Curso HTML y CSS

Esta breve guía pretende sintetizar y resumir a grandes rasgos los elementos principales del html y las hojas de estilo (css). Estos apuntes, pueden ayudarte a modo de esquema o resumen con lo esencial para empezar a crear páginas sencillas desde cero.

Introduccion

El desarrollo se realiza en base a una parte cliente y otra servidor.

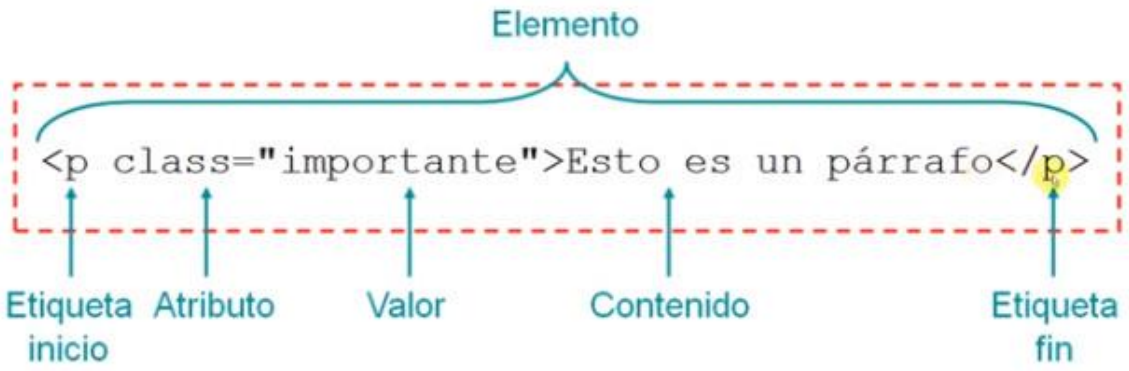
Parte cliente

- HTML + CSS
- JavaScript + DOM

Parte Servidor (Backend)

- Lenguajes de script (php, asp.net,JSP,Perl y Coldfusion)
- Diseño y desarrollo de la Base de datos.
- Seguridad.

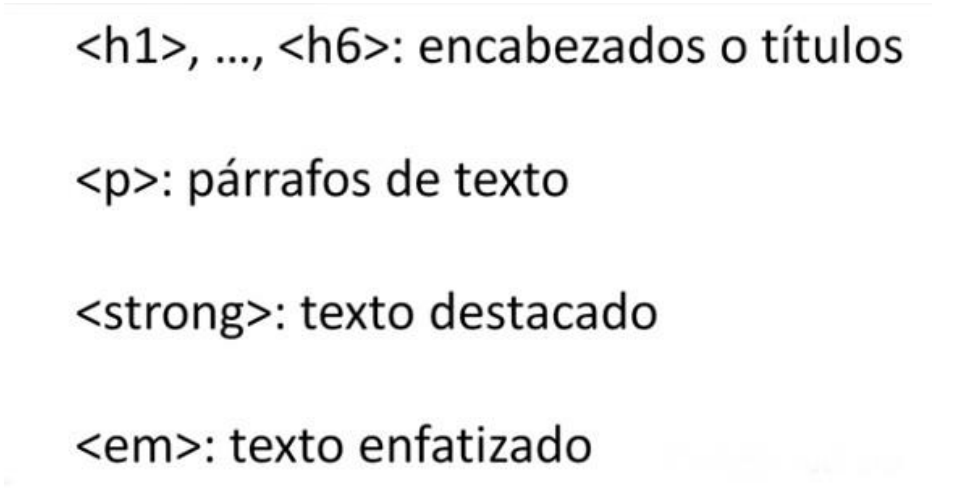
Estructura Principal <></>



Inicialmente podemos escribir código html en cualquier editor de texto. Si estamos en Windows, podemos utilizar el Bloc de Notas. Al guardar debemos especificar la extensión y poner preferiblemente antes de guardar como tipo UTF-8

Algunos ejemplos de etiquetas:

webpositiva



- **Ejemplo de Página**
- <html> Etiqueta principal raid
- <head> Cabecera, metadatos, instrucciones
- <title>Esto es el título de la página. Va anidado dentro de la head</title>
- </head>
- <body>
- Aquí va el contenido visible y bajamos un renglón
- Esto es negrita.
- Y esto itálica.
- </body>
- </html>

En html, los elementos se pueden anidar; es decir meter unos dentro de otros, aunque siempre respetando unas reglas que iremos viendo.

etiquetas básicas (html, head, title, body, h1, h2, h3, h4, h5, h6, p, strong, em).

principales etiquetas:

<!--

<head> para información sobre el documento

 división dentro del contenido

<a> para enlaces

 para poner el texto en negrita

 define textos enfatizados (en cursiva)

 para saltos de línea

<H1>…<H6> para títulos dentro del contenido (y van del 1 al 6)

<img> para añadir imágenes al documento

 para listas ordenadas, <ul> para listas desordenadas, <li> para elementos dentro de la lista

<p> para parágrafos

 para estilos de una parte del texto -->

Conceptos básicos

Podemos utilizar como alternativa al blog de notas el programa de edición llamado OpenSource **Notepad ++**. Este entre otras cosas, nos permitirá ahorrar tiempo para crear etiquetas, nos marcará los errores, posibilita la instalación de plugins para ampliar sus funcionalidades, etc.

Intalamos plugin TextFx (En plugins lo seleccionamos y pulsamos en instalar) y Cerramos Apli.

Abrimos el programa y vamos al menú TextFx/TextFx Settin/Autoclose Tag

* Este plugin permite que cuando abrimos una etiqueta, nos crea la misma con el cierre.

Con Ctrl+ nos aparece listado de etiqutas (el + del teclado alfanumérico, no el del numérico)

Cuando marcamos una etiqueta, el programa nos ayuda y nos señala su correspondiente etiqueta ce cierre.

También nos permite numerar y desplegar o cerrar las etiqutas.

Reglas recomendables

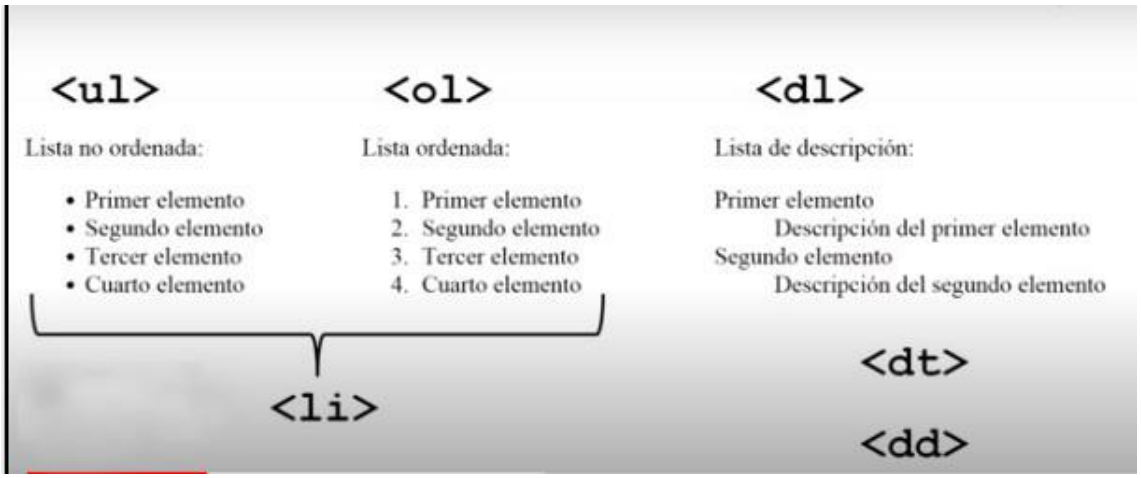
Aunque las siguientes reglas no son necesarias en html5, si lo eran en el html y xml inicial y es recomendable mantenerlas:

1- Todo etiqueta que se abre, se debe cerrar.
2- Con etiquetas anidadas, la última en crearse es la primera en cerrarse y la primera abierta, será por tanto la última cerrada.
3- Mejor escribir las etiquetas en minúsculas.
4- Los valores de los atributos deben ir entre comillas dobles “” o simples ' '.
5- Los nombres de los ficheros, mejor con el alfabeto inglés, números, guión medio, guión bajo, sin espacio, con extensión htm o html.

Tipos de listas

1- Ordenadas. •
2- No ordenadas. **1, 2, 3**
3- De descripción. <dl> **No puntea, mete una sangría en cada punto para describir.**

webpositiva




```
Lista no ordenada:
<ul>
<li>Primer elemento</li>
<li>Segundo elemento</li>
<li>Tercer elemento</li>
<li>Cuarto elemento</li>
</ul>
```

Lista no ordenada:

- Primer elemento
- Segundo elemento
- Tercer elemento
- Cuarto elemento

Lista ordenada:

1. Primer elemento
2. Segundo elemento
3. Tercer elemento
4. Cuarto elemento

Lista ordenada:

A. Primer elemento
B. Segundo elemento
C. Tercer elemento
D. Cuarto elemento

```
<ol type="A">
```

Lista ordenada:

a. Primer elemento
b. Segundo elemento
c. Tercer elemento
d. Cuarto elemento

```
<ol type="a">
```

Lista ordenada:

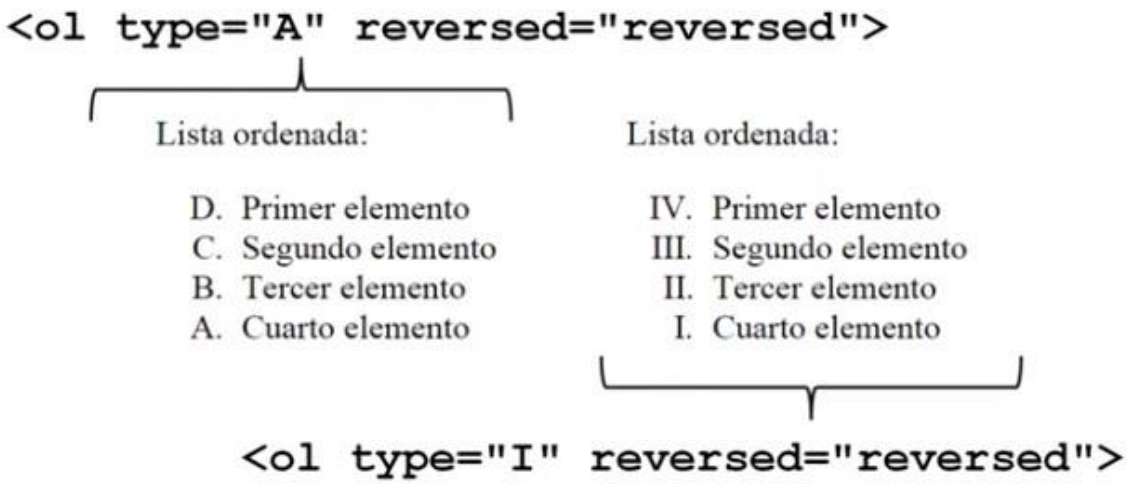
I. Primer elemento
II. Segundo elemento
III. Tercer elemento
IV. Cuarto elemento

```
<ol type="I">
```

Lista no ordenada:

i. Primer elemento
ii. Segundo elemento
iii. Tercer elemento
iv. Cuarto elemento

```
<ol type="i">
```



El atributo reversed es boleano, lo que significa que sólo puede tomar un valor (no admiten los valores true o false). El atributo start, propio de html4, también se pudiera haber utilizado con el mismo resultado; es decir indicar desde qué número empieza, en esta caso hubiera sido el 4.

En HTML5 es posible crear listas anidadas; es ceir listas con diferentes características dentro de otras. Vemaos un ejemplo:

```
<ul>
<li>Números cardinales:
<ol>
<li>Uno</li>
<li>Dos</li>
<li>Tres</li>
</ol>
</li>

<li>Números ordinales:
<ol>
<li>Primero</li>
<li>Segundo</li>
<li>Tercero</li>
</ol>
</li>
</ul>
```

- Números cardinales:
  1. Uno
  2. Dos
  3. Tres
- Números ordinales:
  1. Primero
  2. Segundo
  3. Tercero

Crear enlaces (hipertextos)

Documento digital compuesto por enlaces o vínculos entre nodos

- Hipertextdo: documento digital que se puede leeer de forma no secuencial y está compuesto por:
- Nodos o secciones: partes del hipertexto que contienen información accesibles para el usuario.
- Enlaces: uniones o vínculos que se establecen entre nodos.
- Anclajes: puntos de activación de los anclajes.

Tipos de enlaces:

Se utiliza con la etiqueta <a>enlace

1. Intradocumentales: enlaza en a una parte de la página.
   1.1. Atributo href y almoadilla enlace
   1.2. enlace
   1.3. enlace
   1.4. **En html5** se emplea el atributo id que se puede definir en cualquier etiqueta del html <**h1 id**="nombre">enlace</h1>
   1.5. Otro ejem en html5 <**p id**="nombre">enlace
2. Extradocumentales: a otra documento o página.
   2.1. **Enlace:** enlace
   2.2. **Destino:** Vamos a ver las definiciones **antiguos** y la nueva **html5**
      2.2.1. **Antiguo**: enlace
      2.2.2.Ejem **hml5:** <h1 id="nombre">enlace</h1>
      2.2.3.Ejem **html5:** enlace

* El nombre del destino va precedido de almohadilla, pero en el destino, no se pone.

Estructura de la url (Uniform Resource Loacater)

<servicio>://<usuario>:<contraseña>@<host>:<puerto>/<ruta>/<recurso>

Saltos de línea

</br> también **
</ br>** Esta última es propia de xml/xhtml. Html5 reconoce las dos formas, por lo que recomiendo la segunda.

No debemos confundir con propio de la separación de párrafos.

Referencias de Caracteres

Html ofrece una alternativa para utilizar caracteres que por defecto no ofrece. Esto se llama "escapar los caracteres" **Estas alternativas empiezan por "&" y terminan con ";"**

- **Varios espacios:** Html no distinue por defecto varios espacios. Por ejemplo, las siguientes líneas de código tienen el mismo resultado. En general, no se debe utilizar por una cuestión de presentación, para ello se debería utilizar CSS.

Hola mundo!

Hola mundo!

No obstante, si decidiéramos hacerlo lo haríamos con ** ** que viene de **No-break space**.

- Referencias de **caracteres decimales** &#nnnn; Estas referencias llevan #número
- Referencias de **caracteres hexadecimales** &#xnnnn; (dígitos de 0 a 9 y letras a – f)
- **Acentos** á (acentuará la a)

Como muchos caracteres debemos buscar y hacernos con una tabla para saber qué código corresponde a cada carácter. En cualquier caso podemos ahorrárnoslo para un idioma buscando el corresponidiente ISO; en el caso del **español la codificación correspondiente es la ISO-8859-1 (Latin1) UTF-8**.

El Wireframe (La estructura de diseño, el esqueleto de la página)

Html ofrece una alternativa

Registro de dominios

Está la ICANN como organismo internacional de registro de dominios. Puedes comprobar si un dominio está registrado a través de ellos o más directamente a través de whois.sc

Los dominios nacionales están regulados por redes.es (dominios.es ó nic.es)

Cuando registras, tienes la posibilidad de poner el whois privado de tal manera que el nombre del propietario queda protegido y no visible. Suele quedar visible la empresa por la que has registrado “el registrador”

Hay nombres de dominio que están protegidos o corres el riesgo de que te lo expropien. Por ejemplo reyfelipe6.es

Formularios

Aspectos clave

Explicamos los formularios tradicionales con HTML (qué es un formulario, para qué sirve un formulario), se muestran las etiquetas y atributos que se emplean en un formulario y los métodos de envío (GET, POST). Además, se explican los siguientes controles de un formulario:

Estructura de Formulario
❖ **Form** **|Action | Metod** (get o Post**) |accept-charset** (espera lista de caracteres, por ejemplo UTF 8, no es muy común) **| enctype** (tipo de codificación, se utiliza por ejem cuando vayamos a subir ficheros mediante formulario)

Métodos de Envío de Formularios	
• **Get** (visbles, por ejem formulario de búsqueda)	• **Post** (no visible, no tiene límite de envío, perite envío de ficheros) Por ejemplo un Formulario de Registro.
Principales Etiquetas	
• Select	• Input

Select: Listas desplegables

El atributo value Para definir cada uno delos elementos utilizamos la etiqueta <option value>opción A </option>

Multiple=”multiple” permite generar listas de selección múltiples

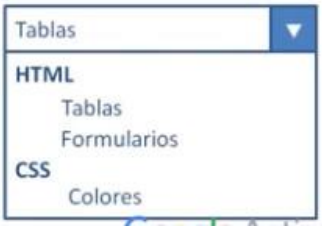
```
<select>
    <option value=“HTML”>
      HTML
    </option>
     <option value=“CSS”>
      CSS
    </option>
</select>
```

Si añadimos la etiqueta multiple lo convierte en lista no desplegable

```
<select multiple=“multiple”>
```

La etiqueta opgroup permite crear bloques

```
<select>
    <optgroup label="HTML">
        <option value="tables"> Tablas</option>
        <option value="forms">  Formularios </option>
    </optgroup>
    <optgroup label="CSS">
    ...
</select>
```



Input. (no necesita etiqueta de cierre)

Es de la más utilizadas, porque podemos usar gran parte de los controles del formulario. No tiene etiqueta de cierre, pero es interesante cerrarla para cumplir con los estándares.

1. Ejemp <input type="value" />
2. Name: El atributo es común a otros elementos, ya que otorga el nombre del campo.
3. Type: es uno de los atributos más importantes.
   - 3.1. text. Campo de texto
     - 3.1.1.Atributo **maxlength=**"número" (máximo de caracteres permitidos)
     - 3.1.2. Atributo **size=**"número" (longitud del campo, su ancho en caracteres. Si usas css mejor no utlizar)
     - 3.1.3. Atributo value""text" (si es de texto o numérico)
   - 3.2. checkbox. Casilla de verificación
     - 3.2.1.El atributo checked="Checked Selecciona un valor por defecto.
   - 3.3. Value: Indica al formulario qué valor debe enviar si la casilla está activada radio o checkbox: Casilla de opción <input type="checkbox" value="html">
   - 3.4. file. Subir ficheros
   - 3.5. password. texto enmascarado aparece como ****
   - 3.6. hidden. campo oculto . Por ejem un script que cronometre el tiempo en rellenar form.
   - 3.7. button. Botón:
     - 3.7.1.submit. Botón de envío
     - 3.7.2.image. Imagen momo botón de enviar
     - 3.7.3.Reset: Botón de limpiar el formulario
     - 3.7.4.eventos javascript
4. Disabled: deshabilita el campo para que no pueda ser completado
5. Readonly: Funciona de manera parecida a Disabled pero sería posible seleccionar el texto y copiarlo en un campo tipo texto.

Textarea

Similar a text, pero define un campo más extenso para escribir texto. Su anchura se configura a través de los atributos cols y rows.

Label

Asocia un texto a un control; para vincularlo con su correspondiente control, se utiliza “for”

```
<form
 <laber for="html">HTML</label>
 <input type="radio" name="tema" id="html" />
 <br />
 <laber for="css">CSS</label>
 <input type="radio" name="tema" id="css" />
</form>
```

fielset

Agrupa elementos de un formulario

```
<fieldset><legend>Perfil</legend>
Nombre: <input type="text"><br/>
Email: <input type="text"><br/>
</fieldset>
<fieldset><legend>Temas</legend>
<input type="checkbox">HTML
<input type="checkbox">CSS
</fieldset>
```

Escribir una página web bien estructurada

Juego de Caracteres

La primera regla es **utiliza siempre el mismo juego de caracteres** para el código, para la BD, para servicio web, etc, css. Lamentablemente, esto no siempre es posible. Como recomendación **utilizar siempre UTF-8 sin Bom**, ya que permite utilizar textos de cualquier idioma. En caso de no poder utilizarlo, entonces debemos utilizar ISO-8859-1 Latin1, que corresponde al alfabeto de los idiomas europeos occidentales, aunque si quieres utilizar el símbolo del euro “€” en esto caso debemos utilizar ISO-8859-15 Latin9, prácticamente no tienen diferencia entre uno y otro, sólo en 4 caracteres poco utilizados.
Utilizar UTF-8 sim Bom siempre va a ocupar un poquito más los ficheros, ya que cada carácter acentuado son 2bytes. También es importante recordar que no será añadir al código la estructura &eacute después de las vocales que precisan acento por ejemp: **a&eacute = á**

¿Cómo indico el juego de caracteres en el html?
Se realiza mediante una etiqueta <meta> que se escribe en el <head> de la página.
Para HTML5:
<meta charset="utf-8">
<meta charset="utf-8" />

Para html4: <meta http-equiv="Content-Type" content="text/html; charset=utf-8">

Para html1: <meta http-equiv="Content-Type" content="text/html; charset=utf-8">
o también <meta http-equiv="Content-Type" content="text/html; charset=utf-8"></meta>

Además, si utilizas juego de caracteres diferente, tendrás que declarar el juego de caracteres mediante la declaración de documento xml al principio de la página
<?xml version="1.0" encoding="ISO-8859-1"?>

*Bom viene de Byte Order Mark, es un carácter "Unicode" que se utiliza para indicar el orden de los bytes de un fichero de texto al principio de este. Existen dos tipos de byte, el "Litle-endian" y el "Big-endian" que definen la forma de almacenamiento de los datos que ocupan más de un byte en el ordenador.

Bom ocupa 3 byte y aparece representado como EF BB BF su representación en forma de carácter es: ï>> ¿

Hay programas que no se llevan bien con Bom por ejemplo php5. Además lógicamente debe ser coincidente la etiqueta meta del juego de caracteres que indiquemos y la codificación cuando almacenemos en blog de notas o Notepad++.

¿Cómo migrar a otro juego de caracteres

Suele ser más necesario cuando son web dinámicas sobre una BD. Si hemos iniciado una web puramente en html que se inició con un juego de caracteres distinto debemos plantearnos si realmente lo necesitamos migrar para los ficheros antiguos. La migración de ficheros depende del Sistema Operativo:

1. Linux: Comando "file" (para conocer tipo y codificación de un fichero) y el comando "iconv" (convierte la codificación de un fichero).

2. Windows: Es más complicado, ya que se tendría que hacer a través de un editor de texto.

Crear Tablas

Principales etiquetas:

<table></table> abre y cierra la tabla
<tr><tr/> filas

<td><td/> celdas
<th><th/>celdas de cabecera (primeras y últimas filas)
<thead></ thead >Agrupa filas de cabecera.
<tbody></ tbody >Agrupa filas de cuerpo.
<tfoot></tfoot > Agrupa filas de pie
<caption><caption/> Añande un título sobre la tabla

Ejem Atributos para combinar celdas:
<td colspan="2012"></ td >
<td rowpan""enero"></ td >
<celspan></celspan >

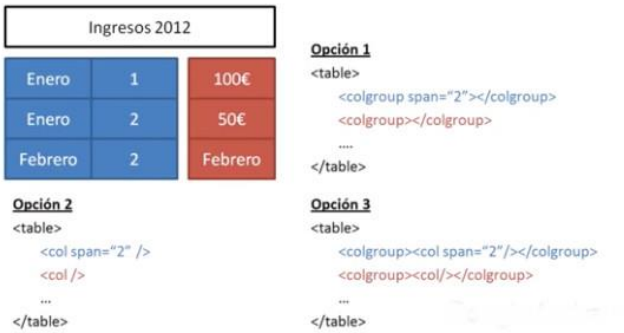
Ingresos	
2012	
Enero	100€
	50€
Febrero	75€

```
<table>
    <caption>Ingresos</caption>
    <thead>
        <tr><td colspan="2">2012</td></tr>
    </thead>
    <tbody>
        <tr>
            <td rowspan="2">Enero</td>
            <td>100€</td>
        </tr>
        <tr><td>50€</td></tr>
    </tbody>
    <tfoot>...</tfoot>
</table>
```

Columnas
<colgroup></colgroup> abre y cierra columna
<col><col/> No tiene etiqueta de cierre, pero conviene hacerlo por seguir con el estándar.

El atributo spam indica el número de columnas agrupadas:
<table>
<colgroup spam="2"></colgroup>

Ejem Atributos para combinar celdas:
<td colspan="2012"><td/>
<td rowpan""enero"><td/>
<celspan><celspan/>



Insertar Imágenes

La etiqueta que se utiliza **es <img>** El tipo de imagen depende del navegador, pero lo normal es jpeg, gif, png.

Atributos obligados:

- Src que indica la ruta
- Alt define el texto alternativo al posicionar el cursor.

Atributos opcionales:

- width y height para definr anchura y altura
- longdesc definir la url de la página en la que se proporciona la descripción larga de la imagen

Atributos obsoletos desplazados por CSS (desanconsejable utilizar)

- align, border, hspace, vspace
- <Body background=""> imágenes como fondo de página.
- <table background=""> imágenes como fondo de tabla.

Mapas de imagen

Fragmentos de la imagen que enlazan con otras partes. Se suelen utilizar en mapas geográficos, mapas de oficinas, barras de navegación, etc. Se pueden **procesar desde el lado del cliente (usemap)** o desde el lado del servidor (ismap).

Mapa de imagen

- Tres tipos de figuras geométricas:

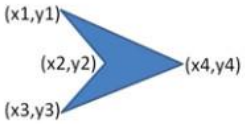
```
<area shape="" coords="" href="" alt="" />
  - shape="rect" → coords="x1,y1,x2,y2"
```

- Tres tipos de figuras geométricas:

```
<area shape="" coords="" href="" alt="" />
  - shape="rect" → coords="x1,y1,x2,y2"
  - shape="circle" → coords="x,y,r"
```

- Tres tipos de figuras geométricas:

```
<area shape="" coords="" href="" alt="" />
  - shape="rect" → coords="x1,y1,x2,y2"
  - shape="circle" → coords="x,y,r"
  - shape="poly" → coords="x1,y1,x2,y2,..,xn,yn"
```



Validación del código html

En la actualidad conviven html4, xhtml1 y html5. Igual que existen errores en la escritura del lenguaje convencional, también existen los errores de código, por lo que debemos de validar el mismo previamente.

La versión de html que vamos a utilizar en documento, se define al principio con <!DOCTYPE> Es bastante desconocida esta etiqueta, porque abarca más de lo que parece, pero en general podría ser así:

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

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

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

```
<!DOCTYPE HTML>
```

Este último, el más sencillo es para HTML5.

Para validar el código se puede hacer a través de varios sw y url, por ejemplo la del w3c https://validator.w3.org/ La validación es importante para asegurarnos que carga bien en los diferentes navegadores y no contiene errores.

DHTML (html dinámico)

Permite crear sitios web interactivos a través del uso combinado de las siguientes tecnologías:

- HTML: Lenguaje de marcado.
- CSS: Hojas de estilo en cascada.
- DOM: Jerarquía de objetos.
- JAVASCRIPT: Lenguaje interpretado.

Existen otras tecnologías como AJAX que se ejecutan en el propio navegador y reúnen las siguientes herramientas como la Api de programación XHR, el metalenguaje XML y el formato de intercambio de datos JSON. Todo esto se conoce como AJAX.

Familia de Tecnologías HTML5

<CANVAS> o lienzo permite definir un área dentro de una página en la que se puede dibujar mediante una API para JavaScript. Esta etiqueta ha sido muy relevante, porque ha desplazado a Flash para por ejemplo juegos online, aplicaciones de dibujo.
<canvas id="myDrowing" width="200" height="200"> </canvas>

SVG es un lenguaje de etiquetas basado en XML que permite crear **gráficos vectoriales**.

```
<rect
   x="0" y="0"
   width="100" height="100"
   fill="blue" stroke="red"
   stroke-width="5px"
   rx="8" ry="8"
   id="myRect" class="chart" />
```

API de geolocalización para JavaScript. Permite la geolocalización cuando el dispositivo lo permite.

```
navigator.geolocation.getCurrentPosition(
  function(position) {
    var lat = position.coords.latitude;
    var lon = position.coords.longitude;
    showLocation(lat, lon);
  }
);
```

La tecnología de Caché, permite que determinados recursos de una página sean almacenados de forma local. De esta forma, se mejora el rendimiento y permite descargar todos los contenidos necesarios y trabajar en local.

La tecnología de Base de Datos es una API que ofrece la posibilidad de almacenar en local, en el navegador mediante una base de datos SQL Lite. Con esta tecnología, podemos crear por ejemplo un sistema de notas que conserva la información aunque se abandone la página web.

Los Web Workers es otra API para JavaScript que permite crear múltiples hilos de ejecución que se ejecutan en paralelo.

De todas estas tecnologías hemos de tener en cuenta donde se va a ejecutar, navegadores, etc. Para estudiar compatibilidad podemos acudir a algunas webs de referencia como:

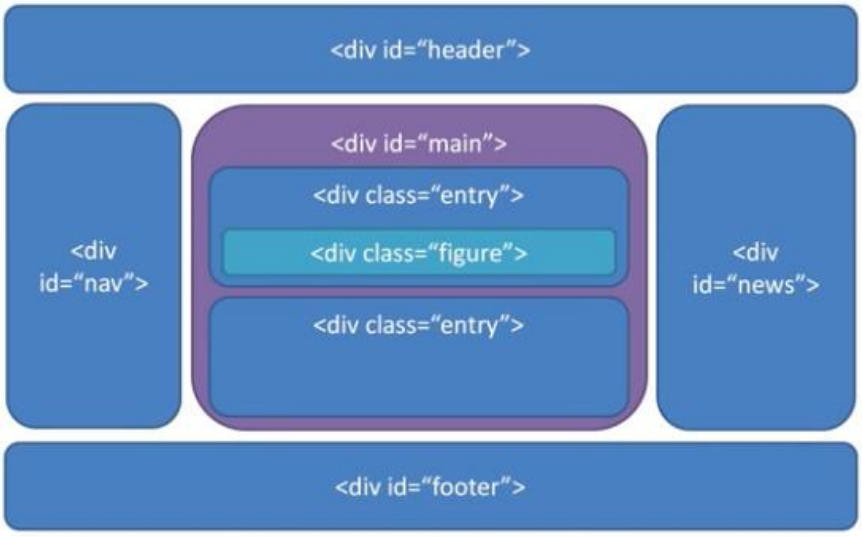
- https://caniuse.com/
- Find me by Ip
- Para saber más sobre html Google dispone de https://html5rocks.com/

Pasar de las versiones antiguas de HTML a HTML5

Con el HTML original existían dos problemas muy claros:

1. <Span> manía
2. itis

La estructura que se utilizaba era algo como lo siguiente:

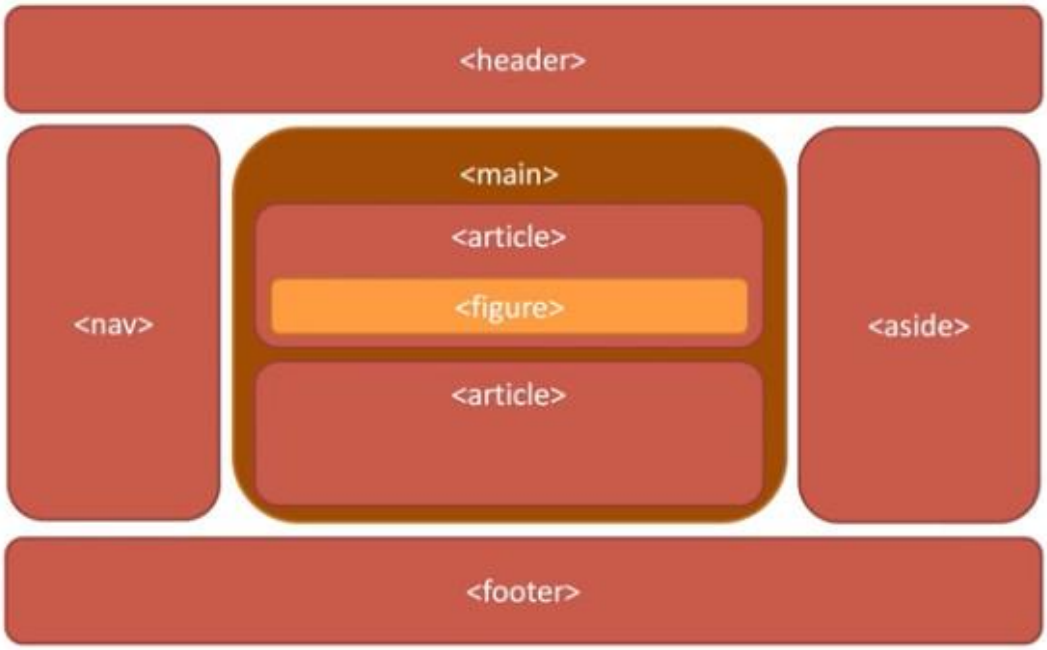


El problema, fue el abuso de éstas, ya que se anidaban demasiadas etiquetas dentro de cada una de ellas.

Con HTML5, se crearon otras etiquetas para evitar esos excesos:

- <header>: cabecera
- <nav> menús de navegación.
- <main> Zona de contenido Principal.
- <article> Entras o Artículos
- <section> Secciones.
- <aside> Paneles de información adicional.
- <figure> Imágenes.
- <footer> Pie de página.

La Estructura pasó a ser algo así en HTML5:



- Conversión de HTML a HTML5. En general debemos de cambiar estas etiquetas:
- <div id=header"> por <header>
- Una etiqueta div que envuelve a una lista se debe cambiar por <nav>
- El contenido principal en html5 podría ser algo como:

HTML5

```
<main>
<article>
<p class="post-date">Diciembre 26, 2011</p>
<h2>
<a href="#" rel="bookmark" title="Enlace a esta
   entrada">HTML5 es lo mejor</a>
</h2>
<!-- Resto de la entrada -->
<p>...</p>
</article>
<!-- Otras entradas -->
</main>
```

Con la etiqueta <figure> podemos relacionar la imagen y su título. También se puede etiquetar fragmentos de código o una cita.

```
<figure>
<img src="foto.jpg" alt="Tim Berners-Lee" />
<figcaption>Tim Berners-Lee en la inauguración del curso
  de verano</figcaption>
</figure>
```

```
<figure>
<figcaption>Validación de una dirección de
  correo</figcaption>
<pre>
function validarEmail(valor) { if (/^\w+([\.-
  ]?\w+)*@\w+([\.-]?\w+)*(\.\w{2,3})+$/.test(valor))
  return (true); else { alert("El usuario introducido es
  incorrecto"); return (false); } }
</pre>
</figure>
```

Formularios HTML5

Permiten ejecutarse desde el lado del cliente y además incorpora funcionalidades que antes requerían de JavaScript. Además permitirá ahorro de tiempo y ancho de tiempo, evita envíos inválidos y otorga mejor compatibilidad con diferentes dispositivos. Estas son las dos principales áreas de funcionalidades resumidas:

1. **Nuevos valores para <input> type:** url, email, number, color, search, range, datetime, datetime-local, date, month, week, time.
2. **Nuevos atributos de contenido:** autocomplete, autofocus, min, max, pattern, paceholder, required. Además Modernizr y Polyfills.

* tel (teléfono), range (valores numéricos entre rangos), datetime (seleccionar hora y fecha teniendo en cuenta zona horaria), datetime-local (igual pero sin tener en cuenta zona horaria. Si este detalle no es importante, mejor usamos este atributo.)

Tipos de input dependiendo del dispositivo móvil.
Hasta hemos tenido en cuenta los navegadores web desde el punto de vista de equipos de mesa. Vamos a ver las mejoras de usabilidad que introducen los tipos de input: time, email y url, cuando lo que estamos usando es un dispositivo con pantalla táctil.

- Input tye:"tel" El teclado numérico cambiará de formato y se actualizará para la introducción de un número de teléfono (teniendo en cuenta sistema operativo: Android, Iphone, etc".

- Input tye:”email” Mostrará teclado alfanumérico. En el caso del Iphone incluye una tecla con “@”
- Input tye:”url” Igual que el anterior, pero incluirá una tecla con “.com” en función del SO.

Las mejoras de los formularios HTML5 son también visibles en los campos, ya que dan pistas de los formatos a introducir para el usuario, calendarios interactivos, selección de colores desde una paleta, etc.

Respecto a los **atributos de contenido** vamos a profundizar un poco en ellos:

- autocomplete permite el autocompletado de un campo que el usuario haya rellenado anteriormente en otros formularios. Se puede activar "on" o desactifar "off" (sobre todo por seguridad) evitando que un campo quede al descubierto.

<form autocomplete"on">
<input type="email" name="email" autocomplete="off">

- Autofocus nos permite determinar en qué campo de entrada queremos que se coloque el foco al cargar la página. Es booleano, por lo que no deberemos definir más de un elemento.

- Min y Max determinan rangos de valor.

<input type="number" name="cantidad" min="2" max="14"> (entre 2 y 14)

- Step especifica los atributos numéricos válidos para un elemento input; Crea una regla. Puede ser usado con min y max:

<input type="number" name="cantidad" min="2" max="14" step="2" value="2">

En este caso sólo se podrá seleccionar valores entre 2 y 14, como valor por defecto el 2 y en valores de 2 en 2. Es decir permitirá seleccionar el 2,4,6,8,10,12 y 14.

- Pattern permite validar una elemento input en base a una expresión regular.Es decir modelos o patrones que se pueden escribir en un campo (text, search, url, tel, email, passord).

<input type="text" name="prefijo:pais" pattern="A-Za-z){3}" title="Código de 3 letras" />

- Title muestra un rótulo a modo de pista al usuario para que entienda la regla.

- Placeholder permite mostrar una pista o ejemplo de lo que el usuario debe introducir. Se muestra mientras está vacío.

<input type="text" name="nombre" placeholder="Juan Pérez" /">

- Required es un atributo booleano que el campo es obligatorio para el input que lo contien.

<input type="text" name="nombre" required />">

Uso de Modernizr y Polyfills:

Modernizr: es una librería Javascript que nos permite conocer la compatibilidad del navegador del usuario con las nuevas características de HTML5 Y CSS3. Con ello y si existen incompatibilidades, es capaz de tomar medidas par reslolver el problema. Esta medida se basan en el uso de los Plyfills.

Polyfills: son librerías Javascript que nos permiten cubrir las capacidades HTML5 y CSS3 que los navegadores no soportan.

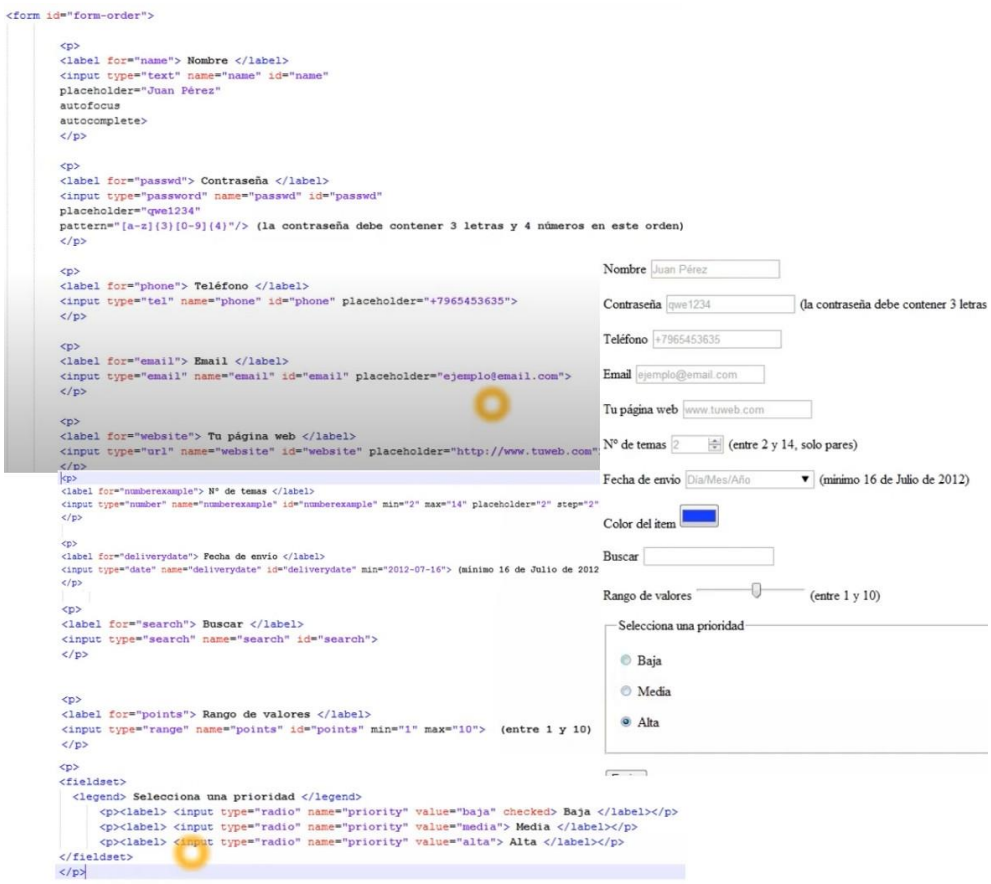
El siguiente ejemplo primero detecta si el navegador soporta la geolocalización. Si lo es se encarga carga el fichero javascript o css3 que hay en la parte del yep. Si no lo se carga el fichero especificado en el nope.

```
Modernizr.load ({
        test: Modernizr.geolocation,
        yep:'geo.js',
        nope:'geo-polyfill.js'
});
```

Urls de interés

- Descarga de Modernizr: https://modernizr.com/
- Polyfills más usados: https://github.com/Modernizr/Modernizr/wiki/HTML5-Cross-browser-Polyfills

Ejemplo Formulario



La importancia de escribir código correcto

Lógicamente es importante para evitar errores de navegación y problemas de desconfiguración.

Errores de código e incompatibilidades

- Fallos de los navegadores.
- Problemas de compatibilidad entre navegadores.
- Soporte de html para los navegadores.
- Errores en el código html

Para verificar si el código es correcto, podemos utilizar el validador del w3c: https://validator.w3.org/

Errores frecuentes

- No cerrar etiquetas.
- No cerrar comillas
- Poner “O” en vez de “0” por ejemplo en los valores de medidas de los input.

La interpretación de código cuando hay un error varía en función del navegador. En algunos casos puede que no muestre nada, en otros que no coja toda la información o que interprete otra cosa. Veamos algunos ejemplos.

- Si usamos el atributo size pero usando una "o" en vez de un "0" Explorer y Opera no tendrán en cuenta el atributo site y mostrarán el valor por defecto; por ejm 20.

- Si abrimos una etiqueta tipo <b> y no la cerramos, en este caso todos los navegadores pondrán negrita hasta el final de la página.

- Si abrimos una etiqueta tipo y no se cierra, Firefox y Chrome la cerrarán automáticamente.

- Si abrimos una etiqueta tipo <link href="hojaestilo.css" rel="stylesheet/> donde faltan las comillas al final, seguido de una etiqueta . Comprobaremos que ningún navegador reconoce el valor de la variable definida en

Por tanto**, la interpretación de los navegadores ante los errores es diferente** y esto es **debido al DOM** (Document Objet Model)

El DOM es una recomendación del W3C, que lo define como una interfaz de programación de aplicaciones (API) para documentos válidos HTML y bien construidos XML. Define la estructura lógica de los documentos y el modo en que se accede y manipula.

En DOM los documentos tienen una estructura lógica similar a un árbol. Cada documento contiene cero o un nodo doctype, un nodo de elemento de documento y cero o más comentarios o instrucciones de tratamiento; el elemento del documento sirve como la raíz del árbol para el documento. Cada navegador tiene una estructura DOM particular.

XML: HTML y XHTML

1. HTML (Hypertex Markup Language)
2. XML (Extensible Markup Language)

3. XHTML (Extensible Hypertex Markup Language)

Surge en el año 2000. Es el lenguaje de marcado que se pensó como estándar para sustituir a HTML. Incorpora a las páginas web el rigor de XML, lo cual se traduce en un mejor procesamiento, un mantenimiento más sencillo y es el primer paso hacia la llamada web semántica

- Etiquetas y atributos en minúscula.
- Última etiqueta que se abre primera que se cierra.
- Los elementos vacios deben cerrarse siempre o se escriben con una etiqueta vacía:
</br><hr></hr> o
</hr>
- Los valores de los atributos tienen que llevar siempre comillas simples o dobles: <table rows='3'>
- No está permitida la minimización de atributos: en html se podría poner <input type="radio" checked> pero en XHTML sería <input type="radio" checked="checked">

- Los espacios en blanco se eliminan en html mientras en en xml uno o más saltos de línea incluyendo los saltos se traducen en un único espacion en blanco entre palabras.
- En XHTML se utiliza el atributo id para identificar fragmento de código, mientras que en html se utiliza name: <a id="algo> <a name="algo">
- En XHTML las referencias de entidad como valores hexadecimales sólo se pueden utilizar en minúsculas: &#xnn;

Preguntas y Respuestas del examen final de certificación
http://elcuadernodenerea.blogspot.com/2018/02/examen-final-idesweb-parte-1.html