HTML

Miguel Ortuño
Escuela de Ingeniería de Fuenlabrada
Universidad Rey Juan Carlos

Febrero de 2025

Introducción a HTML (1)

- HTML *Hypertext Markup Language* es un lenguaje de marcado, inicialmente se usa para que navegadores web compongan páginas con diverso contenido: texto, enlaces, imágenes, audio, vídeo, etc. Páginas *estáticas*: se editaban generalmente a mano
- El éxito de internet y HTML como interfaz de usuario hace que a finales de los años 1990 aparezcan tecnologías que permiten páginas web dinámicas: CGI, PHP, ASP...
- En 1995, JavaScript permite programar scripts en el navegador web, típicamente para componer código HTML
- Muy importante: no confundir HTML con HTTP

Introducción a HTML (2)

- A mediados de los años 2000 se desarrolla una nueva generación de herramientas para facilitar la programación de aplicaciones web: Ruby on Rails, Django...
- En esta misma época, año 2005, la aparición de AJAX permite que las aplicaciones web comiencen a semejarse a las aplicaciones de escritorio
- A principios de la década de 2010, la madurez de las tecnologías web hace que empiecen a usarse también fuera del navegador. *JavaScript everywhere*. Nodejs. HTML como plataforma para interfaz gráfico en el escritorio

Lenguajes de marcado

HTML es un *lenguaje de marcado*: un sistema que permite incluir metainformación en un documento, esto es, información sobre la información

- La metainformación tiene que distinguirse sintácticamente del texto.
- Es la evolución del *lapiz azul* con el que tradicionalmente se editaban documentos cuando la tecnología era analógica
  - Ejemplo de lenguaje de marcado muy elemental: redacto un documento en un procesador de textos, lo imprimo y alguien lo revisa, incluyendo anotaciones a mano. Las anotaciones (metainformación) se distingue fácilmente del texto original Para hacer algo semejante de forma digital, es necesaria una sintaxis que separe el texto de la metainformación
- Ejemplos de lenguajes de marcado: troff, LaTeX, JsonML, SGML, HTML, XML

XML

- XML *Extensible Markup Language*
  Es una forma de describir datos jerárquicamente. Estándar para transferir información entre distintos sistemas sin tener que adaptarlos a cada plataforma concreta, y de forma que sea fácil de leer por un humano y fácil de procesar por un ordenador
- Creado en 1996 por el W3C (*World Wide Web Consortium*)
  Dos versiones: XML 1.0 y XML 1.1
- Algunos autores lo consideran un lenguaje de marcado, otros, un metalenguaje de marcado
- Proviene de SGML, *Standard Generalized Markup Language*, norma ISO 8879:1986
  SGML es un metalenguaje, un lenguaje para definir lenguajes de marcado,

- XML tiene una sintaxis muy similar a la de HTML porque ambos provienen de SGML
- XML y HTML no son lenguajes alternativos
  - XML está diseñado para describir y comunicar datos datos de máquina a máquina
  - HTML está diseñado para presentar en pantalla datos con formato. De máquina a persona

En HTML, como en cualquier lenguaje de marcado, es esencial separar los aspectos semánticos del texto del formato de la representación gráfica

- Con la aparición de las plataformas móviles (smartphones y tablets), esto se vuelve aún más importante
- Las primeras versiones de HTML no eran muy rigurosas en esto, pero se ha ido corrigiendo gradualmente en cada nueva especificación

Internet moderno: HTML

- HTML es la base de la World Wide Web, y por tanto, de la internet moderna
- Para los usuarios, WWW e internet son sinónimos. Pero nosotros debemos distinguirlo

Internet antes del web:

- Los predecesores de internet aparecen en los años 1960
- TCP/IP se desarrolla durante la década de 1970
  - 1 de enero de 1983: *flag day* en que la red ARPANET migra desde NCP hasta TCP/IP
- En la internet primitiva se usaban servicios como el correo electrónico, ftp, telnet, usenet, gopher, irc y algunos otros

Versiones de HTML (1)

- Años 1989-1991. Tim Berners-Lee, un físico del CERN (*European Organization for Nuclear Research, Conseil Européen pour la Recherche Nucléaire*) crea un sistema de hipertexto para internet al que llama WWW, *World Wide Web*.
  - Para ello desarrolla el lenguaje HTML junto con el protocolo HTTP
  - Lo que en origen era un servicio más para un ámbito muy específico, tiene un éxito arrollador que cambia no solo internet y la informática, sino las comunicaciones humanas
  - El resto de servicios de internet siguen diferenciándose del WWW, pero casi todos ellos acaban teniendo un interfaz de usuario web, lo que hace que el usuario lo perciba como la misma cosa

Versiones de HTML (2)

- Año 1995. HTML 2.0. Publicado por el IETF (Internet Engineering Task Force). Añade formularios, tablas, y soporte para internacionalización, entre otros
- Año 1997. HTML 3.2. *Guerra de los navegadores*. Microsoft Internet Explorer y Netscape Navigator intentaban prevalecer en el mercado, añadiendo características propias. HTML 3.2 busca que ambos navegadores vuelvan a ser compatibles. Añade características muy desacertadas, como la etiqueta *font* y el atributo *color*

Versiones de HTML (3)

- Versión 4.0. Año 1997
  Normaliza el uso de marcos *frames*, disponible desde Netscape Navigator 2 (año 1995)
  Los marcos eran documentos HTML dentro de documentos HTML. Concepto problemático, han ido desapareciendo
  Tres variantes de HTML 4.0
  - *Strict*, donde se prohiben elementos obsoletos
  - *Transitional*, que admite elementos obsoletos
  - *Frameset*, solo marcos
- Versión 4.1. Año 2000
  La versión más usada, hasta la aparición de HTML 5

Versiones de HTML (4)

- Año 2004. El W3C decide abandonar HTML y migrar a XHTML
  - XHTML: Extensible Hypertext Markup Language. Lenguaje muy similar a HTML, pero que sigue estrictamente la sintaxis de XML

  El desarrollo de HTML lo retoma el WHATWG (Web Hypertext Application Technology Working Group: Google Apple, Mozilla, Opera)
- Año 2008. Primer borrador de HTML 5 publicado por WHATWG
- Año 2009. El W3C abandona XHTML y vuelve a HTML5

Versiones de HTML (5)

- Año 2014. HTML 5.0
  - Define de forma precisa qué hacer con páginas incorrectas
  - Muchas mejoras en interoperabilidad
  - Mucho mejor soporte para dispositivos móviles
  - Audio y video
  - Gráficos vectoriales
  - Muchas APIs nuevas, como la geolocalización
- Año 2017. HTML 5.2

En la actualidad cualquier desarrollo debería centrarse en HTML 5

- La compatibilidad con HTML 4 es bastante buena
- Existen técnicas y herramientas que permiten que páginas HTML 5 se muestren correctamente en navegadores antiguos

Toda la información contenida en las transparencias de esta asignatura es válida en HTML 4 y HTML 5, salvo indicación contraria

Adobe Flash

Otra de las grandes ventajas de HTML 5 es que permite prescindir de Flash

- Adobe Flash era una plataforma software desarrollada por Adobe Systems para mostrar animaciones, gráficos vectoriales, vídeos, audio, contenido interactivo...
- Fue muy popular entre los años 2000 y 2010
- Muy problemático. Ya en el año 2000 se publican artículos como *Flash: 99 % Bad*, (J.Nielsen)
  No estándar. Dependencia del fabricante. Anima a desarrollar contenido centrado en la apariencia gráfica externa, no en la usabilidad y la semántica
- No soportado por Apple en el iPhone

Desaparece por completo de forma oficial en 2020

Sintaxis HTML: composición de un elemento

Un documento HTML está compuesto por elementos (*elements*)
Un elemento puede ir

- A continuación de otro elemento
- Dentro de otro elemento. Esto es muy habitual, los documentos típicos tienen muchos niveles anidados

La mayoría de los elementos están formados por:

- Etiqueta de apertura (*start tag*)
- Contenido
- Etiqueta de cierre (*end tag*)

Ejemplo:

```
<h1>Este elemento es un título de nivel 1</h1>
```

Una etiqueta de apertura sencilla está formada por:

- Signo de menor
- Nombre de la etiqueta
- Signo de mayor

Ejemplo:

```
<h1>
```

Una etiqueta de cierre está formada por:

- Signo de menor
- Barra (*slash*)
- Nombre de la etiqueta
- Signo de mayor

Ejemplo:

```
</h1>
```

Lo habitual y recomendable es no poner ningún espacio ni después del < ni antes del >

- Un espacio después del < es un error
  Ejemplo :
  ```
  < h1>
  ```
  Esto es un ERROR
- Un espacio antes del > es legal, pero no es recomendable
  Ejemplo :
  ```
  <h1 >
  ```

- En la mayoría de elementos la etiqueta de cierre es obligatoria Ejemplo:

```
<pre>Texto preformateado, con fuente de ancho fijo. Se mantienen
los saltos de línea y los espacios    consecutivos</pre>
```

- En algunos elementos se puede omitir la etiqueta de cierre. Aunque no es recomendable. Ejemplo :

```
<p>Esto es un párrafo correcto</p>
<p>Esto también es un párrafo correcto
```

- En algunos elementos, los de tipo *void*, no puede haber ni contenido ni etiqueta de cierre. Solo pueden tener, opcionalmente, atributos. Ejemplo :

```
<br></br>
```

Esto es un ERROR

Elementos de tipo void muy habituales son: *br*, *hr*, *meta*, *link*, *img*, *input*

- En HTML 4.01 también son de tipo void: *area*, *base*, *col*, *param*
- HTML 5 añade: *source*

Elementos que es recomendable, pero no obligatorio, cerrar:

- Párrafos:

```
<p>
```

- Elementos de alto nivel:

```
<html>, <head>, <body>
```

- Viñetas:

```
<li>, <dt>, <dd>
```

- Tablas:

```
<td>, <th>, <tr>, <tbody>, <tfoot>
```

- Algunos más

¿Etiqueta h1 o elemento h1?

Como hemos visto, HTML está formado por elementos, que siempre tienen etiqueta de apertura y que pueden tener contenido y etiqueta de cierre
Consideremos por ejemplo

```
<h1>Introducción</h1>
```

- En rigor deberíamos decir *el elemento h1*, no *la etiqueta h1*
- Pero en muchos contextos es habitual y por tanto aceptable hablar de *la etiqueta h1*, se entiende que es una sinécdoque, nos estamos refiriendo a *el elemento que empieza por la etiqueta h1*

Sintaxis de HTML: distribución de los elementos

Un documento HTML está formado por

- Declaración de tipo
- Un elemento *html*, que contiene
  - Un elemento *head*, que contiene
    - Título
    - Codificación de caracteres
  - Un elemento *body*

```
<!DOCTYPE html>
<html>
  <head>
    <title>Hola mundo en HTML</title>
    <meta charset="utf-8">
  </head>
  <body>
    Hola, mundo.
  </body>
</html>
```

- Algunas etiquetas de estos elementos se pueden omitir en ciertas circunstancias y el documento sigue siendo válido, quedan sobreentendidas (head, body, html). Pero siempre es preferible ponerlo todo
- Otros elementos como el título y la codificación de caracteres son obligatorios. Si se omiten, herramientas como `https://validator.w3.org` nos indicarán que el documento es erróneo
  - A pesar de eso, los navegadores podrán mostrar el documento de forma satisfactoria, con lo que es muy habitual que nadie se preocupe de corregir estos errores
  - El problema se agrava cuando distintos navegadores tratan los errores de forma distinta

Validación del código

En esta asignatura haremos énfasis en generar siempre código *correcto*, tomando como referencia el *W3C Markup Validation Service*

- Tu código no puede dar ningún error
- Aceptaremos algunos *warnings*, otros será preferible evitarlos

Pregunta típica: *Mi página se ve bien ¿qué más da que tenga errores?*

- Respuesta: no has encontrado ningún error en los navegadores en los que has probado. Pero pueden darse en otros navegadores, en otras plataformas, en versiones del pasado y en versiones del futuro

Cuidado:

- No confundas el *W3C Markup Validation Service* con el *Nu Html Checker* del W3C. De lo contrario, tomarías como erróneas páginas correctas en XHTML o lenguajes similares

DOCTYPE

La declaración `<!DOCTYPE html>` es obligatoria al comienzo de un documento HTML 5

- En rigor no es parte de HTML (no es un elemento HTML) sino una instrucción que le dice al navegador que lo que viene a continuación es un documento HTML, versión 5

En HTML 4.x y anteriores, esto era más complicado
Ejemplos:

- ```
  <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
  "http://www.w3.org/TR/html4/loose.dtd">
  ```

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

Case insensitive

HTML es insensible a mayúsculas (*case insensitive*), aunque lo habitual es usar siempre las minúsculas.
Hay una excepción:

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

Ambas formas son idénticas y correctas, pero lo habitual es usar la primera, posiblemente por influencia de XHTML (que es sensible a mayúsculas, y donde la única forma válida es la primera)

Comentarios

Los comentarios son iguales que en XML. Se pueden poner en cualquier lugar del documento

```
<!-- Esto es un comentario -->
```

Etiquetas autocerradas

En XML, cuando un elemento no tiene texto, hay dos alternativas posibles

- Usar una etiqueta de cierre y otra de apertura

  ```
  <holamundo></holamundo>
  ```

- Usar una etiqueta *autocerrada*

  ```
  <holamundo/>
  ```

  Signo de menor, nombre, barra, signo de mayor

Por influencia de XML, algunos desarrolladores o herramientas de HTML usan las etiquetas autocerradas

- En los elementos void son válidas. Pero no aportan nada, no tienen significado especial
- En otros elementos son incorrectas. Aunque el navegador suele ignorarlos y mostrar la página igualmente

Conclusión: no debemos usar etiquetas autocerradas

Atributos

Dentro de la etiqueta de apertura puede haber uno o más *atributos*, que son modificadores del elemento
Ejemplo:

```
<html lang="es-ES">
<!--   etc etc -->
Esto es texto en español de España
<!--   etc etc -->
</html>
```

- El atributo *lang* indica el idioma del texto, especificado en ISO 639-1
- Su sintaxis es similar pero no idéntica a la variable LANG de Unix, donde se indicaría `es_ES.UTF-8`

- Un atributo es un par formado por un nombre y un valor. Su sintaxis es
  - Nombre del atributo
  - Signo igual
  - Valor del atributo
    - Siempre es de tipo texto
    - Es recomendable meterlo siempre entre comilla dobles, aunque si el atributo no contiene espacios, se pueden omitir
- Si hay varios atributos, se separan por espacios

```
<img src="urjc.gif" alt="Logo de la URJC">
```

- El nombre del atributo no se puede repetir dentro del mismo elemento. Sí puede aparece el mismo nombre de atributo en un elemento distinto
- Los atributos no están ordenados, no hay garantía de que se mantenga el orden

Elemento head

El elemento head es la cabecera del documento.
Es un contenedor para metadatos del documento HTML. Esta información nunca se muestra directamente. Sus elementos son

`<title>`, `<base>`, `<style>`, `<link>`, `<meta>`, `<script>`

- El elemento `<title>` define el título del documento. Es de inclusión obligatoria
- El elemento `<base>` define la base de las direcciones relativas
- Los elemento `<style>` y `<link>` especifican las hojas de estilo CSS
- El elemento `<meta>` se usa para añadir diversa metainformación
- El elemento `<script>` contiene código JavaScript, o un enlace a una página con el código

CSS

CSS (*Cascading Style Sheets*) es un lenguaje de diseño gráfico para crear hojas de estilo, que son son una sucesión de reglas que especifican el formato gráfico de un documento
Las hojas CSS pueden ubicarse

- En el propio documento HTML, dentro del elemento `<style>`

```
<style>
  p {
    background-color: salmon;
  }
</style>
```

En versiones anteriores de HTML se escribía `<style type="text/css">`, pero en HTML5 el atributo *type* en el elemento *style* es obsoleto.

- En un documento distinto, especificando con el elemento `<link>`

  Ejemplo: `<link href="css/bootstrap.min.css" rel="stylesheet">`

  - Es de tipo *void*
  - Puede aparecer varias veces, pero solo en la sección *head* nunca en *body*
  - No confundir con los enlaces a otros documentos HTML dentro del cuerpo del documento, que se indican con `<a>`

Elemento meta

Contiene diversos atributos con metainformación

- El único obligatorio es charset

```
<head>
  <meta charset="UTF-8">
  <meta name="description" content="Tutorial sobre tecnologías web">
  <meta name="keywords" content="HTML,CSS,Bootstrap,JavaScript">
  <meta name="author" content="Juan García">
</head>
```

Codificación de caracteres

En HTML antiguo lo habitual era emplear la codificación ISO-8859. En europa occidental, ISO-8859-1, también llamada latin1. O más bien windows-1252, que es muy similar

- En la actualidad la recomendación es usar UTF-8
- UTF-8 es una forma de codificar unicode, la más habitual pero no la única

La sintaxis de HTML 4 era muy farragosa

```
<head>
<META http-equiv="Content-Type" content="text/html;
↪  charset=ISO-8859-1">
...
</head>

<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
...
</head>
```

En HTML 5 :

```
<meta charset="UTF-8">
```

Problema: Hay varios lugares donde indicar la codificación

- Desde HTTP (tal y como lo haya configurado el administrador del servidor web)
- Dentro del HTML (tal y como lo configure el autor de la página)

Ambas informaciones pueden ser discrepantes. El convenio es dar precedencia a HTTP. Problema: un servidor que tenga mezcladas páginas HTML 4 (normalmente en ISO-8859-1) y HTML 5 (normalmente UTF-8)

- Se puede configurar el servidor para usar una codificación distinta para cada fichero o directorio, pero esto es específico de cada servidor web. Además, el autor de las páginas y el administrador del servidor web suelen ser personas distintas, no siempre bien coordinadas

Cuerpo del documento

El elemento body contiene el cuerpo del documento, su contenido principal

- El contenido del cuerpo incluye parrafos de texto, hiperenlaces (también llamados hipervínculos y enlaces), imagénes, tablas, listas, etc
- Solo puede haber un elemento body
- En HTML 4 tenía atributos como *background*, *bgcolor*, *link* o *text* para modificar los colores, pero no se admiten en HTML 5

Elementos de bloque / en línea

En el cuerpo del documento, los elementos pueden ser de dos tipos

- *Block elements*
  Siempre empiezan por nueva línea, y se muestran con cierto margen antes y después del elemento
  Ejemplos: *h1*, *h2*, ... *h6*, *table*, *form*, *ul*, *ol*
- *Inline elements*
  No empiezan con una nueva linea, ocupan el espacio que necesitan, pero sin margen adicional
  Ejemplos: *span*, *a*, *em*, *label*, *img*

p

Párrafo: secuencia de oraciones con unidad temática. Acaba en punto y aparte

- El elemento `<p>` crea un párrafo
- Entre un párrafo y otro hay
  - Un salto de línea
  - Una separación adicional
- La etiqueta de cierre `</p>` es opcional, si no aparece, se considera implícita antes de la apertura del siguiente elemento. Pero es preferible cerrar explícitamente con la etiqueta de cierre

Dentro de un `<p>` hay elementos

- Que no están permitidos.
  - Ningún elemento *de bloque* está permitido.
    (p.e. *h1*, *table*, *form*, *ul*)
  - Algunos elementos *de línea*, tampoco están permitidos
- Que sí están permitidos. Se denominan *phrasing content*

  `https://html.spec.whatwg.org/multipage/dom.html#phrasing-content`

  Son todos de tipo inline. P.e. *span*, *a*, *em*, *label*, *img*

Atención:

- Si dentro de un `<p>`, por error, escribimos un elemento no permitido

```
<p>Hola, mundo. <h1>Encabezado 1</h1></p>   <!-- ERROR -->
```

  cabría espera que el *W3C Validator* nos dijera algo como *elemento no permitido dentro de párrafo*

- Pero lo que sucede es que como un `h1` no puede estar incluido dentro de un `<p>`, se supone que hay un cierre implícito del párrafo antes de empezar el `h1`

```
<p>Hola, mundo.</p><h1>Encabezado 1</h1></p>   <!-- ERROR -->
```

- El mensaje de error será *No p element in scope but a p end tag seen*
  En otras palabras: *sobra la (segunda) etiqueta de cierre de párrafo*

br

La etiqueta `<br>` define el elemento *breaking line*

- En un documento HTML, los saltos de línea se ignoran
- Si queremos forzar un salto de línea (sin definir un párrafo), usamos `<br>`
- En cuanto al formato, es similar a un nuevo párrafo, pero solo con el salto de línea, sin la separación adicional
- Es de tipo *void* (no puede tener ni contenido ni etiqueta de cierre)

em

La etiqueta `<em>` define el elemento *emphasized*

- Típicamente el navegador mostrará el texto en cursiva

No confundir con la unidad *em*, que en español traduciríamos como *eme*, esto es, la letra eme

- Es una unidad clásica de tipografía, para referirse al ancho de la M mayúscula. `1em, 1.2em`, etc
- En la actualidad, las M mayúsculas suelen ser un poco más estrechas que un em. Así que *em* en la práctica significa *letra de tamaño normal*

pre

La etiqueta `<pre>` define el elemento *texto preformateado*

- El navegador muestra el texto respetando los espacios entre palabras y los saltos de línea
- Normalmente se usa una fuente de ancho fijo, típicamente courier

```
<pre>
#include <stdio.h>
int main() {
   printf("Hello World!");
   return 0;
}
</pre>
```

h1-h6

Las etiquetas `<h1>`, `<h2>`, ... `<h6>`, definen elementos *heading* (encabezado)

- Están organizados jerárquicamente, siendo h1 el más importante y h6 el menos

a

La etiqueta `<a>` define el elemento *anchor* (ancla), que sirve para hacer anclas y también para hacer hiperenlaces
Su nombre es poco afortunado, no describe lo que hace

- La invención del hipertexto se atribuye a D.Engelbart y T.Nelson, por separado, en 1962/1963
- En el hipertexto original solo había *anchors*: enlaces al documentos dentro de la misma sede
- Tim Berners-Lee inventa los enlaces a documentos en otros sitios (la WWW). Pero en el lenguaje HTML mantiene el nombre *anchor*, y por tanto, usa la etiqueta `<a>`
- En la actualidad, un *anchor* es un enlace desde una parte de un documento a otra parte del mismo documento. Pero el elemento `<a>` se usa para hiperenlaces

- Normalmente el elemento `<a>` se usar para hacer hiperenlaces, con el atributo *href (hypertext reference)*

```
<a href="https://www.urjc.es">Página de la URJC</a>
```

- También se pueden enlazar un correo electrónico

```
<a href="mailto:jperez@miempresa.com?Subject=Contacto%20web">
    Envíame un correo</a>
```

(Aunque hoy esto no es recomendable porque la dirección de correo queda expuesta a los spammers)

- O enlazar un script

```
<a href="javascript:alert('¡Hola, mundo!');">Holamundo en
↪  JavaScript</a>
```

Los enlaces pueden ser

- Absolutos a una dirección

```
<a href="http://linkedsite/url.html">Documento</a>
```

- Absolutos dentro del mismo sitio

```
<a href="/url.html">Documento en mismo sitio web</a>
```

- Relativos

```
<a href="url.html">Documento en mismo sitio web y "dir"</a>
```

anchors

Un anchor es una referencia a un punto concreto dentro de un documento

- Un anchor tiene un nombre, con las mismas reglas de cualquier otro atributo
- Para usar un anchor, a su nombre se le antepone la almohadilla
- Ejemplo de dirección con anchor:

```
https://gsyc.urjc.es/~mortuno/index_at.html#evaluacion
```

- Ejemplo de hiperenlace con anchor absoluto

```
<a href="https://www.urjc.es/universidad/org#rector>rector</a>
```

- Ejemplo de hiperenlace con anchor relativo

```
<a href="#inicio>inicio</a>
```

Anchor al estilo HTML 4

En HTML 4, un anchor se creaba definiendo un elemento `<a>`

- Sin atributo href
- Con atributo name

Ejemplo

```
<a name="punto07">Esto es el punto 7</a>

[.....]

<a href="#punto07">Volver al punto 7</a>
```

En HTML 5, esto sigue funcionando en los navegadores, pero es obsoleto, no deberíamos usarlo

Anchor al estilo HTML 5

En HTML 5, un anchor se crea añadiendo el atributo *id* a cualquier elemento contenedor *p, h1, div, pre...*

```
<h1 id="punto07">Punto 7</h1>

[.....]

<a href="#punto07">Volver al punto 7</a>
```

- Naturalmente, estos anchors pueden emplearse tanto de forma relativa como de forma absoluta
- Esta forma de crear anchors también solía funcionar en los navegadores HTML 4, aunque no era la más común

Atributo target

Añadiendo a un enlance el atributo `target="_blank"`, este se abrirá en una nueva pestaña del navegador

```
<a href="http://www.urjc.es" target="_blank">Abrir en nueva
↪  pestaña</a>
```

- En HTML 4 el atributo target tenía otros posibles valores relacionados con el uso de marcos (*_self*, *_parent*, *_top*, *framename*), pero en HTML 5 desaparecen los marcos

div

La etiqueta `<div>` define una división o sección dentro del documento

- Su uso habitual es el de contenedor genérico: delimita un bloque de texto, al que luego se le dará formato mediante reglas CSS

Ejemplo

```
<div class="respuesta">Todas son falsas</div>
```

En HTML 5, además de este elemento se definen otros con el mismo propósito, pero con una semántica más específica

- `<section>`, `<nav>`, `<article>`, `<aside>`, `<hgroup>`, `<header>`, `<footer>`, `<time>`, `<mark>`

span

La etiqueta `<span>` (espacio, longitud, lapso) define una división o sección dentro del documento

- Muy similar a div, pero no crea un bloque nuevo y por tanto, no crea una nueva línea
  - Y como no crea línea, no se pueden aplicar atributos de alineación
    `text-align: left | right | center | ... etc`
- Se usa típicamente para dar formato a un grupo de palabras

table, th, tr, td

Las etiquetas `<table>` (tabla), `<th>` (table header), `<tr>` (table row), `<td>` (table data) permiten crear tablas

```
<table>
  <tr>
    <th>Cabecera, primera columna</th>
    <th>Cabecera, segunda columna</th>
  </tr>
  <tr>
    <td>Primera fila, primera columna</td>
    <td>Primera fila, segunda columna</td>
  </tr>
  <tr>
    <td>Segunda fila, primera columna</td>
    <td>Segunda fila, segunda columna</td>
  </tr>
</table>
```

ol,li

Las etiquetas `<ol>` (ordered list), `<li>` (list item), permiten crear listas numeradas

```
<ol>
  <li>Sota</li>
  <li>Caballo</li>
  <li>Rey</li>
</ol>
```

- Las listas numeradas usan, por omisión, números naturales para cada item
- Añadiendo el atributo *type* al elemento *ol* se puede cambiar el tipo de marcador
  Los valores posibles son `1`, `A`, `a`, `I`, `i` para emplear números, letras mayúsculas, minúsculas, números romanos en mayúsculas y en minúsculas, respectivamente

ul,li

Las etiquetas `<ul>` (unordered list), `<li>` (list item), permiten crear listas sin numerar

```
<ul>
  <li>Sota</li>
  <li>Caballo</li>
  <li>Rey</li>
</ul>
```

dl,dt,dd

Las etiquetas `<dl>` (description list), `<dt>` (description term), `<dd>` (description), permiten crear listas de descripciones o definiciones de términos

```
<dl>
   <dt>
      Nombre
   </dt>
   <dd>
      Juan García
   </dd>
   <dt>
      Centro de origen
   </dt>
   <dd>
      ESTIT-URJC
   </dd>
</dl>
```

Ejemplos de tablas y listas:
`http://ortuno.es/tablas_listas.html`

img

La etiqueta `<img>` permite insertar imágenes
Ejemplos

```
<img src="urjc.png" alt="Logo de la URJC">
<img src="urjc.png" alt="Logo de la URJC" width="300" height="240">
```

Este elemento tiene dos atributos obligatorios

- *src*, que especifica el origen del fichero
- *alt*, que indica una descripción en texto del contenido de la imagen, para los navegadores sin gráficos

- El atributo *width* permite indicar el ancho en pixeles de la imagen en pantalla. Puede ser conveniente que la imagen original tenga un ancho algo mayor (para tener flexibilidad en el diseño). Pero evita que sea muy superior
- Es muy habitual incluir una imagen dentro de un elemento `<a>`, de esa forma la imagen se convierte en un hipervínculo

```
<a href="https://www.urjc.es">
   <img src="images/urjc.png" width="120" alt="logo de la URJC">
</a>
```

Observa que normalmente indicaremos el *path* (trayecto) del fichero con la imagen de manera relativa. El trayecto no empieza por el carácter *barra* (/)

- `urjc.png`

  En el mismo directorio donde está este html, hay un fichero llamando *urjc.png*

- `images/urjc.png`

  En el mismo directorio donde está este html, hay un directorio llamado *images*, y dentro, un fichero llamando *urjc.png*

- `../practica03/urjc.png`

  El directorio padre del directorio donde está este html, hay un directorio llamado *practica03*, y dentro, un fichero llamado *urjc.png*

Si anteponemos la barra, el significado cambia por completo

- `/urjc.png`

  En el directorio raiz, el fichero *urjc.png*

- `/images/urjc.png`

  En el directorio raiz, el subdirectorio llamado *images*, y dentro, un fichero llamando *urjc.png*

- `/practica03/urjc.png`

  El directorio raiz, el directorio *practica03*, y dentro, un fichero llamado *urjc.png*

El directorio raiz será:

- Si el fichero lo leemos localmente, el directorio raiz de nuestro sistema de ficheros (disco duro)
- Si el fichero se exporta via web, el directorio raiz establecido por el servidor web,

Naturalmente, estas mismas ideas sobre los trayectos relativos y absolutos, son aplicables en cualquier lenguaje de programación y a cualquier fichero: una imagen jpg, una librería, etc

- Observa que un *path* que incluya la dirección absoluta del usuario casi siempre es un error muy severo, porque deja de funcionar en cuanto cambia el usuario o la máquina

```
/home/alumnos/jperez/images/urjc.png
```

form

Un formulario HTML es un elemento que permite aceptar entrada de información por parte del usuario.

```
<form>
    (Elementos del formulario)
</form>
```

Los elementos posibles son varios

- El principal es el elemento `<input>` que puede de ser de diferentes tipos (`text, password, radio, checkbox`) entre otros
- Otros elementos son

  `<fieldset>` , `<select>` , `<textarea>` y `<button>`
- HTML5 añade los elementos `<datalist>` y `<output>`

Ejemplos de formularios:
`http://ortuno.es/form`

input: text, password, submit

`input` es un elemento HTML de tipo void

- El atributo *name* indica el nombre de campo
- Con el atributo *value* se pueden asignar valores por omisión
- El input de tipo *submit* es el botón *enviar*. Se puede cambiar su texto con el atributo *value*

```
<form action="/action_page.html">
  Nombre de usuario:<br>
  <input type="text" name="usuario" ><br>
  Contraseña:<br>
  <input type="password" name="contrasenya" ><br><br>
  País:<br>
  <input type="text" name="pais" value="España" ><br><br>

  <input type="submit">
</form>
```

Observa que

- Para el *name* no podemos usar letras españolas como la eñe (porque este será el nombre de una variable en el código del servidor que procese el formulario)
- Para el *value* sí podemos. Este será el valor de una variable en el código

input: radio

`<input type="radio">` define un *radio button*, que permite elegir una (y solo una) opción entre varias

```
<form>
  <input type="radio" name="os" value="Linux" checked>Linux<br>
  <input type="radio" name="os" value="macOS" >macOS<br>
  <input type="radio" name="os" value="Windows">Windows<br>
  <input type="radio" name="os" value="other">Otro<br>
</form>
```

input: checkbox

Checkbox es un tipo de input que permite elegir 0 o más opciones de una lista

```
<form>
  <input type="checkbox" name="terminos" value="si">
        He leido los términos y condiciones<br>
  <input type="checkbox" name="publicidad" value="si">
        Deseo recibir comunicaciones comerciales<br>
</form>
```

input: tipos de HTML5

HTML 5 añade nuevos tipos de input

- color (no soportado por todos los navegadores)
- date (no soportado por todos los navegadores)
- datetime-local (no soportado por todos los navegadores)
- email
- month
- number
- range
- search
- tel
- time
- url
- week

fieldset

Un conjunto de entradas se pueden agrupar en un `<fieldset>`, con un título indicado en un elemento `<legend>`

```
<form>
  <fieldset>
    <legend>
      Datos personales
    </legend>

    Elija un color:
    <input type="color" name="favcolor">
    <br> Fecha de nacimiento:
    <input type="date" name="nacimiento">
    <br> Fecha y hora de nacimiento:
    <input type="datetime-local" name="nacimiento-hora">
    <br> E-mail:
    <input type="email" name="email">
    <br> Indica un número del 1 al 10:
    <input type="number" name="numero" min="1" max="10">
    <br>
    <input type="submit">
  </fieldset>
</form>
```

select

El elemento `<select>` permite elegir una opción entre varias

```
<form>
   Indique el departamento:
   <select name="departament">
      <option value="sales">Comercial</option>
      <option value="technical">Técnico</option>
      <option value="webmaster">Webmaster</option>
   </select>
   <input type="submit">
</form>
```

El elemento `<datalist>` es similar pero permite que el usuario escriba una respuesta distinta a las propuestas

textarea

Con el elemento `<textarea>` el usuario puede introducir varias líneas de texto

```
<form>
   <textarea name="mensaje" rows="10" cols="30">
   Escriba aquí su mensaje.
   </textarea>
</form>
```

label

Para que el usuario sepa qué es cada elemento de un formulario, se puede usar:

- Texto HTML ordinario
- Un elemento `<label>`
  Ventajas:
  - Haciendo clic sobre esta etiqueta, se activa el elemento
  - Facilita su interpretación en entornos distintos a navegadores tradicionales, p.e. lectores de HTML
  - Facilita el estilo consistente

Para usar `<label>` hay que

- Añadir un atributo `<id>` al elemento
- Poner en el label un atributo `for` cuyo valor sea el del id

```
<form>
  <label for="ciudad">Ciudad de procedencia:</label>
  <input type="text" name="ciudad" id="ciudad">
  <input type="submit">
</form>
```

- `name` lo use el servidor
- `id` lo usa el navegador

Típicamente se hace que coincidan pero son independientes, no tienen por qué ser iguales

En el caso del `<input type="radio">` y el `<input type="checkbox">`

- Normalmente es innecesario usar `<label>`, el texto dentro del elemento suele ser bastante descriptivo
- Si queremos incluir un `<label>`, se usa como en cualquier otro elemento, un `<label>` por cada input
  - Aunque es recomendable que en el HTML escribamos el `<label>` después del `<input>`, para que el navegador muestre la etiqueta a la derecha del `<input>`, no a la izquierda

Elementos y Atributos obsoletos en HTML5

En HTML 4 había muchos elementos y atributos relacionados con el formato. Algunos de los más habituales:

- align (left, right, center)
- color
- font
- u (underline)

Todos ellos han desaparecido en HTML 5, en su lugar debe usarse CSS [1]

[1] En el elemento span sigue siendo legal usar atributos gráficos, pero no es recomendable. Siempre es preferible CSS

Entities

Las *entities* se usan para

- Representar caracteres que coinciden con metacaracteres de HTML. Es la única forma de indicar caracteres como <
- Representar caracteres que el desarrollador no tenga en su teclado. Su uso es opcional, con UTF-8 siempre se puede escribir cualquier caracter (mediante teclados virtuales o copiando y pegando desde otro sitio)

Cada entity tiene un nombre y un número, se puede usar cualquier de las dos formas

- *Ampersand*, nombre, punto y coma

```
&lt;
```

- *Ampersand*, almohadilla, número, punto y coma

```
&#60;
```

Algunas entities habituales

- <

```
&lt;      &#60;
```

- >

```
&gt;      &#62;
```

- €

```
&euro;    &#8364;
```

- Ñ

```
&Ntilde;  &#209;
```

- ñ

```
&ntilde;  &#241;
```

Otra entity frecuente es ` ` *non-breaking space*
Es un espacio que:

- Siempre se representa
- Nunca se usa para partir una linea. Ejemplo: para escribir 2 € con garantías de que ambos símbolos estarán en la mísma línea:

```
2 &euro;
```

Material complementario

- HyperText Markup Language (Wikibook):
  `http://en.wikibooks.org/wiki/HTML_Programming`
- HTML5: A tutorial for beginners:
  `http://www.html-5-tutorial.com/`
- Dive into HTML5:
  `http://diveintohtml5.info`
- HTML5 (Wikipedia):
  `http://en.wikipedia.org/wiki/HTML5`
- Web Fundametals (Code Academy):
  `http://www.codecademy.com/tracks/web`