de bordes en 3D.

A continuación, podemos ver algunos ejemplos:

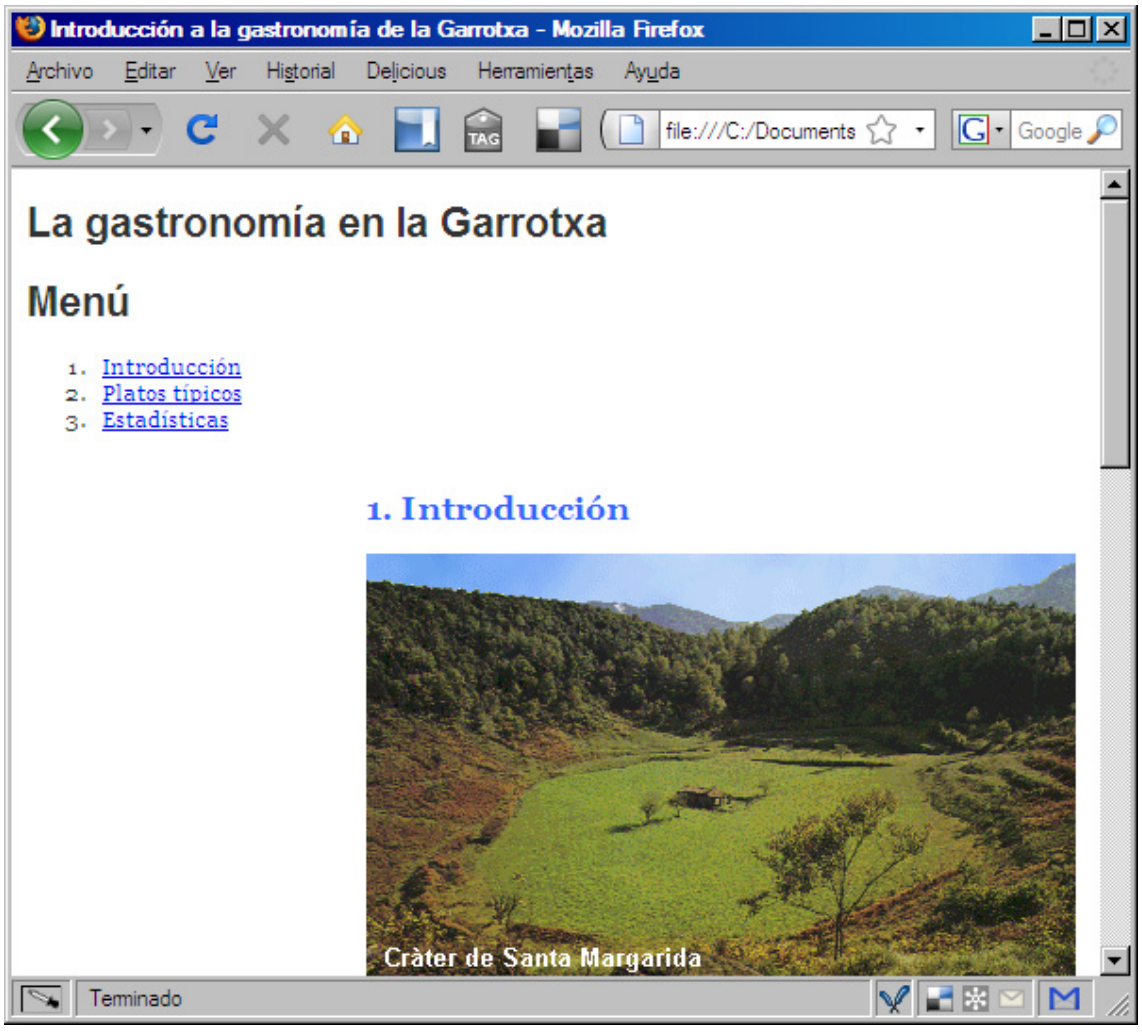
```
p{
border: 1px solid #FFFFFF;
}

. caja {
border-bottom: 1px dashed #FF0000; border-top: 4px groove #00FF00; border-left:1px double #00FF00; border-right: 2px solid #0000FF;
}
```

Añadiremos el siguiente código en el .css para conseguir que nuestra tabla tenga un pequeño filete:

```
body {
padding-left: 5px;
font-family: Georgia,"Times New Roman", Serif;
color: #333333;
font-size: 11px;
}

h1 {
font-family: Helvetica, Geneva, Arial, Sans-serif;
}

h2 {
color:#3366FF;
}

#cabecera {
padding-left:5px;
}

#menu {
padding-top:5px;
padding-bottom:5px;
padding-left: 5px;
}

#contenido {
margin: 30px 10px 0px 32%;
padding-left:5px;
}

#contenido table {
```

```
border:1px solid #3366FF;
}

#contenido address {
margin-top:10px;
}

#contenido img { padding-right:10px;
padding-bottom:5px;
}
```

Vemos cómo se visualiza en nuestro navegador:




Hemos añadido un filete de 1 píxel de grueso, de estilo "sólido" y de color azul.

### 2.5.6. Border-width

Especifica el grueso del borde. Este valor tiene que ser bastante grande para apreciarlo correctamente. Podemos especificar a la vez las cuatro propiedades del grueso del borde. Cada uno de los valores corresponde al grueso del borde superior, derecho, inferior e

izquierdo, respectivamente. Si sólo tenemos un valor, se aplica a todos los lados. Si tenemos dos o tres valores, los valores restantes se toman del lado opuesto, tal como hemos visto con el "margin" o el "padding".

Los valores que se pueden especificar son: una de las tres palabras clave ("thin", "médium" o "tic") o un número entre el 1 y el 4. Es una propiedad que no hereda del padre el valor si no se especifica, es aplicable a elementos y la soportan todos los navegadores.

```
p{
border-width:thin;
}
```

### 2.5.7. Border-top-width, border-right-width, border-bottom-width, border-left-width

Especifican la anchura del borde superior, lateral derecho, inferior o lateral izquierdo de un elemento.

Los valores que se pueden especificar son: una de las tres palabras clave ("thin", "médium" o "thick") o un número entre el 1 y el 4. Es una propiedad que no hereda del padre el valor si no se especifica, es aplicable a todos los elementos y la soportan todos los navegadores.

```
.caixa {
border-bottom-width: 1px;
border-top-width: 4px;
border-left-width: 1px;
border-right-width: 2px;
}
```

### 2.5.8. Border-style

Especifica el estilo de hasta los cuatro bordes de la caja de un elemento a la vez. Los valores que se pueden especificar son una de las siguientes palabras clave: none, dotted, dashed, solid, double, groove, ridge, inset, outset, hidden. Es una propiedad que no hereda del padre el valor si no se especifica, es aplicable a elementos y la soportan todos los navegadores.

```
p{
border-style:double;
}
```

### 2.5.9. Border-top-style, border-right-style, border-bottom-style, border-left-style

Especifica el estilo del borde de un lado específico del elemento.

Los valores que se pueden especificar son una de las siguientes palabras clave: none, dotted, dashed, solid, double, groove, ridge, inset, outset, hidden. Es una propiedad que no hereda del padre el valor si no se especifica, es aplicable a todos los elementos y la soportan todos los navegadores.

```
.caixa {
border-bottom-style: dotted;
border-top-style: groove;
border-left-style: dashed;
border-right-style: solid;
}
```

### 2.5.10. Border-color

Especifica el color de hasta los 4 bordes de la caja de un elemento a la vez.

Los valores que se pueden especificar pueden ser: un valor rgb, un valor hexadecimal, "transparente" o uno de los 17 nombres de color predeterminados que hemos visto antes en la propiedad "color". Por defecto, coge la propiedad de color del elemento. Es una propiedad que no hereda del padre el valor si no se especifica, es aplicable a todos los elementos y la soportan todos los navegadores.

```
p{
border-color: #000000;
}
```

### 2.5.11. Border-top-color, border-right-color, border-bottom-color, border-left-color

Especifica el color de un borde concreto de la caja del elemento.

Los valores que se pueden especificar son: un valor rgb, un valor hexadecimal, "transparente" o uno de los 17 nombres de color predeterminados que hemos visto antes en la propiedad "color". Por defecto coge la propiedad de color del elemento. Es una propiedad que no hereda del padre el valor si no se especifica, es aplicable a todos los elementos y la soportan todos los navegadores.

```
.caixa {
border-bottom-color: #CC9933;
border-top-color: #00FF99;
border-left-color: #999900 ;
border-right-color: #33CCCC;
}
```

### 2.5.12. Width

Especifica la anchura del área de contenido de un elemento.

Los valores que se pueden especificar pueden ser: uno, un porcentaje, o "auto". Es una propiedad que no hereda, es aplicable a todos los elementos de blog y elementos de sustitución, y la soportan todos los navegadores.

```
#content{
width: 100%;
}
```

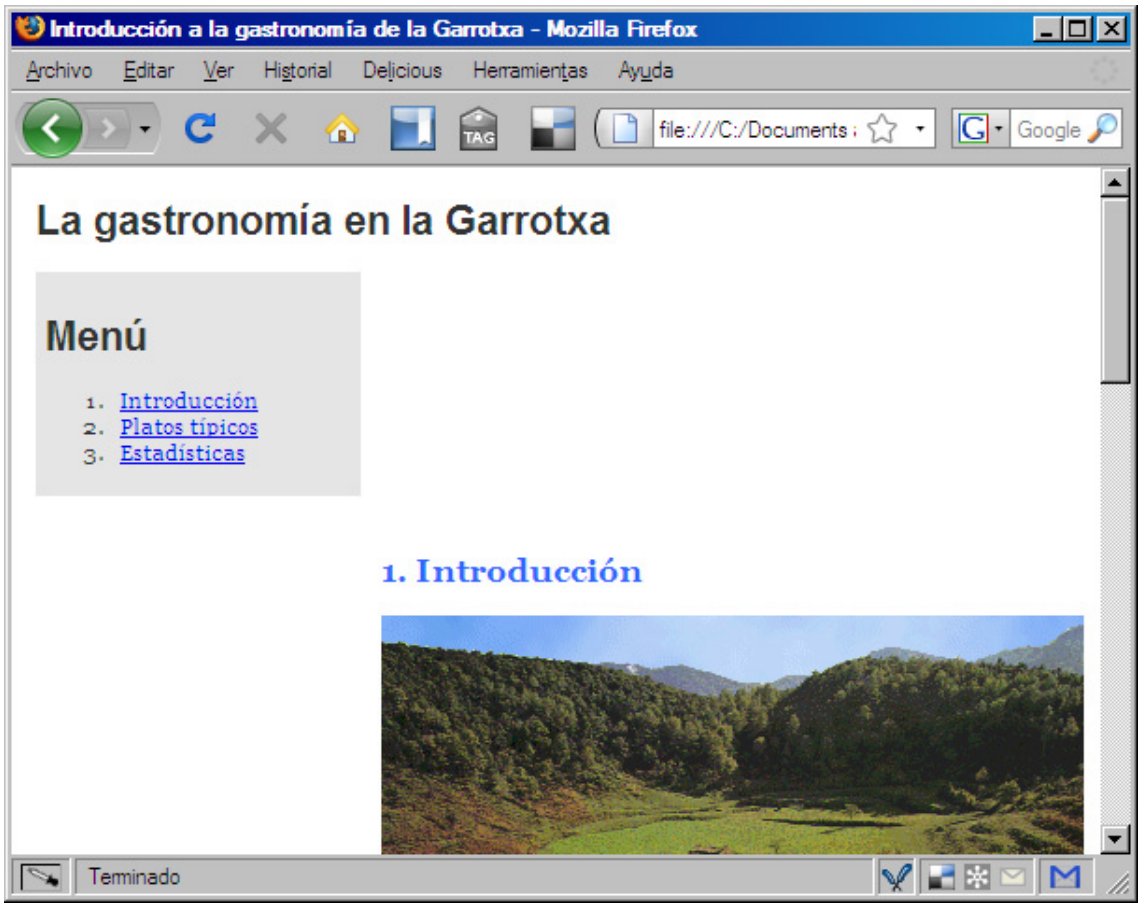
A continuación, especificaremos una anchura en la parte del menú de nuestro ejemplo:

```
body {
padding-left: 5px;
font-family: Georgia,"Times New Roman", Serif;
color: #333333;
font-size: 11px;
}

h1 {
font-family: Helvetica, Geneva, Arial, Sans-serif;
}
```

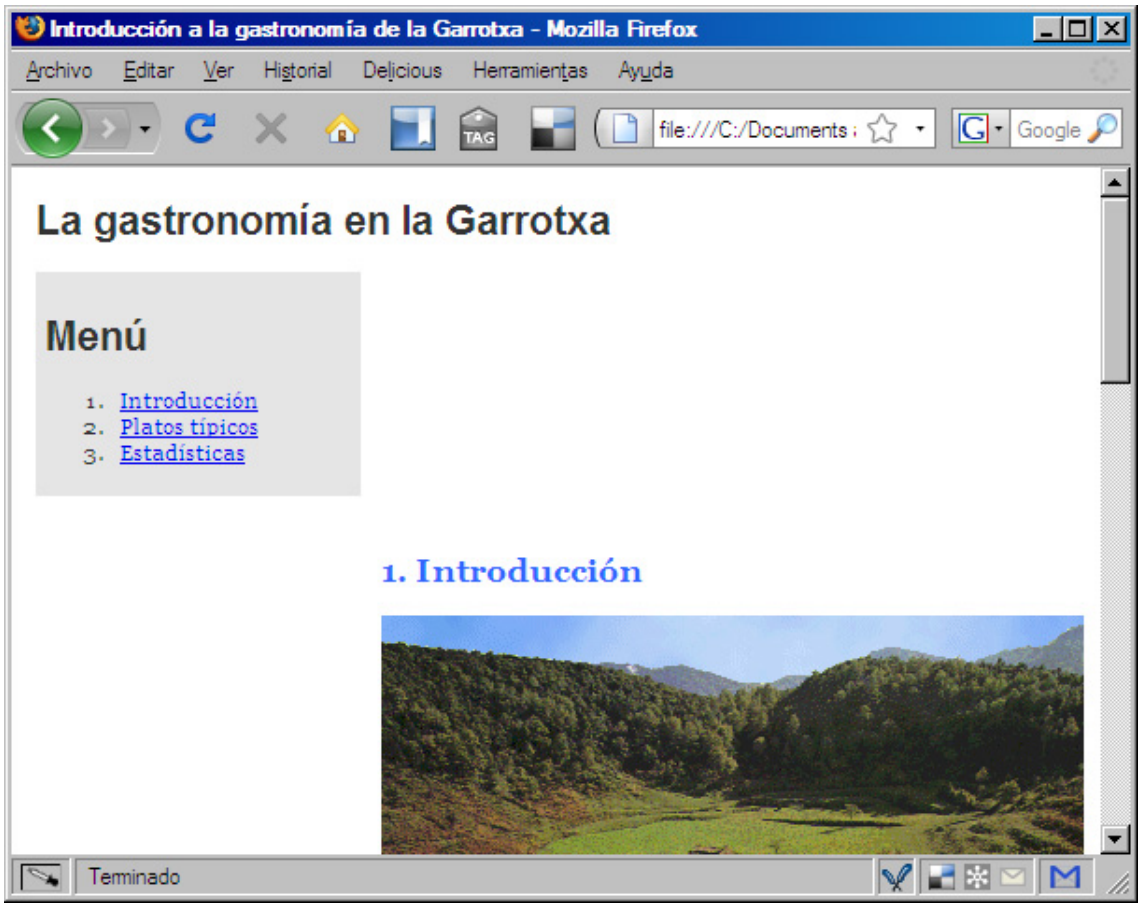
```
h2 {
color:#3366FF;
}

#cabecera {
padding-left:5px;
}

#menu {
width: 30%;
background-color:#e5e5e5;
padding-top:5px;
padding-bottom:5px;
padding-left: 5px;
}

#contenido {
margin: 30px 10px 0px 32%;
padding-left:5px;
}

#contenido table {
border:1px solid #3366FF;
}

#contenido address {
margin-top:10px;
}

#contenido img {
padding-right:10px;
padding-bottom:5px;
}
```

Para poder visualizar cómo afecta el hecho de haber asignado una medida relativa del 30% a nuestro menú, hemos añadido un color de fondo al menú.

Veremos cómo se visualiza en el navegador este código:




Para comprobar el efecto de haber asignado una medida relativa al menú, podemos estirar y encoger la ventana de nuestro navegador. De esta manera, podremos ver cómo el contenido se ajusta a la medida especificada:




### 2.5.13. Min-width

Protege el elemento de ser demasiado estrecho.

Los valores que se pueden especificar pueden ser: una medida o un porcentaje. Por defecto es "0". Es una propiedad que no hereda del padre el valor si no se especifica, es aplicable a todos los elementos excepto los elementos de no sustitución y los elementos de tabla, y la soportan todos los navegadores excepto el Internet Explorer.

```
#content{
min-width: 25px;
}
```

### 2.5.14. Max-width

Protege el elemento de ser demasiado amplio.

Los valores que se pueden especificar pueden ser: una medida, un porcentaje o "none". Por defecto es "none". Es una propiedad que no hereda del padre el valor si no se especifica, es aplicable a todos los elementos, excepto a los elementos de no sustitución y a los elementos de tabla, y la soportan todos los navegadores excepto Internet Explorer.

```
#content{
max-width: 100%;
}
```

### 2.5.15. Height

Especifica la altura del área de contenido del elemento.

Los valores que se pueden especificar pueden ser: una medida, o "auto". Por defecto es "auto". Es una propiedad que no hereda del padre el valor si no se especifica, es aplicable a todos los elementos de blog y los elementos de sustitución, y la soportan todos los navegadores excepto Internet Explorer.

```
#content{
height: 100%;
}
```

### 2.5.16. Min-height

Protege el elemento de ser demasiado pequeño en altura.

Los valores que se pueden especificar pueden ser: una medida o un porcentaje. Por defecto es "0". Es una propiedad que no hereda del padre el valor si no se especifica, es aplicable a todos los elementos excepto los elementos de no sustitución y los elementos de tabla, y la soportan todos los navegadores excepto Internet Explorer.

```
#content{
min-height: 20px;
}
```

### 2.5.17. Max-height

Protege el elemento de ser demasiado alto.

Los valores que se pueden especificar pueden ser: una medida, o "auto". Por defecto es "auto". Es una propiedad que no hereda del padre el valor si no se especifica, es aplicable a todos los elementos de blog y los elementos de sustitución, y la soportan todos los navegadores excepto Internet Explorer.

```
#content{
max-height: 100%;
}
```

### 2.5.18. Overflox

Determina la manera de visualizar los elementos hijos de un elemento que no caben en el área de contenido de su padre.

Los **valores** pueden ser:

- **Visible**: si el contenido del elemento excede las dimensiones se mostrará igualmente. El comportamiento puede ser variable dependiendo del navegador.
- **Hidden**: el elemento se mostrará correctamente, mientras que el contenido que exceda de las dimensiones no será visible.
- **Auto**: si el contenido excede las dimensiones del padre o contenedor, aparecerán las barras de "scroll" para posibilitar la lectura.
- **Scroll**: siempre aparecen las barras de "scroll". En caso de que el contenido no exceda las dimensiones, éstas quedarán deshabilitadas. En el momento en que el contenido exceda las dimensiones, se habilitarán.

Por defecto, es "visible". Es una propiedad que no hereda del padre el valor si no se especifica, es aplicable a todos los elementos de blog y los elementos de sustitución, y la soportan todos los navegadores.

```
#content{
overflow: auto;
}
```

### 2.5.19. Clip

Define qué parte del contenido será visible.

Los valores que se pueden especificar pueden ser: una forma, o "auto". La forma se define de la siguiente manera:

```
rect(5px, 110px, 55px, 10px)
```

Donde los valores corresponden a superior, derecha, inferior e izquierda. Este ejemplo se refiere a un rectángulo que empieza 5px desde la parte superior y 10px desde la izquierda y tiene 100px de anchura y 50px de altura.

Por defecto, es "auto". Es una propiedad que no hereda del padre el valor si no se especifica, es aplicable a todos los elementos con posición absoluta, y la soportan todos los navegadores excepto Internet Explorer.

## 2.6. Propiedades de disposición: posicionamiento

Las propiedades que nos permitirán modificar el posicionamiento de los elementos son las siguientes:

### 2.6.1. Display

Determina cómo se mostrará un elemento. Los valores que podemos especificar son:

- **Block**: este valor provoca que un elemento genere una caja de bloque principal, como si hubiera un salto de línea antes y después del elemento.
- **Inline**: este valor provoca que un elemento genere unas o más cajas de línea, es decir, que los elementos se mostrarán uno tras otro en la misma línea mientras quepan allí.
- **List-item**: este valor provoca que un elemento genere una caja de blog principal y una caja de línea de tipo "list-item".
- **Marker**: este valor declara que el contenido generado antes o después de una caja será un marcador.
- **None**: Este valor provoca que un elemento no genere ninguna caja (es decir, el elemento no tiene ningún efecto sobre la composición) y los elementos descendentes tampoco generen cajas. Se tiene que destacar que este valor no crea una caja invisible, sino que hace que el elemento desaparezca del todo. Esta es la diferencia hacia las propiedades sobre visibilidad, que provocan que un elemento pueda ser invisible pero siga ocupando un espacio en la página.
- **Run-in y compact**: estos valores crean cajas de blog o de línea según el contexto y tienen un comportamiento como el que conocemos para las listas de definición.
- **Table, inline-table, table-row-group, table-column, table-column- group, table-header-group, table-footer-group, table-row, table-cell y table-capt**: Estos valores provocan que un elemento se comporte como un elemento tabla.

El valor por defecto de la propiedad display es "inline". Es una propiedad que no hereda del padre el valor si no se especifica, es aplicable a todos los elementos con posición absoluta, y la soportan todos los navegadores.

```
#contenido {
display: blog;
}
```

### 2.6.2. Position

Especifica el método por el cual se determina la posición de la caja del elemento. Los elementos que no son "estáticos" utilizan las propiedades de posicionamiento "top, right, bottom, y left" que veremos más adelante.

Los **valores** que podemos especificar son:

- **Static**: con este valor, la caja se sitúa dentro del flujo normal de la página.
- **Relative**: la posición de la caja se ajusta en relación con su posición normal dentro de la página. Cuando una caja se posiciona relativamente, la caja siguiente se sitúa como si aquélla no se hubiera desplazado.
- **Absolute**: las cajas son sacadas de su flujo natural de página y su posición se especifica con las propiedades "left", "right", "top","bottom". Estas propiedades especifican los desplazamientos con respecto al blog padre, por lo cual los elementos posicionados absolutamente no tienen ninguna influencia en las cajas siguientes.
- **Fixed**: el posicionamiento "fixed" es una subcategoría del posicionamiento absoluto. La única diferencia es que, para una caja posicionada fija, el blog padre es establecido por el acceso visual (la pantalla del monitor) y el elemento no se mueve cuando se realiza un desplazamiento. Eso quiere decir que cuando se hace scroll en la página, estos elementos mantienen su posición fija.

El valor por defecto es "static". Es una propiedad que no hereda del padre el valor si no se especifica, es aplicable a elementos con posición absoluta y la soportan todos los navegadores.

```
#contenido {
position:absolute;
top:20px;
left:20px;
}
```

### 2.6.3. Top, right, bottom, left

Especifica la posición de un elemento fijo, absoluto, o posicionado de forma relativa.

Los valores que se pueden especificar son: una medida, un porcentaje o "auto". Por defecto, es "auto". Es una propiedad que no hereda del padre el valor si no se especifica, es aplicable a elementos con una posición diferente a estática y la soportan todos los navegadores.

En el ejemplo de la propiedad "position" anterior el elemento con id contenido de nuestro documento se colocaría absolutamente en 20px de separación de la parte superior y de la parte izquierda.

### 2.6.4. Float

Mueve un elemento a derecha o izquierda del elemento del que es hijo hasta que no encuentra otro elemento de blog de nivel. Los elementos flotados son eliminados del flujo normal de la página.

Los valores que se pueden especificar son: "left", "right" o "none". Por defecto, es "none". Es una propiedad que no hereda del padre el valor si no se especifica, es aplicable a todos los elementos y la soportan todos los navegadores.

```
#contenido {
float: left;
```

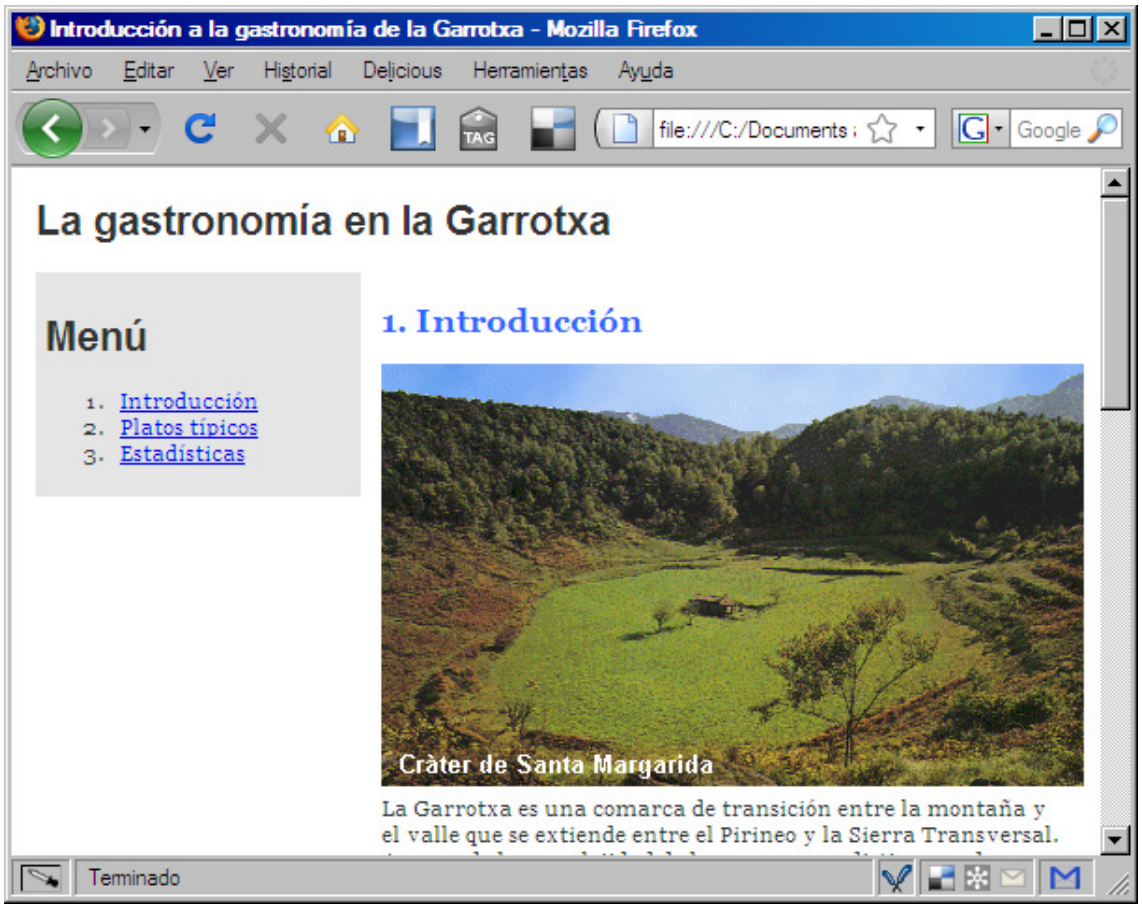
```
}
```

Nuestro siguiente objetivo es hacer que el menú quede a la izquierda del contenido; por lo tanto, añadiremos un "float:left" a nuestro código:

```
body {
padding-left: 5px;
font-family: Georgia,"Times New Roman", Serif;
color: #333333;
font-size: 11px;
}

h1 {
font-family: Helvetica, Geneva, Arial, Sans-serif;
}

h2 {
color:#3366FF;
}

#cabecera {
padding-left:5px;
}

#menu {
float: left;
width: 30%;
background-color:#e5e5e5;
padding-top:5px;
padding-bottom:5px;
padding-left: 5px;
}

#contenido {
margin: 30px 10px 0px 32%;
padding-left:5px;
}

#contenido table {
border:1px solid #3366FF;
}

#contenido address {
margin-top:10px;
}
```

```
#contenido img {
float:left;
padding-right:10px;
padding-bottom:5px;
}
```

Vemos ahora cómo se visualiza el código en el navegador:




### 2.6.5. Clear

Esta propiedad indica cuáles de los lados de la caja de un elemento no pueden quedar adyacentes a una caja flotante anterior. Esta propiedad sólo se puede especificar para elementos de flujo de blog (incluyendo también los elementos flotantes).

Los valores tienen los siguientes **significados**:

- **Left**: el margen superior de la caja generada se aumenta bastante para que su parte superior quede justo debajo de la parte inferior de cualquier caja flotante a la izquierda que aparezca antes en el documento.
- **Right**: el margen superior de la caja generada se aumenta bastante para que su parte superior quede justo debajo de la parte inferior de cualquier caja flotante a la derecha que aparezca antes en el documento.
- **Both**: la caja generada se mueve debajo de todas las cajas flotantes que aparezcan antes en el documento.

- **None**: no existe ninguna restricción a la posición de la caja del elemento respecto de los elementos flotantes.

Por defecto, es "none". Es una propiedad que no hereda del padre el valor si no se especifica, es aplicable a todos los elementos y la soportan todos los navegadores.

```
#contenido {
clear: both;
}
```

### 2.6.6. Z-index

Especifica en qué orden de superposición se apilan los elementos unos sobre los otros. Nos dice el nivel de profundidad en que se encuentra cada elemento. Normalmente, los últimos elementos que aparecen en el código del documento se apilan sobre los elementos que aparecen antes. Cuanto mayor sea el "z-index", más por debajo de él estará.

Los valores que se pueden especificar pueden ser: un número entero, o "auto". Por defecto, es "auto". Es una propiedad que no hereda del padre el valor si no se especifica, es aplicable a elementos posicionados, y la soportan todos los navegadores.

```
#contenido {
position:absolute;
z-index: 4;
}
```

### 2.6.7. Visibility

Hace que un elemento sea completamente transparente sin eliminarlo del flujo del documento.

Los valores que podemos especificar son:

- **Visible**: el elemento es visible.
- **Hidden**: el elemento es invisible (totalmente transparente), pero sigue afectando a la composición.
- **Collapse**: si se utiliza en elementos que no sean filas o columnas de una tabla, "collapse" tiene el mismo significado que "hidden".

Por defecto, es "visible". Es una propiedad que hereda del padre el valor si no se especifica, es aplicable a todos los elementos y la soportan todos los navegadores.

En el siguiente ejemplo el contenido estaría escondido.

```
#contenido {
visibility:hidden;
}
```

## 2.7. Propiedades de disposición: fondo

Las propiedades que nos permitirán modificar el fondo de los elementos son las siguientes:

### 2.7.1. Background-attachment

Determina si una imagen aparece fijada o acompañará el contenido si desplazamos la página haciendo scroll.

Los valores que se pueden especificar pueden ser: "scroll" o "fixed". Por defecto, es "scroll". Es una propiedad que no hereda del padre el valor si no se especifica, es aplicable a elementos, y la soportan todos los navegadores.

```
#contingut {
background-attachment: fixed;
}
```

### 2.7.2. Background-color

Especifica el color de fondo del contenido del elemento.

Los valores que se pueden especificar pueden ser: un valor rgb, un valor hexadecimal, "transparente", o uno de de los 17 nombres de colores predefinidos. Por defecto, es "transparente". Es una propiedad que no hereda del padre el valor si no se especifica, es aplicable a elementos y la soportan todos los navegadores.

Antes hemos utilizado el background-color para mostrar visualmente el efecto de la anchura del menú. Ahora añadiremos colores de fondo a otras partes de nuestro archivo:

```
body {
padding-left: 5px;
font-family: Georgia, "Times New Roman", Serif;
color: #333333;
background-color: #d6d6d6;
font-size: 11px;
}

h1 {
font-family: Helvetica, Geneva, Arial, Sans-serif;
}

h2 {
color:#3366FF;
}

#cabecera {
padding-left:5px;
}

#menu { float: left;
width: 30%;
background-color:#e5e5e5;
padding-top:5px;
padding-bottom:5px;
padding-left: 5px;
}

#contenido {
```

```
margin: 30px 10px 0px 32%;
padding-left:5px;
}

#contenido table {
border:1px solid #3366FF;
}

#contenido address {
margin-top:10px;
}

#contenido img {
float:left;
padding-right:10px;
padding-bottom:5px;
}
```

Si visualizamos el código anterior en un navegador veremos que el efecto es bastante evidente:


### 2.7.3. Background-image

Con esta propiedad podemos especificar una imagen de fondo para un elemento.

Los valores que se pueden especificar pueden ser: una dirección URL o "none". Por defecto es "none". Es una propiedad que no hereda del padre el valor si no se especifica, es aplicable a elementos y la soportan todos los navegadores.

```
#contenido {
background-image:url(/imgs/mural.jpg);
}
```

### 2.7.4. Background-position

Especifica la posición inicial de la imagen de fondo del elemento.

Como valores, podemos especificar una posición horizontal seguida de una posición vertical. Cada una de las dos posiciones las podemos especificar indicando una medida, un porcentaje o una de las 5 palabras clave de posicionamiento. Por defecto es "0 0".

Es una propiedad que no hereda del padre el valor si no se especifica, es aplicable a elementos y la soportan todos los navegadores.

```
#contenido {
background-position:left top;
}
```

### 2.7.5. Background-repeat

Determina de qué manera la imagen de fondo se repite en el elemento.

Los valores que podemos especificar son: repeat, repeat-x, repeat-y, y no-repeat.

- **repeat**: la imagen se repite horizontal y verticalmente.
- **repeat-x**: la imagen se repite sólo horizontalmente.
- **repeat-y**: la imagen se repite sólo verticalmente.
- **no-repeat**: la imagen no se repite. Por defecto es "repeat".

Es una propiedad que no hereda del padre el valor si no se especifica, es aplicable a elementos y la soportan todos los navegadores.

```
#contenido {
background-repeat:repeat-x;
}
```

### 2.7.6. Background

Nos permite especificar las 5 propiedades del background al mismo tiempo. Los valores a especificar son los valores de las propiedades de background-attachment background-color background-image background-position background-repeat. Si se omite algún valor, se aplicará el valor por defecto de la propiedad.

Es una propiedad que no hereda del padre el valor si no se especifica, es aplicable a elementos y la soportan todos los navegadores.

```
#contenido {
background: fixed url(/imgs/ad.png) left top no repeat;
}
```

## 2.8. Elementos

En este capítulo, veremos las propiedades específicas de ciertos elementos. Básicamente, las listas y las tablas.

### 2.8.1. Listas

Comenzamos con las propiedades específicas que podemos dar a las listas de un documento (ul, ol, dl).

**List-style-type**

Especifica el tipo de marcador de los ítems de la lista. Los valores que podemos especificar son:

- **disc**: un disco.
- **circle**: un círculo.
- **square**: un cuadrado.
- **decimal**: números decimales empezando por 1.
- **decimal-leading-zero**: números decimales empezando por 0.
- **lower-roman**: números romanos en minúsculas.
- **upper-roma**: números romanos en mayúsculas.
- **lower-greek**: letras griegas en minúsculas (alfa/α, beta/β, gamma/γ).
- **lower-latin**: letras ASCII en minúsculas.
- **upper-latin**: letras ASCII en mayúsculas.
- **armenian**: numeración armenia tradicional (ayb/ayp, ben/pen, gim/keem...).
- **georgian**: numeración georgiana tradicional
- **lower-alpha**: igual que lower-latin
- **upper-alpha**: igual que upper-latin.
- **none**: no saldrá ningún marca.

El valor por defecto es "disc".

Es una propiedad que hereda del padre el valor si no se especifica, es aplicable a elementos de una lista y la soportan todos los navegadores.

```
li {
list-style-type:georgian;
}
```

Cambiaremos, en nuestro ejemplo, el estilo de la visualización de la lista ordenada; para hacerlo, nos referiremos a los elementos ol dentro de la capa menú:

```
body {
padding-left: 5px;
font-family: Georgia, "Times New Roman", Serif;
color: #333333;
background-color: #d6d6d6;
font-size: 11px;
}

h1 {
font-family: Helvetica, Geneva, Arial, Sans-serif;
}

h2 {
color:#3366FF;
}

#cabecera {
```

```
padding-left:5px;
}

#menu { float: left; width: 30%;
background-color:#e5e5e5;
padding-top:5px;
padding-bottom:5px;
padding-left: 5px;
}

#menu ol {
list-style-type:circle;
}

#contenido {
margin: 30px 10px 0px 32%;
padding-left:5px;
}

#contenido table {
border:1px solid #3366FF;
}

#contenido address {
margin-top:10px;
}

#contenido img {
float:left;
padding-right:10px;
padding-bottom:5px;
}
```

Si lo visualizamos en un navegador, veremos lo siguiente:


Podemos comprobar cómo la lista ordenada del menú que antes aparecía con números ahora nos aparece con círculos.

### List-style-position

Especifica la posición del marcador de la lista en relación con la caja principal. Los valores que se pueden especificar son:

- **outside**: el marcador se sitúa fuera de la caja principal
- **inside**: el marcador queda en el interior de la caja junto con el texto.

El valor por defecto es "outside".

Es una propiedad que hereda del padre el valor si no se especifica, es aplicable a todos los elementos de una lista y la soportan todos los navegadores.

```
li {
```

```
list-style-type:georgian;
}
```

**List-style-image**

Especifica la imagen que se utilizará como marcador.

Los valores que se pueden especificar son o bien una dirección URL o bien "none". Por defecto será "none".

Es una propiedad que hereda del padre el valor si no se especifica, es aplicable a todos los elementos de una lista y la soportan todos los navegadores.

```
li {
list-style-image:url(/imgs/ad.png);
}
```

**List-style**

Nos permite especificar las propiedades individuales del "list-style" al mismo tiempo.

Los valores a especificar son los valores de las propiedades de list-style-type list-style-position list-style-image. Si se omite algún valor, se aplicará el valor por defecto de la propiedad.

Es una propiedad que hereda del padre el valor si no se especifica, es aplicable a todos los elementos de una lista, y la soportan todos los navegadores.

```
li {
list-style:decimal-leading-zero outside url(/imgs/ad.png);
}
```

### 2.8.2. Tablas

Las propiedades que podemos modificar específicamente en el caso de las tablas son las siguientes:

**Border-collapse**

Determina si las celdas de tablas tienen bordes separados o comparten los bordes con las celdas adyacentes, grupos de filas y columnas o con la misma tabla.

Los valores que se pueden especificar son: "collapse", o "separate". Por defecto, será "separate". Es una propiedad que hereda del padre el valor si no se especifica, es aplicable a elementos de tipo table y la soportan todos los navegadores.

```
table{
border-collapse: separate
}
```

**Border-spacing**

Determina el espacio entre las parejas de bordes.

Podemos darle uno o dos valores numéricos. Si los dos valores están presentes, el primero determina la distancia horizontal y el segundo la distancia vertical. El valor por defecto será "0".

Es una propiedad que hereda del padre el valor si no se especifica, es aplicable a todos los elementos de tipo table con bordes separados, y la soportan todos los navegadores.

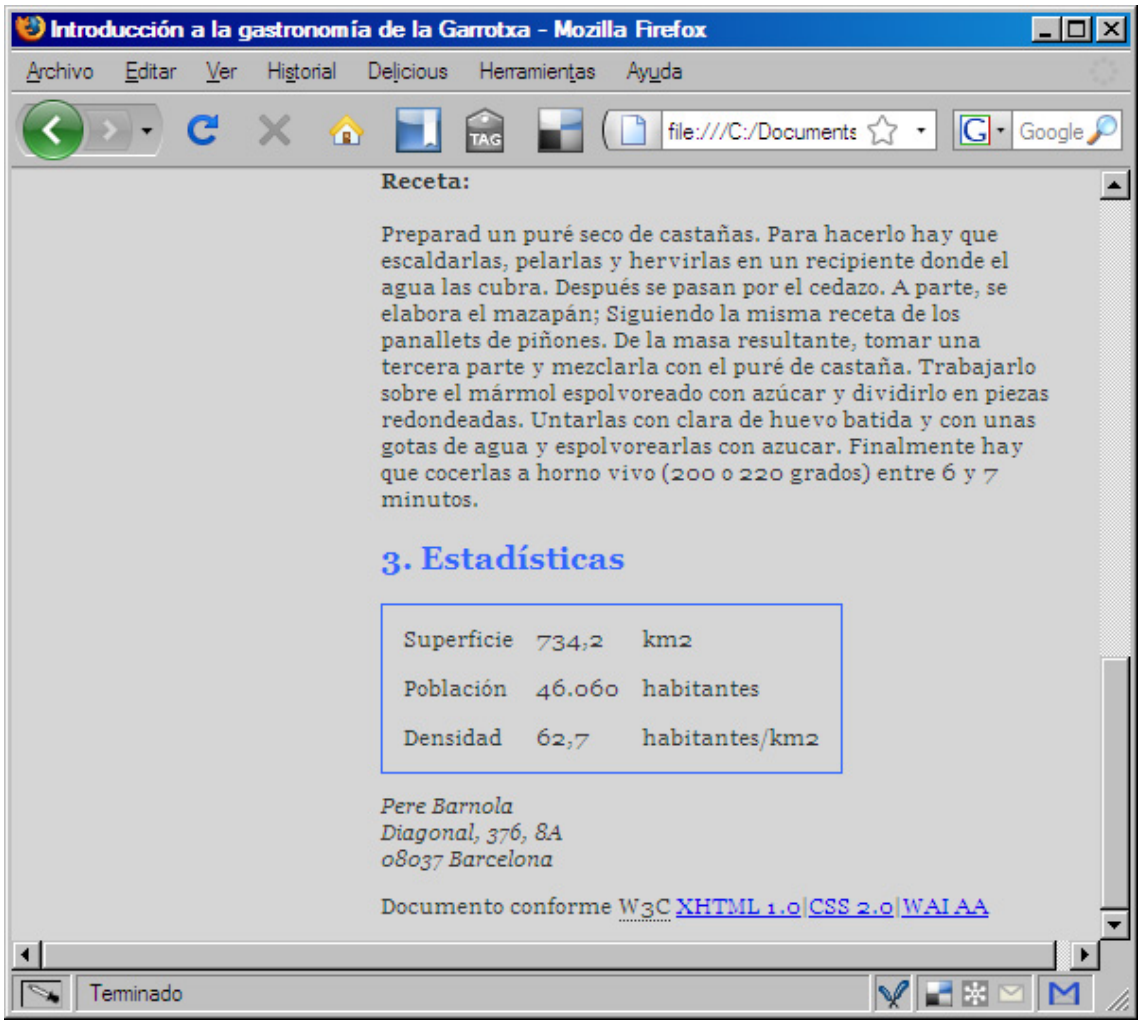
Daremos a nuestro ejemplo un espacio en los bordes de nuestra tabla. Con el fin de hacerlo añadiremos el siguiente código:

```
body {
padding-left: 5px;
```

```
font-family: Georgia, "Times New Roman", Serif;
color: #333333;
background-color: #d6d6d6;
font-size: 11px;
}

h1 {
font-family: Helvetica, Geneva, Arial, Sans-serif;
}

h2 {
color:#3366FF;
}

#cabecera {
padding-left:5px;
}

#menu {
float: left;
width: 30%;
background-color:#e5e5e5;
padding-top:5px;
padding-bottom:5px;
padding-left: 5px;
}

#menu ol {
list-style-type:circle;
}

#contenido {
margin: 30px 10px 0px 32%;
padding-left:5px;
}

#contenido table {
border:1px solid #3366FF;
border-spacing:10px 10px;
}

#contenido address {
margin-top:10px;
```

```
}

#contenido img {
float:left;
padding-right:10px;
padding-bottom:5px;
}
```

Y lo visualizaremos de la siguiente manera:




Vemos cómo ahora hay más espacio entre el contenido y las celdas de nuestra tabla.

**Empty-cells**

Oculta o muestra los bordes de las celdas vacías.

Los valores pueden ser: "show" o "hide". Por defecto, el valor será "show".

Es una propiedad que hereda del padre el valor si no se especifica, es aplicable a todos los elementos de tipo "table" con bordes separados y la soportan todos los navegadores excepto Internet Explorer.

**Table-layout**

Determina qué algoritmo hay que utilizar para el navegador a la hora de visualizar las celdas, filas y columnas de una tabla.

Los valores pueden ser: "auto" o "fixed".

- **fixed**: permite que el navegador renderice más rápidamente el contenido de la tabla que de la forma automática. La presentación horizontal sólo depende de la anchura de la tabla y las columnas, no del contenido de la tabla. Utilizando este método se puede visualizar la tabla una vez la primera fila se ha descargado correctamente.
- **auto**: la anchura de la columna se determina por la anchura mayor del contenido indivisible dentro de la celda. El navegador tiene que leer toda la tabla antes de mostrarla.

Por defecto, el valor será "auto".

Es una propiedad que hereda del padre el valor si no se especifica, es aplicable a todos los elementos de tipo "table" que no tengan la propiedad "width" especificada como "auto", y la soportan todos los navegadores excepto Internet Explorer.

```
table{
table-layout: fixed
}
```

**Caption-side**

Determina la posición del título de la tabla, si éste va encima o debajo de la tabla.

Los valores pueden ser: "top" o "bottom". Por defecto, el valor será "top".

Es una propiedad que hereda del padre el valor si no se especifica lo contrario, es aplicable a todos los elementos de tipo tabla, y la soportan todos los navegadores excepto Internet Explorer.

En el caso de las tablas, hay propiedades que ya hemos visto anteriormente que tienen un efecto especial cuando se aplican a tablas o celdas. Son el "width" y el "vertical-align".

**Width**

Determina una anchura mínima de una tabla o de una celda.

Los valores pueden ser un número, un porcentaje, o "auto". Por defecto será "auto".

Es una propiedad que no hereda del padre el valor si no se especifica, es aplicable a elementos de tipo table, table-column y table-cell y la soportan todos los navegadores.

**Vertical-align**

Determina la alineación vertical del texto dentro de la fila o celda.

Los valores pueden ser: "baseline", "top", "bottom", o "middle". Por defecto será "baseline".

Es una propiedad que no hereda del padre el valor si no se especifica, es aplicable a elementos de tipo "table-cell", y la soportan todos los navegadores.

## 2.9. Otros

### 2.9.1. Herencia

Una de las características más importantes del CSS es la que se conoce como herencia u orden en cascada. Lo que significa trabajar en cascada se puede entender fácilmente a partir del ejemplo siguiente.

Imaginad que queréis crear un documento que tenga toda la tipografía de color gris, de una medida de 12 píxeles, sobre un fondo claro. Para conseguirlo, tendréis que definir la regla siguiente:

```
*{
```

```
color:#33ff00;
background-color:#eeeeee;
font-size:12px;
}
```

Esta regla de selector universal (el asterisco) afectará a todos los elementos del documento. Sin embargo, poned por caso que no queréis que la cabecera del primer nivel tenga el mismo color que el resto de elementos, sino que lo que queréis es que sea de color rojo. Por eso habrá que añadir una segunda regla:

```
h1 {
color: red;
}
```

El navegador sobrescribirá el valor del color selector h1 sobre el selector universal, pero el selector h1 heredará el resto de propiedades de la regla más general, y mostrará, por lo tanto, la misma medida y el mismo color de fondo que el resto de elementos.

Se puede decir que las diferentes reglas compiten entre ellas según un orden de jerarquía o prioridad, de manera que las propiedades de una regla de un orden superior prevalecerán siempre sobre las de un orden inferior.

Este orden en cascada separa las reglas en seis grupos diferentes según el tipo de selector utilizado. El listado de los seis grupos está ordenado de mayor a menor prioridad. En términos generales, tendrán siempre preferencia los tipos de selectores de más precisión frente a los selectores más generales.

Los seis grupos en que quedan agrupadas las reglas, según el tipo de selector utilizado, ordenados de mayor a menor prioridad, son los siguientes:

Propiedades que contengan la expresión "!important" detrás de su valor:

```
h1{
color:red !important;
}
```

Estilos declarados como valor del atributo "style" del elemento XHTML:

```
<h1 style="color red;">Lorem Ipsum</p>
```

Reglas definidas por uno o más selectores del tipo ID:

```
#vermell {
color: red;
}
```

Reglas definidas por uno o más clases, atributos o pseudoselectores:

```
. rojo {
color:red;
}
p:first-letter {
color: red;
}
```

Reglas que contienen uno o más selectores XHTML:

```
h1 {
color: red;
}
```

Reglas que contienen el selector universal:

```
* {
color: red;
}
```

## 2.10. Caso práctico: "dando estilos a nuestra primera web"

A partir de los conocimientos que hemos adquirido, daremos estilo al documento de ejemplo que construimos en el segundo capítulo mediante las hojas de estilo.

La idea es que partáis del código XHTML y CSS que tenéis a continuación, y modifiquéis las reglas para ir observando cómo los diferentes valores que hemos visto en cada una de las propiedades afectan a la visualización del documento.

**Archivo XHTML llamado "index_estilos.html"**

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="es" lang="es">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
<meta name="author" content="Pere Barnola"/>
<meta name="keywords" content="garrotxa, gastronomía"/>
<title>Introducción a la gastronomía de la Garrotxa</title>
<link href="estilos.css" rel="stylesheet" type="text/css" />
</head>
<body>
<div id="cabecera">
<h1>La gastronomía en la Garrotxa</h1>
</div>
<div id="menu">
<h1>Menú</h1>
<ol>
<li><a href="#introduccion" title="Descripción del
taller">Introducción</a></li>
<li><a href="#platos_tipicos" title="Conocimientos a adquirir">Platos
típicos</a></li>
<li><a href="#estadisticas" title="Guías que se
utilizan">Estadísticas</a></li>
</ol>
</div>
<div id="contenido">
<h2>1. Introducción<a name="introduccion"></a></h2>
<img src="foto_garrotxa.gif"
alt="imagen del cráter de Santa Margalida" width="376" height="226" />
<p>La Garrotxa una comarca de transicitre la montaña y la planallanura, que
se estentre el Pirineo y la Cordillera Transversal. Aun la complejidad de
```

```
la comarca se distinguen dos sectores bien diferenciados y delimitados: la Alta Garrotxa y la Baja Garrotxa.

La primera, ferega y de relieve difícil, con poca vegetación llena de desfiladeros, simas y cestos, da nombre a la comarca -Garrotxa, 'tierra áspera y rota'-, y un paradper a los excursionistas; la Baja Garrotxa, de sierras suaves y depresiones volcques, mplanera, con prados abundantes y corrientes de agua</p>

<h2>2. Platos típicos<a name="platos_tipicos"></a></h2>

<h3>PANECILLOS DE CASTAÑA</h3>

<h4>Ingredientes:</h4>

<ul>

<li> 500 gr de almendras ralladas</li>

<li> 450 gr de azúcar</li>

<li> 200 gr de patatas</li>

<li> 25 gr de azúcar lustre</li>

<li> 2 huevos</li>

<li> 250 gr de castañas</li>

</ul>

<h4>Receta:</h4>

<p>Preparad un puré seco de castañas. Para hacerlo, hay que escaldarlas, pelarlas y hervirlas en un recipiente que las cubra con agua. Después, las pasáis por el cedazo. Aparte, elaborad el mazapán; siguiendo la misma receta de los panecillos de piñones. De la masa resultante, tomad una 1/3 parte y mezcladla con el puré de castaña. Trabajadlo sobre el mármol empolvado con azúcar y divididlo en piezas redondeadas. Untadlas con clara de huevo batida con unas gotas de agua y empolvadlas con azúcar. Finalmente, hay que cocerlas en el horno (200 o 220 grados) entre 6 o 7 minutos.</p>

<h2>3. Estadísticas<a name="estadisticas"></a></h2>

<table>

<tr>

<td>Superficie</td>

<td>734,2</td>

<td>km2</td>

</tr>

<tr>

<td>Población</td>

<td>46.060</td>

<td>habitantes</td>

</tr>

<tr>

<td>Densidad</td>

<td>62,7</td>

<td>habitantes/km2</td>

</tr>

</table>

<address>

Pere Barnola<br /> Diagonal, 376, 8A<br />
```

```
08037 Barcelona
</address>
<p>Documento conforme <acronym title="World Wide Web Consortium" xml:lang="en">W3C</acronym> <a href="http://validator.w3.org/" title="Código revisado con el validador del W3C">XHTML 1.0</a>|
<a href="http://jigsaw.w3.org/css-validator/" title="Hojas de estilo revisadas con el validador del W3C">CSS 2.0</a>|
<a href=http://www.tawdis.net/taw3/online
title="Nivel de adecuación doble A, conforme a las directrices de la Web Accessibility Initiative">WAI AA</a></p>
</div>
</body>
</html>
```

**Código del archive estilos.css resultante**

```
body {
padding-left: 5px;
font-family: Georgia, "Times New Roman", Serif;
color: #333333;
background-color: #eeeeee;
}

h1 {
font-family: Helvetica, Geneva, Arial, Sans-serif;
}

h2 {
color:#3366FF;
border-bottom:1px solid #3366FF;
}
a {
color:#3366FF;
}
a:hover {
color:#33CCFF;
}

#cabecera {
border: dotted 1px #33CCFF;
padding-left:5px;
}

#menu {
float: left; width: 30%;
background: #eeeeee;
padding-top:5px;
```

```
padding-bottom:5px;
padding-left: 5px;
background-color:#e5e5e5;
}

#menu ol {
list-style-type:circle;
}

#contenido {
margin: 30px 10px 0px 32%;
padding-left:5px;
}

#contenido table {
border:1px solid #3366FF;
border-spacing:10px 10px;
}


#contenido address {
margin-top:10px;
color: #999999;
font-size:11px;
}

#contenido img {
float:left;
padding-right:10px;
padding-bottom:5px;
}

#contenido ul {
list-style-type: square;
}
```

**Validación final**

Finalmente, una vez hechas las modificaciones, hará falta que os aseguréis de que el documento CSS sea un documento CSS válido mediante el Servicio de Validación de CSS del W3C.

# Formularios y objetos

## 1. Formularios

El formulario es posiblemente la herramienta más utilizada en Internet para obtener datos e información sobre la gente que navega por la web.

La idea de los formularios es recoger información que el usuario, mediante su interacción con éstos, nos envía para, después, ejecutar una determinada acción.

**Ejemplo**

Los casos más comunes en Internet de formularios son, por ejemplo, los buscadores, donde el usuario introduce un texto y la web le devuelve los resultados más adecuados, o por ejemplo, cuando queremos comprar un producto y tenemos que rellenar una serie de datos personales para completar el proceso de compra.

A grandes rasgos, diremos que la información que el usuario de una web introduce en el formulario es enviada para que se procese en el servidor. El servidor es el ordenador donde la web está alojada, y será quien realice los procesos convenientes para después enviar un mensaje de respuesta al usuario. No entraremos en más detalles sobre la parte que realiza el servidor, ya que se va bastante del ámbito de esta asignatura, y no tendríamos bastante tiempo para verlo adecuadamente.

Para hacer más comprensible este punto, iremos construyendo un formulario a medida que vamos estudiando las partes, de la misma manera que hicimos en el módulo 2. Para esta tarea, crearemos un archivo nuevo con el blog de notas que guardaremos con el nombre de "formulario.html". Para empezar, introduciremos el siguiente código, con el cual crearemos un archivo XHTML básico:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="es" lang="es">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>formulario contacto</title>
</head>
<body>
</body>
</html>
```

## 1.1. Elementos

Los elementos interactivos que tenemos a nuestra disposición para crear formularios son los siguientes: <form>, <input>, <select>, <option>  y <textarea>.

Estos elementos son suficientes, como veremos, para indicar a nuestro navegador cómo tiene que recoger la información, y cómo se enviará al servidor.

### 1.1.1. Form

Lo primero que nuestro formulario tiene que tener es el elemento "form". Este elemento no sólo es imprescindible para la construcción de cualquier formulario, sino que todos los elementos que necesitamos para construirlo tienen que estar dentro de este elemento.

Hallamos tres posibles **atributos**:

**1) Action**

Este atributo indica la acción que realizará nuestro formulario. Por ejemplo, si queremos que nuestro formulario envíe un correo electrónico, el código sería una cosa así:

```
<form action=mailto:direccion@correo.com></form>
```

Si lo que queremos es que la información del formulario sea enviada al servidor para que ejecute el proceso, tenemos que indicar la URL del archivo donde se realizará este proceso.

El código lo escribiríamos de esta manera:

```
<form action="/php/proceso_formulario.php"></form>
```

**2) Method**

Este atributo define de qué manera los datos del formulario serán transmitidos al servidor. Las dos opciones que puede tener este atributo es "GET" o "TABLA". Más adelante, veremos en qué consiste cada una.

```
<form action="/php/proceso_formulario.php" method="POST"></form>
```

**3) Enctype**

Con este atributo, especificamos el tipo de encriptación de datos del formulario antes que el servidor las reciba. Sólo funcionará en el caso de que el atributo method tenga como valor "TABLA". Encriptar los datos quiere decir, a grandes rasgos, codificar estos datos antes de enviarlos al servidor.

Los tipos de valores que puede tener este atributo son:

- **"application/x-www-form-urlencoded"**: es el valor por defecto si no le especificamos ninguno. Esta codificación será suficiente para la mayoría de los formularios.
- **"multipart/form-data"**: especificaremos este valor en caso de que en nuestro formulario esté la posibilidad de adjuntar algún tipo de archivo.

```
<form action="/php/proceso_formulario.php" enctype="
application/x-www-form-urlencoded"></form>
```

Siguiendo con nuestro documento de ejemplo, añadiremos el código siguiente:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="es" lang="es">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>formulario contacto</title>
</head>
<body>
<form id="form1" method="post" action="proceso_formulario.php">
</form>
</body>
</html>
```

### 1.1.2. Input

El elemento "input" es el segundo en importancia, ya que podemos decir que éste es un elemento multifunción. Esto quiere decir que con este mismo elemento se pueden generar "radiobuttons", "checkboxes" o campos para introducir texto, entre otros. Para hacerlo, al atributo "type" tendremos que especificar el tipo de elemento que queremos generar.

Los posibles valores de este atributo son: "text", "password", "checkbox", "radio", "submit", "resend", "image" y "hidden". Veremos los más importantes acto seguido.

Otro atributo a tener en cuenta es "**name**". Todos los elementos de tipo "input" tienen que tener este atributo, que nos permitirá asignar un nombre al elemento.

Así nos será posible relacionar el contenido o valor de este elemento con el nombre que hemos especificado.

### 1.1.3. Campos de texto

Para crear una caja de texto donde el usuario pueda introducir texto, lo haremos indicando al atributo "type" el valor "text":

```
<input type="text" >
```

Los atributos que pueden tener este tipo de elementos y sus funcionalidades son los siguientes:

- **Maxlenght**. Determina el máximo de caracteres que se pueden introducir en la caja de texto.
- **Size**. Con este atributo podemos especificar la medida de la caja de texto que verá al usuario.
- **Value**. El valor que contenga este atributo indicará el texto que, por defecto, saldrá inicialmente en la caja de texto.
- **Disabled**. Nos sirve para desactivar la caja de texto. Una vez desactivada, el usuario no podrá escribir nada en el campo de texto.
- **Tabindex**. Especifica el orden del tabulador para este elemento con respecto a los otros elementos del formulario. Es decir, cada vez que el usuario haga clic en la tecla "TABULADOR" de su teclado, se activará el siguiente elemento según el orden que tenga en su atributo "tabindex".
- **Alt**. Sirve para indicar una pequeña descripción al elemento.

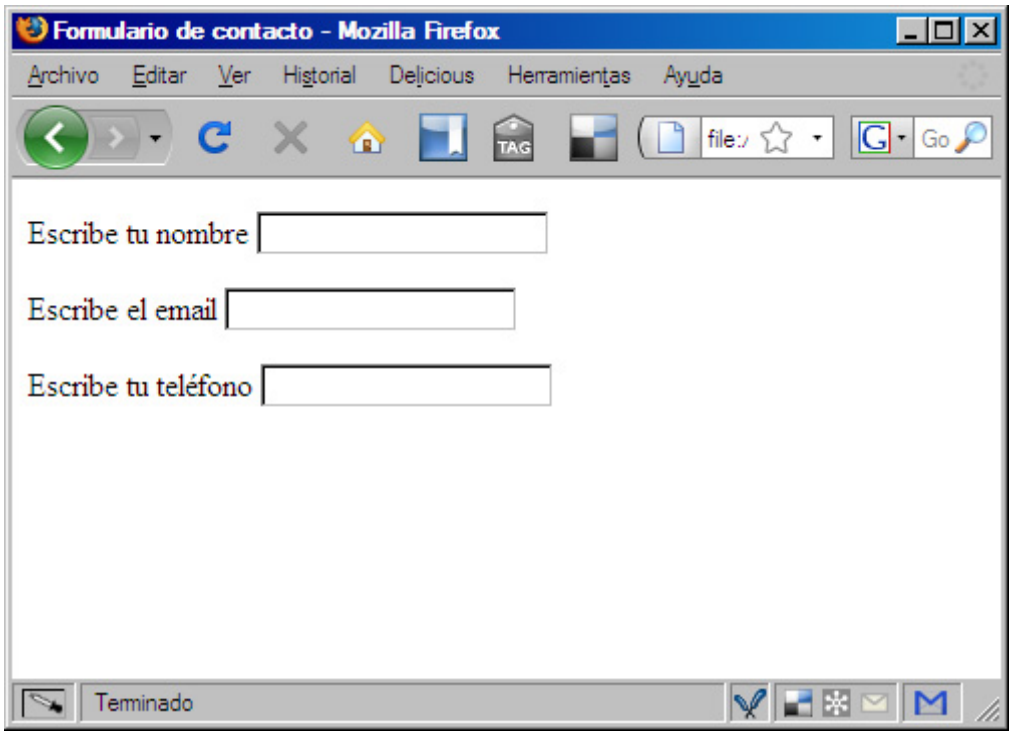
Introduciremos unos cuantos campos de texto en nuestro documento de ejemplo:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="es" lang="es">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>formulario contacto</title>
</head>
<body>
<form id="form1" method="post" action="proceso_formulario.php">
<p>
<label>Escribe tu nombre
<input type="text" name="nom" id="nom" />
</label>
```

```
</p>
<p>
<label>Escribe el email
<input type="text" name="apellido" id="apellido" />
</label>
</p>
<p>
<label>Escribe tu teléfono
<input type="text" name="telefono" id="telefono" />
</label>
</p>
</form>
</body>
</html>
```

Como vemos, en el código nos encontramos, aparte del elemento <input> que acabamos de estudiar, el elemento <label>. Este elemento lo explicaremos con detalle más adelante, pero lo incluiremos ahora para hacer más comprensible la elaboración del formulario:

Si visualizamos ahora este archivo en un navegador, lo veremos así:




Como vemos en la imagen, tenemos tres cajas de texto donde el usuario podrá introducir los diferentes datos que se piden (nombre, correo electrónico y teléfono).

### 1.1.4. Radiobuttons y checkboxes

Tanto los radiobuttons como los checkboxes son unos botones especiales que permiten al usuario escoger una determinada opción.

**Ejemplo**

Por ejemplo, si en nuestro formulario queremos hacer una pregunta al usuario del estilo "¿Tienes carné de conducir?", podremos indicarle las diferentes respuestas ("sí" y "no") con dos radiobuttons, uno para cada posible respuesta.

**La diferencia mayor entre los radiobuttons y los checkboxes es que los primeros permiten al usuario seleccionar una entre varias opciones, mientras que los segundos dan la posibilidad de escoger una o más opciones de respuesta.**

Pasaremos a comentar cada uno en detalle.

### Radiobuttons

Como hemos avanzado antes, los radiobuttons permiten que el usuario sólo pueda escoger una de las posibles opciones de selección cuando el atributo "name" de los radiobuttons es el mismo. De hecho, cuando hacemos clic sobre un radiobutton, éste quedará seleccionado y el resto de radiobuttons que tengan el atributo "name" igual se deseleccionarán.

Gráficamente, los radiobuttons se visualizan como pequeños botones redondos que, una vez están seleccionados, aparecen con un pequeño icono redondo dentro.

Veremos un ejemplo de ello y lo comentaremos:

```
<p>¿Te gusta nuestra web? <br />
<label>
<input type="radio" name="gusta" id="si" value="si" checked="true" />
Si</label>
<br />
<label>
<input type="radio" name="gusta" id="no" value="no" /> No</label>
</p>
```

Como vemos, todo el conjunto de radiobuttons que forman las diferentes respuestas a una misma pregunta (en nuestro ejemplo es "¿Te gusta nuestra web?"), llevan el mismo valor para el atributo "name". No obstante, el valor del atributo "value" sí que es diferente para cada opción. Esto hará que, si seleccionamos la primera opción, el servidor recibirá la siguiente información:

[gusta=sí]

y si seleccionamos la segunda, el servidor recibirá:

[gusta=no]

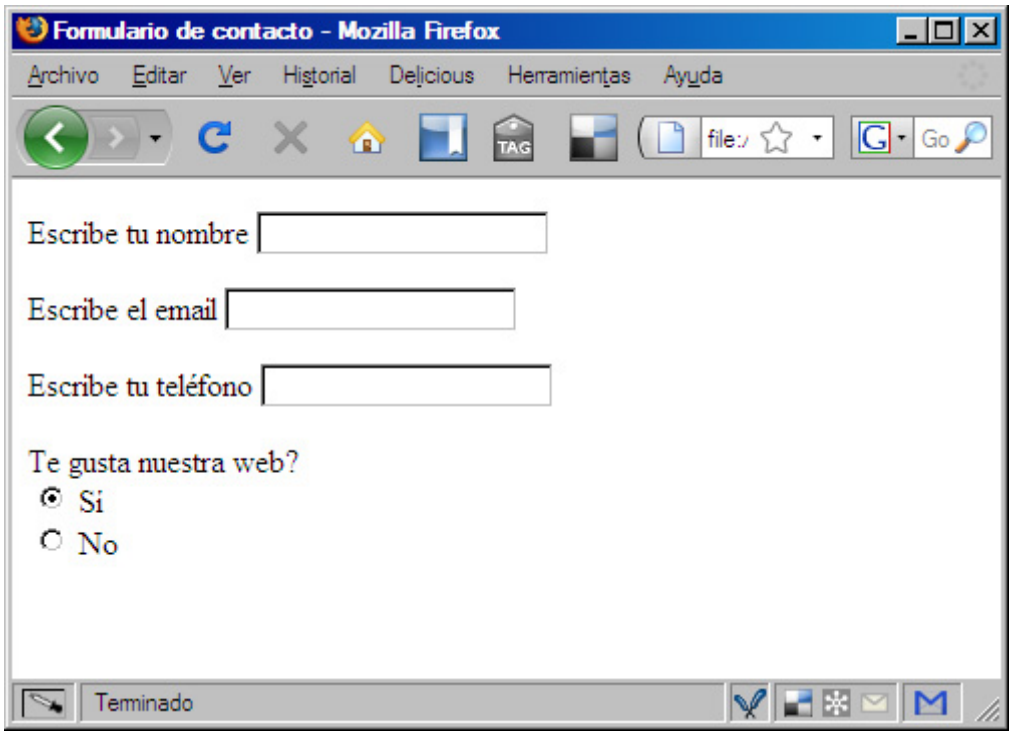
Otro atributo que vemos que presenta el primer radiobutton es "checked". Con este atributo indicamos cuál de las opciones tiene que aparecer como seleccionada por defecto, en caso de que así lo quisiéramos hacer.

Volvamos al archivo que estamos creando y añadamos los radiobuttons:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="es" lang="es">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>formulario contacto</title>
</head>
```

```
<body>
<form id="form1" method="post" action="proceso_formulario.php">
<p>
<label>Escribe tu nombre
<input type="text" name="nombre" id="nombre" />
</label>
</p>
<p>
<label>Escribe el e-mail
<input type="text" name="apellido" id="apellido" />
</label>
</p>
<p>
<label>Escribe tu teléfono
<input type="text" name="telefono" id="telefono" />
</label>
</p>
<p>¿Te gusta nuestra web? <br />
<label>
<input  type="radio"  name="gusta"  id="si"  value="si"  checked="true"  />
Sí</label>
<br />
<label>
<input type="radio" name="gusta" id="no" value="no" /> No</label>
</p>
</form>
</body>
</html>
```

Vemos cómo lo visualiza el navegador:




Podemos ver, tal como hemos explicado, que de las dos opciones de respuesta que damos a la pregunta "¿te gusta nuestra web?", una sale marcada por defecto con una redonda verde dentro del radiobox de la respuesta "sí".

### Checkboxes

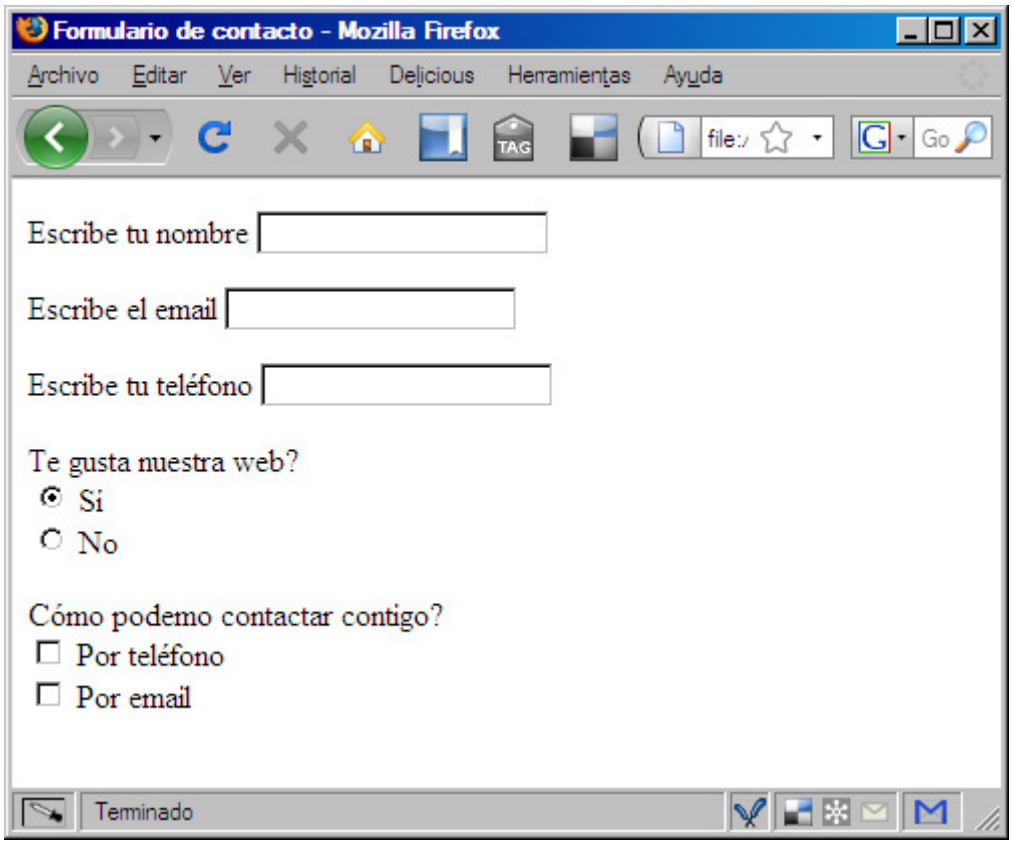
Los checkboxes, a diferencia de los radiobuttons, mantienen el mismo valor para el atributo "value", pero cambian el valor del atributo "name". Es así porque las respuestas pueden ser múltiples; por lo tanto, no hay una relación entre ellas de forma excluyente.

Lo veremos claramente añadiendo unos checkboxes en nuestro archivo de ejemplo:

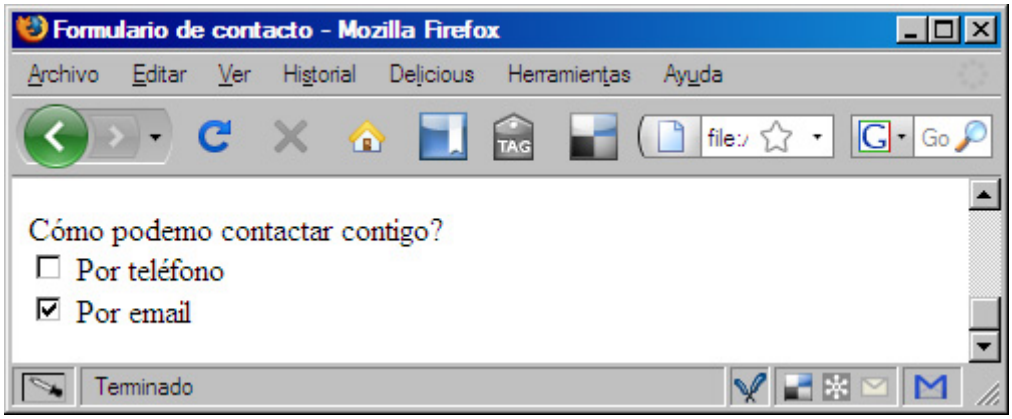
```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="es" lang="es">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>formulario contacto</title>
</head>
<body>
<form id="form1" method="post" action="proceso_formulario.php">
<p>
<label>Escribe tu nombre
<input type="text" name="nombre" id="nombre" />
</label>
</p>
<p>
<label>Escribe el e-mail
<input type="text" name="apellido" id="apellido" />
```

```
</label>
</p>
<p>
<label>Escribe tu teléfono
<input type="text" name="telefono" id="telefono" />
</label>
</p>
<p>¿Te gusta nuestra web? <br />
<label>
<input  type="radio"  name="gusta"  id="si"  value="si"  checked="true"  />
Sí</label>
<br />
<label>
<input type="radio" name="gusta" id="no" value="no" /> No</label>
</p>
<p>¿Cómo podemos contactar contigo? <br />
<label>
<input     type="checkbox"     name="contactar"     id="contactar_telefono"
value="contactar_telefono"/>
Por teléfono
</label>
<br />
<label>
<input      type="checkbox"      name="contactar"      id="contactar_email"
value="contactar_email" /> Por email
</label>
</p>
</form>
</body>
</html>
```

Y si lo visualizamos en un navegador tendríamos este resultado:




Como veremos en la imagen, la diferencia entre los checkboxes y los radiobuttons también se ve en lo visual. Los checkboxes aparecen como botones cuadrados que, cuando están seleccionados, se muestran con una "marca" dentro. Lo vemos en la siguiente imagen:




De la misma manera que hemos visto con los radiobuttons, podemos hacer uso del atributo "checked" para marcar una opción por defecto.

### 1.1.5. Botones de acción

Los botones de acción son útiles en los formularios para enviar los datos, ejecutar funciones personalizadas, etc.

Los botones de acción más comunes en los formularios son los de **submit** (envío de los datos) y los de **reset** (restablece los campos del formulario a sus valores por defecto). Veremos en profundidad estos dos tipos.

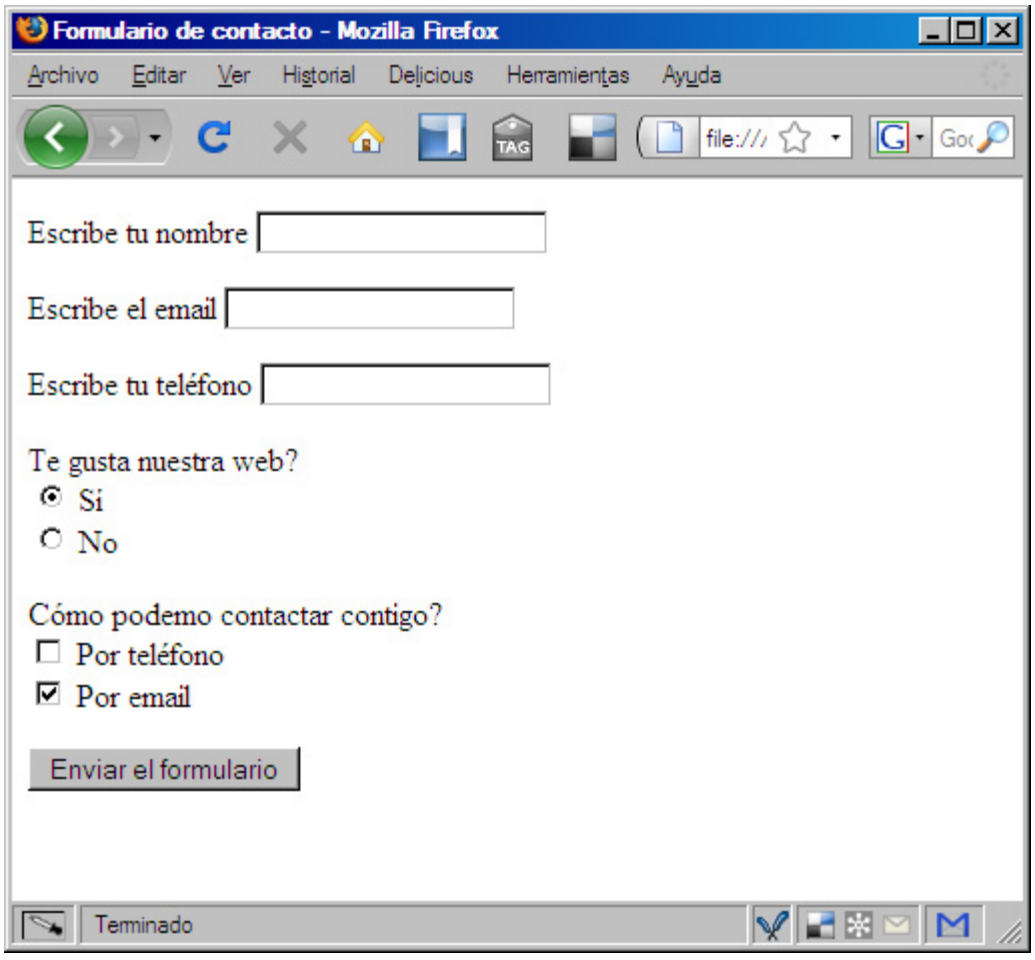
**Submit**

Este es el botón que permitirá enviar automáticamente la información del formulario cuando hagamos clic encima. Para conseguir un botón de este tipo tenemos que dar al atributo "type" el valor de "submit", tal como vemos en el ejemplo.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="es" lang="es">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>formulario contacto</title>
</head>
<body>
<form id="form1" method="post" action="proceso_formulario.php">
<p>
<label>Escribe tu nombre
<input type="text" name="nombre" id="nombre" />
</label>
</p>
<p>
<label>Escribe el e-mail
<input type="text" name="apellido" id="apellido" />
</label>
</p>
<p>
<label>Escribe tu teléfono
<input type="text" name="telefono" id="telefono" />
</label>
</p>
<p>¿Te gusta nuestra web? <br />
<label>
<input type="radio" name="gusta" id="si" value="si" checked="true" />
Sí</label>
<br />
<label>
<input type="radio" name="gusta" id="no" value="no" /> No</label>
</p>
<p>¿Cómo podemos contactar contigo? <br />
<label>
<input type="checkbox" name="contactar" id="contactar_telefono"
value="contactar_telefon"/> Por teléfono
</label>
<br />
<label>
```

```
<input type="checkbox" name="contactar" id="contactar_email"
value="contactar_email" /> Por email
</label>
</p>
<p>
<input type="submit" name="submit" id="submit" value="Enviar el formulario"
/>
</p>
</form>
</body>
</html>
```

Vemos cómo interpreta nuestro código el navegador:




Con el atributo "value", especificaremos el texto que queremos que aparezca dentro del botón de "submit", en nuestro caso "Enviar el formulario".

**Reset**

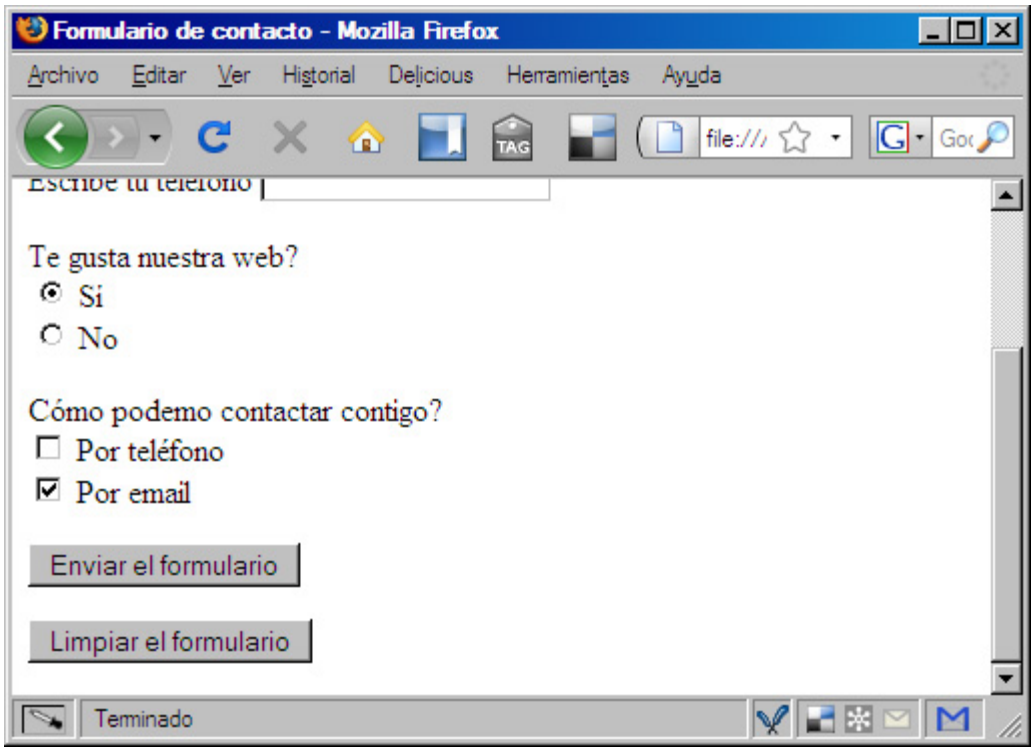
Es posible que el usuario, una vez haya rellenado el formulario, quiera reiniciarlo y volver a escribir la información. Para esta tarea utilizaremos el botón de "reset", que sirve para restablecer los campos del formulario a sus valores por defecto.

Para crear un botón "reset", lo haremos de la forma siguiente:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="es" lang="es">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>formulario contacto</title>
</head>
<body>
<form id="form1" method="post" action="proceso_formulario.php">
<p>
<label>Escribe tu nombre
<input type="text" name="nom" id="nom" />
</label>
</p>
<p>
<label>Escribe el e-mail
<input type="text" name="apellido" id="apellido" />
</label>
</p>
<p>
<label>Escribe tu teléfono
<input type="text" name="telefono" id="telefono" />
</label>
</p>
<p>¿Te gusta nuestra web? <br />
<label>
<input type="radio" name="gusta" id="si" value="si" checked="true" />
Sí</label>
<br />
<label>
<input type="radio" name="gusta" id="no" value="no" /> No</label>
</p>
<p>¿Cómo podemos contactar contigo? <br />
<label>
<input type="checkbox" name="contactar" id="contactar_telefono"
value="contactar_telefono"/> Por teléfono
</label>
<br />
<label>
<input type="checkbox" name="contactar" id="contactar_email"
value="contactar_email" /> Por email
</label>
</p>
```

```
<p>
<input type="submit" name="submit" id="submit" value="Enviar el formulario"
/>
</p>
<p>
<input type="reset" name="limpieza" id="limpieza" value="Limpiar el
formulario" />
</p>
</form>
</body>
</html>
```

Vemos el resultado de nuestro código en el navegador:




### 1.1.6. Listas de opciones o desplegables

Las listas de opciones son otra manera de dar al usuario la opción de escoger una o varias respuestas.

Veremos un ejemplo de ello y lo comentaremos:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="es" lang="es">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>formulario contacto</title>
</head>
```

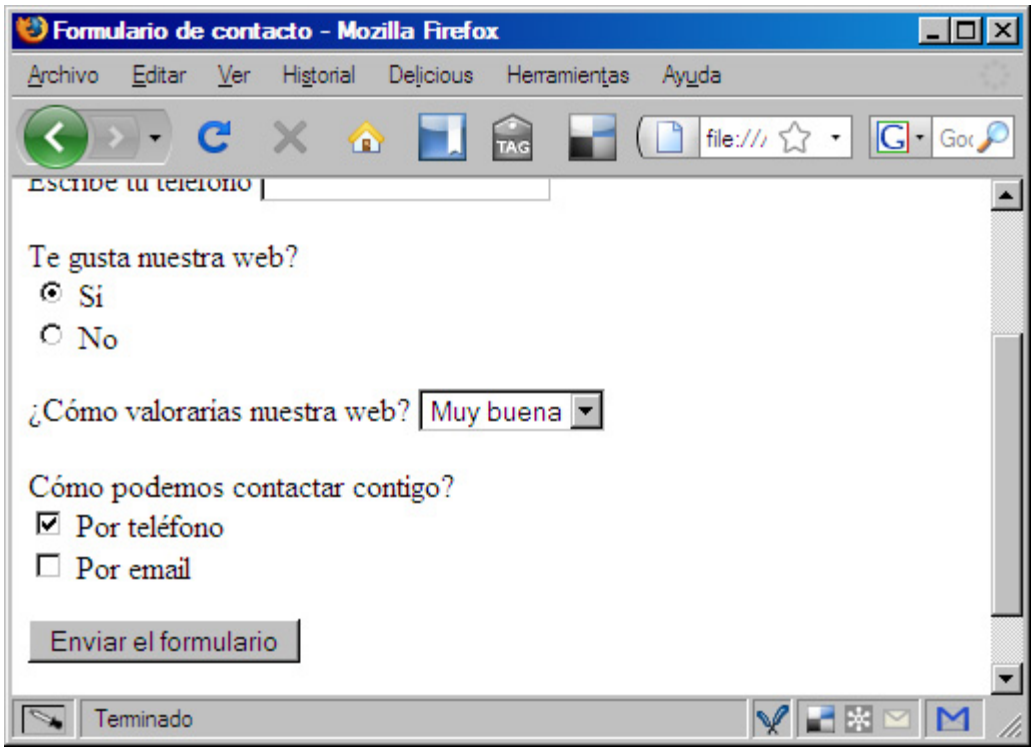
```
<body>
<form id="form1" method="post" action="proceso_formulario.php">
<p>
<label>Escribe tu nombre
<input type="text" name="nombre" id="nombre" />
</label>
</p>
<p>
<label>Escribe el e-mail
<input type="text" name="apellido" id="apellido" />
</label>
</p>
<p>
<label>Escribe tu teléfono
<input type="text" name="telefono" id="telefono" />
</label>
</p>
<p>¿Te gusta nuestra web? <br />
<label>
<input  type="radio"  name="gusta"  id="si"  value="si"  checked="true"  />
Sí</label>
<br />
<label>
<input type="radio" name="gusta" id="no" value="no" /> No</label>
</p>
<p>
¿<label>Cómo valorarías nuestra web?
<Select name="valoracion" id="valoracion">
<option value="10" selected="selected">muy buena</option>
<option value="7">buena</option>
<option value="5" >normal</option>
<option value="3">mala</option>
<option value="1">muy mala</option>
</Select>
</label>
</p>
<p>¿Cómo podemos contactar contigo? <br />
<label>
<input     type="checkbox"     name="contactar"     id="contactar_telefono"
value="contactar_telefono"/> Por teléfono
</label>
<br />
<label>
```

```
<input type="checkbox" name="contactar" id="contactar_email"
value="contactar_email" /> Por email
</label>
</p>
<p>
<input type="submit" name="submit" id="submit" value="Enviar el formulario"
/>
</p>
<p>
<input type="submit" name="limpieza" id="limpieza" value="Limpiar el
formulario" />
</p>
</form>
</body>
</html>
```

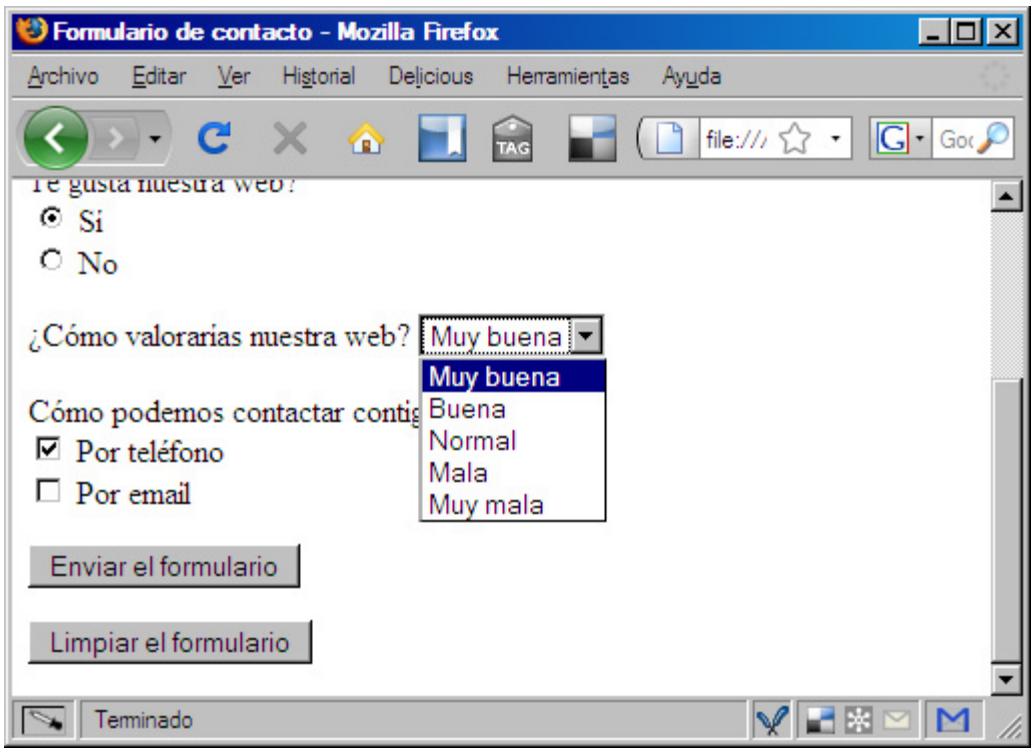
El par de etiquetas <select> definen nuestra lista. Una vez creada, iremos introduciendo las diferentes opciones que el usuario tiene que escoger. Eso lo haremos mediante el elemento <option>. Vemos que cada elemento <option> tiene un atributo "value" diferente, mientras que el elemento <select> tiene un atributo "name". Eso quiere decir que, cuando enviamos la información del formulario, el atributo "name" se asociará con el valor del atributo "value" del elemento que el usuario haya escogido. Es decir, si el usuario escoge la segunda opción, el formulario se enviará con la siguiente información:

[valoracion=7]

Vemos en la imagen siguiente cómo se visualizaría nuestra lista:

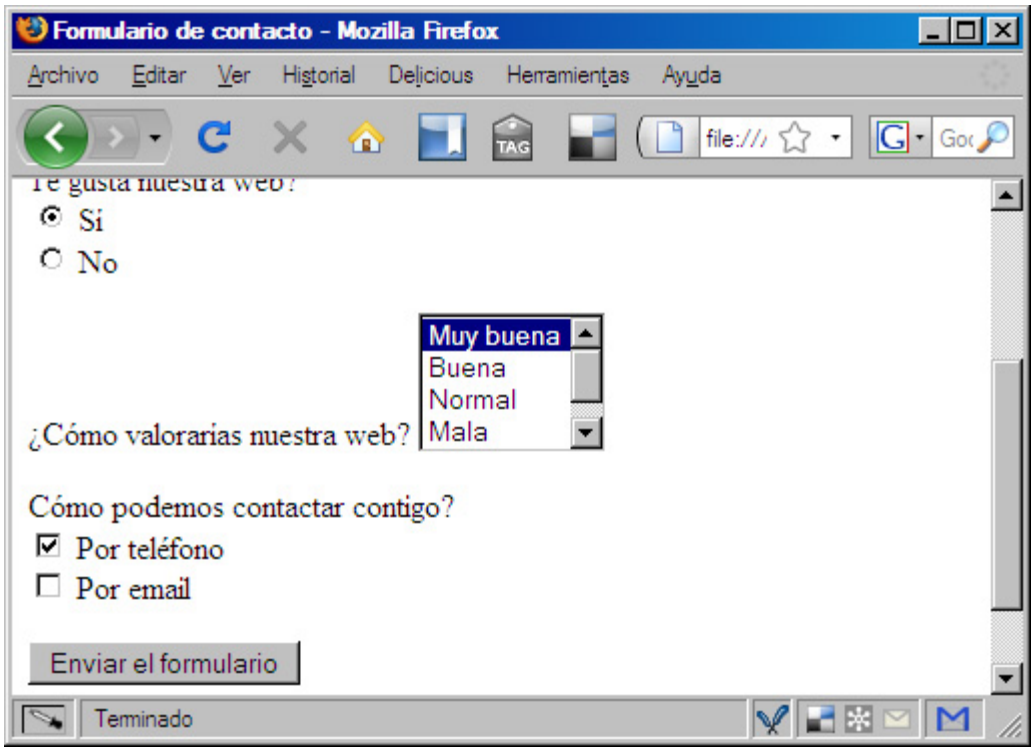


Como podemos ver en la imagen, las listas, por defecto, vienen juntas y una vez hacemos clic encima, se despliegan y muestran todas las opciones que tienen. Veamos en la imagen siguiente el efecto que tiene sobre nuestra lista cuándo hacemos clic encima:




Si queremos que nuestras listas aparezcan siempre desplegadas, añadiremos el atributo "size" al elemento "select". El valor que indicamos en este atributo delimitará la altura que tendrá nuestra lista. Veámoslo en un ejemplo:

```
<label>¿Cómo valorarías nuestra web?
<Select name="valoracion" id="valoracion" size=4>
<option value="10" selected="selected">muy buena</option>
<option value="7">buena</option>
<option value="5" >normal</option>
<option value="3">mala</option>
<option value="1">muy mala</option>
</Select>
</label>
```




Como vemos en la imagen, la altura de nuestra lista equivale a cuatro elementos. Si hay más opciones de las especificadas en el atributo "size", como es el caso de nuestro ejemplo, saldrá un pequeño scroll en la parte derecha que nos permitirá acceder a todas las opciones.

### 1.1.7. Etiquetas (label)

Las etiquetas son los textos que describen cada elemento que forma parte de un formulario. Por lo tanto, siempre irán asociadas a otros elementos. Estos elementos recibirán el nombre de controles.

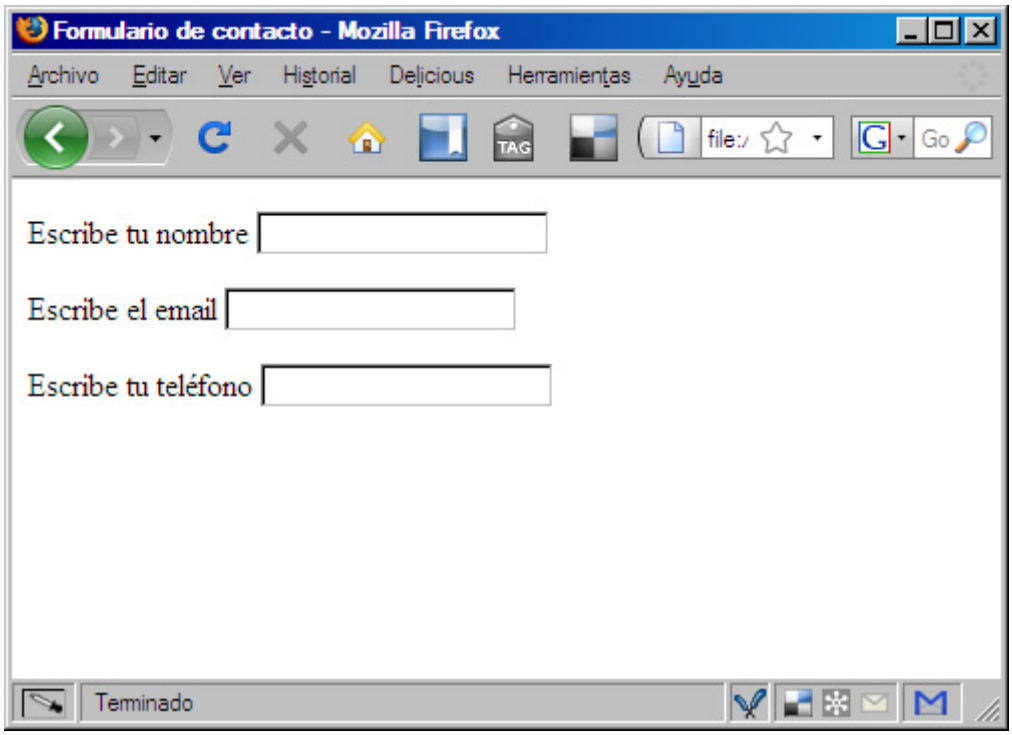
Las etiquetas sirven para ayudar al usuario a entender la función de cada control. Entre otras cosas, describen el contenido de los campos de texto, y dan nombre a los valores de los checkboxes y radiobuttons.

Al principio de este capítulo hemos comentado que utilizaríamos etiquetas para hacer más entendedor nuestro ejemplo. Recordemos el código que vamos a utilizar:

```
<p>
<label>Escribe tu nombre
<input type="text" name="nombre" id="nombre" />
</label>
</p>
<p>
<label>Escribe el e-mail
<input type="text" name="apellido" id="apellido" />
</label>
</p>
<p>
<label>Escribe tu teléfono
<input type="text" name="telefono" id="telefono" />
```

```
</label>
</p>
```

El resultado que entonces obteníamos en el navegador era así:




### 1.1.8. Agrupación de campos (fieldset)

Este elemento nos permite agrupar visualmente una serie de elementos del formulario. Es decir, si en nuestro formulario tuviéramos dos partes que quisiéramos diferenciar de manera visual las podríamos englobar con dos elementos "fieldset".

Por defecto, el impacto visual que genera un fieldset es la aparición de un rectángulo que engloba los elementos dentro del fieldset. Este rectángulo puede llevar un título para hacer más comprensible la diferenciación. Este título lo especificaremos con el elemento "legend", que puede admitir un atributo "align" que nos permitirá indicar la posición del título. Los valores que puede soportar "align" son "left", "center", "right, top" y "bottom".

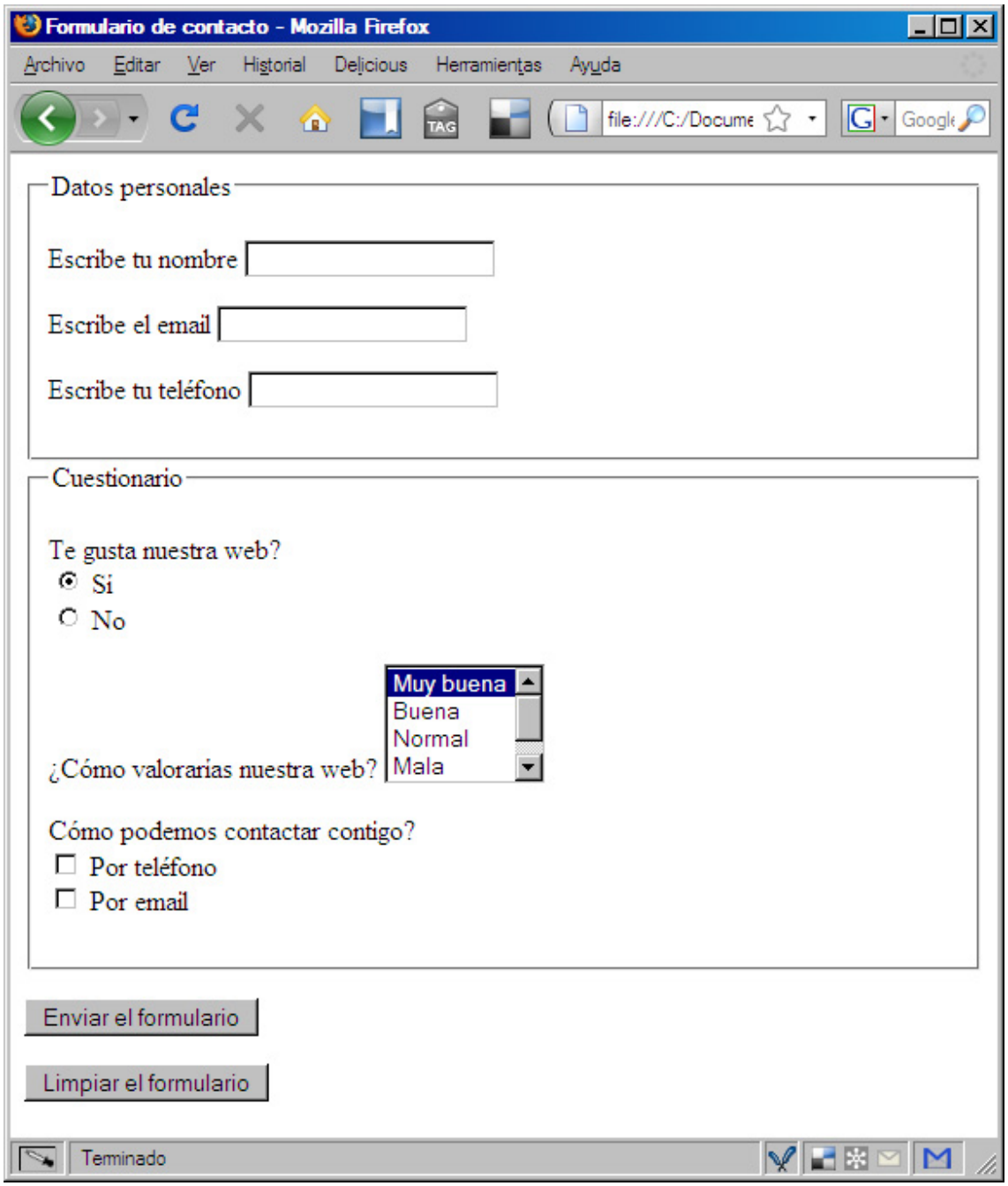
Continuando con nuestro ejemplo, agruparemos el formulario que hemos creado con dos partes. Una para la parte donde pedimos los datos y otra para la parte del pequeño cuestionario:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="es" lang="es">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>formulario contacto</title>
</head>
<body>
<form id="form1" method="post" action="proceso_formulario.php">
<fieldset>
```

```
<legend>Datos personales</legend>
<p>
<label>Escribe tu nombre
<input type="text" name="nombre" id="nombre" />
</label>
</p>
<p>
<label>Escribe el e-mail
<input type="text" name="apellido" id="apellido" />
</label>
</p>
<p>
<label>Escribe tu teléfono
<input type="text" name="telefono" id="telefono" />
</label>
</p>
</fieldset>
<fieldset>
<legend>Cuestionario</legend>
<p>¿Te gusta nuestra web? <br />
<label>
<input  type="radio"  name="gusta"  id="si"  value="si"  checked="true"  />
Sí</label>
<br />
<label>
<input type="radio" name="gusta" id="no" value="no" /> No</label>
</p>
<p>
¿<label>Cómo valorarías nuestra web?
<Select name="valoracion" id="valoracion">
<option value="10" selected="selected">muy buena</option>
<option value="7">buena</option>
<option value="5" >normal</option>
<option value="3">mala</option>
<option value="1">muy mala</option>
</Select>
</label>
</p>
<p>¿Cómo podemos contactar contigo? <br />
<label>
<input     type="checkbox"     name="contactar"     id="contactar_telefono"
value="contactar_telefono"/> Por teléfono
</label>
<br />
```

```
<label>
<input type="checkbox" name="contactar" id="contactar_email" value="contactar_email" /> Por email
</label>
</p>
</fieldset>
<p>
<input type="submit" name="submit" id="submit" value="Enviar el formulario" />
</p>
<p>
<input type="submit" name="limpieza" id="limpieza" value="Limpiar el formulario" />
</p>
</form>
</body>
</html>
```

Vemos el resultado de este código:




## 1.2. Los métodos POST y GET

Al principio del capítulo, hemos dicho que los datos del formulario se envían al servidor mediante el método indicado al atributo "method" del elemento "form". Pues bien, los dos métodos posibles son "GET" y "POST".

La diferencia entre estos dos métodos radica en la forma de enviar los datos del formulario.

El **método "GET"** envía los datos mediante la URL.

**Ejemplo**

Por ejemplo, si enviamos un formulario con este método veríamos en la barra de dirección una URL de este estilo:

http://url?parametre1=valor1&parametre2=valor2

Este método tiene como inconveniente la limitación de la URL. Es decir, si tenemos un formulario con bastantes campos y en el que el usuario puede introducir grandes cantidades de texto, es muy probable que sobrepasemos el límite de 256 caracteres que, como máximo, puede soportar la URL.

Una ventaja de este método es que podemos construir enlaces con los valores ya introducidos. Este sistema resulta bastante útil cuando necesitamos comprobar el funcionamiento del formulario y queremos evitar que se tenga que llenar continuamente.

Con el **método "POST"** los datos son enviados de manera invisible por el usuario, ya que la información del formulario se almacena en una parte del servidor. Además, si utilizamos el método "TABLA" no tendremos ninguna restricción con respecto a la cantidad de información que podemos enviar.

## 2. Objetos

Actualmente, además de texto e imágenes, en la web se encuentran a menudo vídeos, sonidos o elementos interactivos. Estos contenidos, independientemente del formato, se identifican como objetos. La etiqueta que los representa es, por lo tanto, "object".

### 2.1. Propiedades y definición

Los objetos dependerán siempre de un conector (plug-in) para ser reproducidos correctamente en el navegador.

**Ejemplo**

Un ejemplo generalizado del uso de conectores es el reproductor de archivos .swf, más conocido por "Flash Player" o reproductor de archivos flash. El hecho de no tenerlo instalado en el navegador hará imposible la visualización de archivos flash.

El W3C recomienda el uso del elemento object para incluir estos objetos, declarando, dentro del elemento object, de qué tipo se trata por medio del atributo type, que indicará al navegador cuál es el tipo de conector que tiene que utilizar, su ubicación con el atributo data, así como los parámetros del objeto con los valores respectivos.

### 2.2. Ejemplos comunes

Los ejemplos más comunes a la hora de incluir objetos son para visualizar películas flash interactivas.

Para incorporar una de estas películas en nuestra web, tendríamos que especificar el código siguiente:

```
<object type="application/x-shockwave-flash" data="prova.swf" width="550" height="450">
<param name="quality" value="high" />
<p>No tienes el conector de Flash instalado en el navegador</p>
</object>
```

Como vemos, tendremos que tener un archivo "prova.swf" en nuestro disco duro, con el fin de poder visualizarlo. Si no tenemos el plugin de Flash instalado en nuestro navegador, se visualizará la parte dentro del elemento object; en nuestro caso, se visualizaría el párrafo siguiente:

```
<p>No tienes el conector de Flash instalado en el navegador</p>
```

# La publicación

Para publicar nuestra web en Internet, necesitaremos tres **elementos básicos**:

**1)** contratar un dominio,

**2)** contratar un *hosting* donde almacenar nuestra web,

**3)** y, por último, colgar nuestros archivos en este *hosting* mediante el protocolo de transferencia de archivos FTP.

## 1. El dominio

Técnicamente, un dominio es una dirección mnemotécnica o alias de Internet. Esto puede sonar muy complicado, pero veremos que no lo es. Cada página web del mundo está almacenada en un ordenador que llamaremos servidor.

Cada ordenador o servidor, en Internet, tiene asignado un identificador numérico único. A esto se le llama **dirección IP**. La dirección IP es un conjunto de 4 series de números.

**Ejemplo**

Por ejemplo: 214.23.87.125.

Recordar un conjunto de series de 4 números es bastante complejo, por lo que se decidió que una dirección podría estar asociada a un nombre alfanumérico.

**Ejemplo**

Por ejemplo: **www.google.com**

Es decir, que cuando escribimos en el navegador la dirección **www.google.com,** por decirlo de alguna manera, nuestra conexión a Internet (los servidores dns) convierte este nombre en una serie de 4 números para conseguir conectar con el servidor.

Hay **dos tipos de dominios**:

**1)** Los **genéricos** son los dominios básicos en Internet y los más utilizados por todo el mundo. Están organizados de forma conceptual según sus terminaciones: .com, .net, .org, .gov, .biz, etc.

**2)** Los **territoriales** son los dominios mantenidos en cada país y son utilizados por las organizaciones y empresas que desean establecerse en Internet y proteger la identidad de su marca o su nombre comercial en un país concreto.

**Empresas gestoras de registro de dominios**

Podemos encontrar diferentes empresas dedicadas al registro de dominios, pero al escoger una tenemos que asegurarnos que sea una empresa fiable, que ofrezca un precio razonable y las mínimas posibilidades exigibles al servicio del dominio: gestión de DNS, redirección automática y gratuita, etc.

**Valor económico del dominio**

En cuanto al valor económico de los dominios, no existe un baremo oficial, ya que cada compañía es, en definitiva, quien establece los precios a su voluntad en una economía de libre mercado. Además, una empresa o institución siempre estará interesada en utilizar un nombre de dominio que coincida con su nombre comercial o marca. Esta particularidad hace que ciertos nombres de dominio sean susceptibles de tener un valor económico más alto, ya que pueden tener una alta atracción comercial para los usuarios, como **www.cocacola.com**, **www.nike.com**, etc.

## 2. Alojamiento web

El alojamiento web, también llamado web *hosting*, es el servicio de almacenaje, acceso y mantenimiento de los archivos que integran una web.

Es un servicio que posibilita alojar nuestra web en un servidor para que cualquier persona del mundo pueda visitarla cuando lo desee.

Los planes de web *hosting*, varían en cuanto al espacio en disco que ofrecen, al sistema operativo que utilizan para mostrar la web, y a los conjuntos de servicios y herramientas que ofrecen conjuntamente.

Las empresas que se dedican a ello ofrecen diferentes tipos de web *hosting*, entendiendo cada tipo como un conjunto de servicios y características que definen el *hosting*. Estas **características** pueden ser:

- Cantidad de espacio web para alojar los archivos que forman nuestra web.
- Transferencia máxima mensual. Esta cantidad es un máximo que está compuesto por la transferencia de archivos al servidor, así como del tráfico de archivos generados por los visitantes en la web.
- Número de cuentas de correo asociadas al dominio. Según qué opción se haya contratado, las cuentas de correo que podemos disfrutar oscilarán entre los 10 y los 100 para un paquete normal.
- Posibilidad de registrar un dominio de primer nivel (.com, .net, .org...)
- Número de bases de datos.
- Lenguaje de programación soportado: PHP (en servidores Linux), ASP (en servidores Windows)...

A continuación, explicaremos los diferentes **tipos de web *hosting*** que existen:

- ***Hosting gratuito***: es extremadamente limitado comparado con el *hosting* de pago. Los proveedores de este tipo de alojamiento normalmente requieren poner sus propios anuncios en la web y tienen límites muy grandes en cuanto a espacio en el disco y tráfico de archivos.
- ***Hosting compartido***: funciona cuando en un mismo servidor se alojan distintas webs de clientes diferentes. El *hosting* compartido también tiene algunas restricciones con respecto a ciertas funcionalidades (no puedes instalar programas directamente en el servidor), aunque se trata de un servicio económico con un muy buen rendimiento.
- **Servidores virtuales**: estos tipos de alojamiento funcionan mediante el uso de una máquina virtual dentro de un servidor. Esto quiere decir que la empresa proveedora ofrece el control de un ordenador aparentemente no compartido.
- ***Hosting dedicado***: con este tipo de alojamiento dedicado se alquila un servidor entero para un solo cliente. Además, no tienen ninguna restricción y se tiene el control total y, eso sí, la responsabilidad de administrar el servidor.

**Lo primero que tenemos que tener en cuenta a la hora de escoger el tipo de web *hosting* es determinar qué requisitos técnicos pedirá el funcionamiento de nuestra web: si necesitará programación en el servidor o utilización de bases de datos, qué requerimientos de transferencia mensual necesitaremos y el espacio en disco necesario para todo el conjunto.**

También tenemos la opción de escoger una web *hosting* que cuente con un panel de control para su gestión. La existencia de esta herramienta nos permite crear y modificar nuestras cuentas de correo, controlar la transferencia de archivos mensual consumida, además de dar acceso a estadísticas, crear subdominios, configurar cuentas de FTP, etc.

También hay otros factores para realizar esta elección, como la capacidad de respuesta del servicio técnico ante una avería de sistema, que la empresa contratada asegure que realiza copias de seguridad diarias para evitar posibles pérdidas de información, etc.

**Empresas de web hosting**

Algunas de las empresas más conocidas que ofrecen estos servicios son:

• En España: Arsys, Acens, Cdmon, Nexica, Nominalia.

• En el extranjero: Dream-host, Mediatemple, OVH.

## 3. FTP

FTP es uno de los protocolos de Internet. Aunque es muy antiguo, es ideal para transferir datos por la red. La mayoría de las páginas web a escala mundial sube sus archivos a Internet mediante este protocolo. De hecho, será la herramienta que utilizaremos para subir nuestros archivos al servidor que hayamos contratado, para que nuestra web pueda ser visible en Internet.

Para poder enviar archivos mediante este protocolo, hace falta un servidor de FTP y un cliente FTP. Tener un servidor FTP quiere decir que tengamos contratado este servicio por parte de nuestra web *hosting*, tal como es habitual, y tener un cliente FTP quiere decir que disponemos de una herramienta o programa para poder transferir nuestros archivos a Internet. Generalmente, todos los sistemas operativos llevan una herramienta de cliente FTP.

## 3.1. Cliente FTP

Un cliente FTP utiliza el protocolo FTP para conectarse a un servidor FTP para poder transferir archivos de cualquier tipo.

Algunos clientes de FTP básicos vienen integrados en los sistemas operativos, en los cuales se incluye Windows, DOS, Linux y Unix. No obstante, hay disponibles clientes FTP con más funcionalidades, habitualmente en forma de software libre para Windows y Unix. Muchos navegadores actuales también llevan integrados clientes FTP.

**Clientes FTP**

A causa de la gran necesidad que existe de ellos, hay muchos clientes FTP. Para mencionar sólo unos cuantos: FileZila, CuteFTP, WSS FTP, Coffe Lagar, CoreFTP, WorldWide FTP, FTP Now, Shuttle FTP Suite, etc.

Algunos son de pago, otros son software libre.

**Recomendamos, para seguir los ejemplos que comentaremos, descargar el FileZilla por varios motivos:**

**Está en diferentes idiomas, cosa que no todos los clientes FTP tienen.**

**Se conecta de manera rápida.**

**Permite conexiones seguras.**

**Es gratuito y tiene una comunidad detrás muy activa que saca nuevas versiones a menudo.**

### 3.1.1. Cómo subir archivos mediante el cliente FTP

Primero, tendremos que tener un servidor donde alojar nuestra web. Como hemos visto antes, hay de pago y gratuitos. Para empezar y poder seguir los ejemplos que veremos, tenemos que tomar un servidor gratuito.

Un vez hemos escogido el servidor gratuito, tendremos que crear una cuenta. Para ello, iremos siguiendo las indicaciones y, tras finalizar el proceso de registro, nos darán unos datos para configurar el cliente de FTP. Los datos más importantes para configurar el cliente FTP son los siguientes:

- **Host:** es la dirección FTP para poder hacer la conexión. Es una dirección del tipo ftp.dominio.com, o ftp.usuario.dominio.com.
- **Usuario:** es el nombre de usuario que habéis escogido a la hora de registraros.
- Y la **contraseña**.

**Servidores gratuitos**

Hay muchos, como:

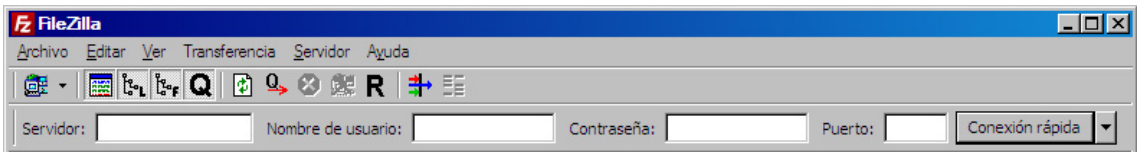
• **www.tripod.lycos.es/**

• **http://www.vivelared.com/**

• **http://members.freewebs.com/**

### Configuración de nuestro cliente de FTP

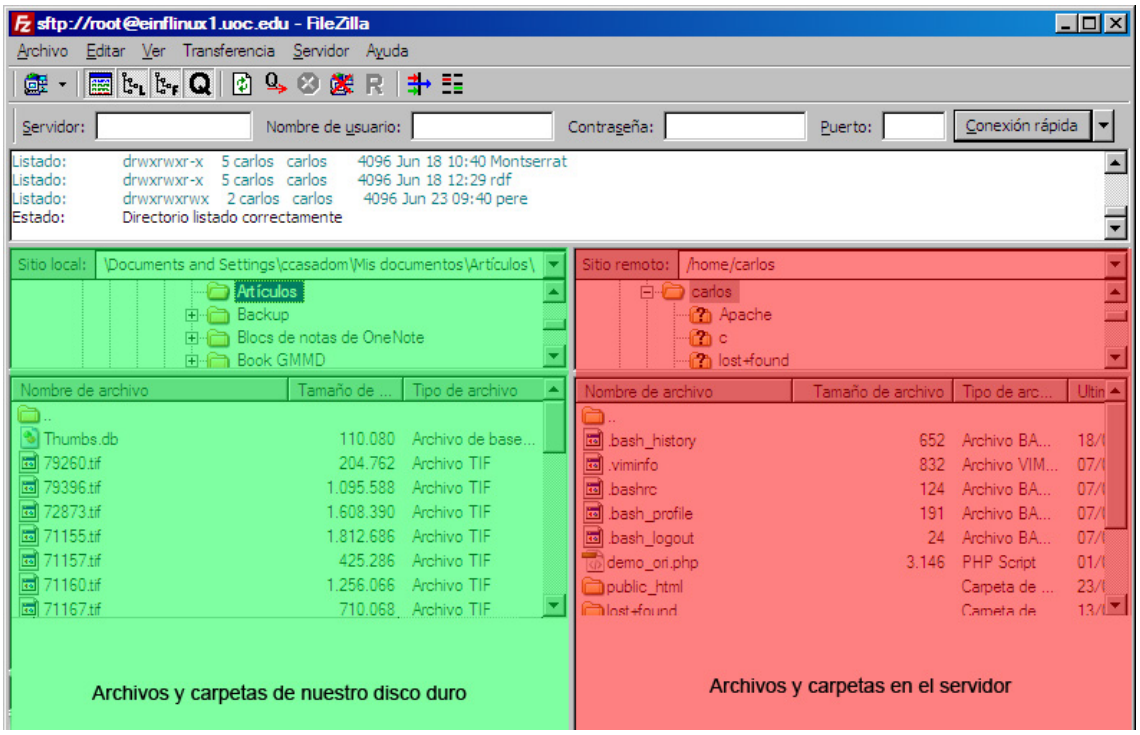
Abriremos nuestro cliente FTP FileZilla y rellenaremos los campos siguientes:




En la parte "Anfitrión", introduciremos los datos del host. En "Nombre de usuario" y "contraseña" introducimos los datos personales. En "Port" tenemos que poner lo que nos ha proporcionado el servidor FTP: en principio, el puerto por defecto es el 21.

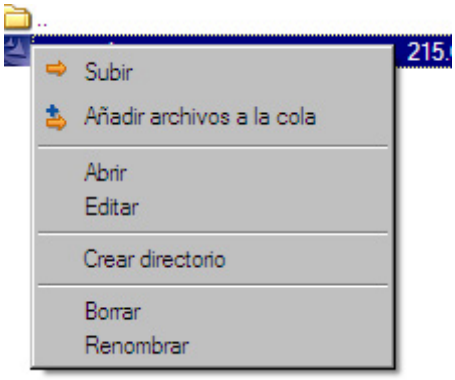
Una vez introducidos todos los datos, hacemos clic en el botón "Conexión rápida", con el fin de conectarnos a nuestro servidor. Cuando este proceso haya finalizado, nos tiene que salir un mensaje del tipo "Listado de los directorios con éxito".

Veremos que, en la parte izquierda del programa, tenemos los archivos y carpetas de nuestro ordenador, y en la parte derecha se mostrarán los archivos que están alojados en el servidor. En la imagen siguiente, se visualiza claramente esta disposición:



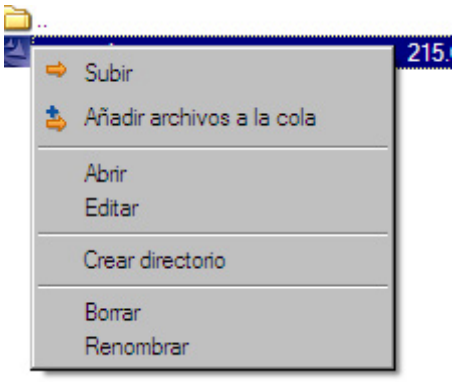

### Subir un archivo

Para subir un archivo o carpeta de nuestro ordenador al servidor web, tenemos que hacer clic con el botón derecho del ratón sobre el archivo o carpeta que nos interese subir y, de las opciones que nos salen, optar por "Cuelga".




### Descargar un archivo

Para descargar archivos desde el servidor web en nuestro ordenador, tendremos que ir a la parte derecha y escoger el archivo que queramos descargar, hacer clic con el botón derecho del ratón y escoger la opción "Baja":




Estas que hemos visto son las opciones básicas de un cliente FTP; con ellas, podremos subir los archivos de nuestra web a cualquier servidor FTP.

En principio, pues, ya conocemos la base. Sabemos cómo hacer nuestra web básica y cómo publicarla en Internet. Sólo queda poner imaginación y muchas ganas para seguir haciendo grande este mundo que llamamos Internet.