

I.E.S. DOMINGO PÉREZ MINIK



CICLO FORMATIVO DE GRADO SUPERIOR:
*"ADMINISTRACIÓN DE SISTEMAS
INFORMÁTICOS EN RED (L.O.E.)"*



MÓDULO:

Lenguajes de marcas y sistemas de
gestión de información

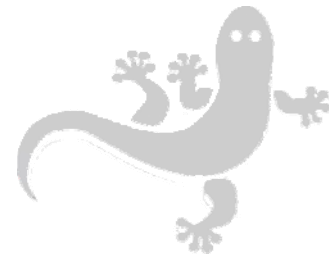
(4 horas semanales)





Índice

TEMA 3.- XHTML	3
1.- EL LENGUAJE XHTML	3
2.- ELEMENTOS EN LÍNEA FRENTE A ELEMENTOS EN BLOQUE	5
3.- RESTRICCIONES DEL XHTML	5
3.1.- XHTML es case-sensitive:	6
3.2.- El orden pasa a ser importante:	6
3.3.- Atributos siempre encerrados entre comillas:	6
3.4.- Los atributos no se comprimen:	6
3.5.- Todas la etiquetas se cierran:	7
3.6.- El atributo name es obsoleto:	7
3.7.- Otras restricciones:	7
4.- LA CABECERA XHTML, METADATOS Y DOCTYPE	8
4.1.- Estructura de la cabecera:	8
4.2.- Etiquetas meta:	9
4.3.- El título:	9
4.4.- DOCTYPE:	10
4.5.- Espacios de nombres (namespace):	11
5.- VALIDACIÓN DE DOCUMENTOS	13
6.- USO DE DIV Y SPAN	13



TEMA 3.- XHTML.

1.- El lenguaje XHTML.

XHTML es un lenguaje de marcas que ha supuesto la evolución en la creación de páginas web desde el **HTML 4.0**. Recuerda que vimos en el tema 1 que el XHTML está basado en XML

La primera versión de XHTML se denomina XHTML 1.0 y se publicó el 26 de Enero de 2000 (y posteriormente se revisó el 1 de Agosto de 2002).

XHTML, que significa Extended HTML, es un lenguaje **semántico**, lo que quiere decir que no definimos el aspecto de las cosas, sino lo que significan. Por ejemplo, si tenemos el título de nuestra página, en lugar de decir “Lo quiero grande en letras rojas”, le indicamos al navegador que “Este es el título principal de la página. Haz algo para que destaque”. Y ese “algo” lo dejamos a decisión del navegador.

Obviamente, podemos controlar el aspecto que tienen nuestras páginas, pero eso es tarea de las hojas de estilo CSS, no del XHTML.

La idea principal es que XHTML sea un lenguaje sólo para contenidos. Las cuestiones de presentación se dejarán en manos de CSS.



La segunda idea es poder utilizar ventajas propias de aplicaciones XML creando documentos perfectamente estructurados.

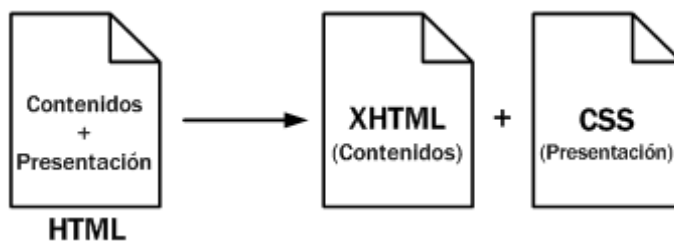


Dado que XHTML 1.0 es una adaptación de HTML 4.01 al lenguaje XML mantiene casi todas sus etiquetas y características, pero añade algunas restricciones y elementos propios de XML. De hecho XHTML contiene más o menos el 95% del HTML 4.01 que has aprendido en el tema anterior.

La versión XHTML 1.1 ya ha sido publicada en forma de borrador y pretende modularizar XHTML. También ha sido publicado el borrador de XHTML 2.0, que supondrá un cambio muy importante respecto de las anteriores versiones de XHTML.

Si quieres consultar la especificación del lenguaje oficial, publicada por el W3C, puedes obtenerla en <http://www.w3.org/TR/xhtml1/>.

Recuerda que perseguimos el objetivo de separar contenidos de presentaciones:



EJEMPLO:

Con XHTML podemos crear una página de contenidos en crudo, sin cuestiones de diseño. Y luego, usando CSS podemos adaptar el diseño de esa página para distintos medios como impresoras, pantallas, PDAs, etc.

- **Mejoras con respecto a HTML:**

- 1) Los documentos XHTML son conformes a XML. Como tales, son fácilmente visualizados, editados y validados con herramientas XML estándar.
- 2) Los documentos XHTML pueden escribirse para que funcionen igual o mejor que lo hacían antes tanto en las aplicaciones de usuario conformes a HTML 4.0 como en las nuevas aplicaciones conformes a XHTML 1.0.
- 3) Los documentos XHTML pueden usar aplicaciones (e.g. scripts y applets) que se basen ya sea en el Modelo del Objeto Documento de HTML o bien del XML.
- 4) A medida que la familia XHTML evolucione, los documentos conformes a XHTML 1.0 estarán más preparados para interactuar dentro de y entre distintos entornos XHTML.
- 5) La familia XHTML es el siguiente paso en la evolución de Internet. Al migrar en este momento hacia XHTML, los desarrolladores de contenidos web entran en el mundo de XML con todos los beneficios que se esperan de él a la vez que se aseguran la compatibilidad con aplicaciones de usuario pasadas y futuras.

2.- Elementos en línea frente a elementos en bloque.

Este es un aspecto que resultará relevante sobre todo en CSS y que no lo explicamos cuando vimos el lenguaje HTML.

El lenguaje HTML clasifica a todos los elementos en dos grupos: elementos en línea (inline elements en inglés) y elementos de bloque (block elements en inglés).

La principal diferencia entre los dos tipos de elementos es la forma en la que ocupan el espacio disponible en la página. Los elementos de bloque siempre empiezan en una nueva línea y ocupan todo el espacio disponible hasta el final de la línea, aunque sus contenidos no lleguen hasta el final de la línea. Por su parte, los elementos en línea sólo ocupan el espacio necesario para mostrar sus contenidos.

Por ejemplo, la etiqueta <P> en HTML es un elemento en bloque. Esto supone que cada vez que la utilizamos para definir un párrafo se ocupa todo el espacio hasta el final de línea. Por este motivo si utilizas dos <P> seguidos verás que se incluye ya un salto de línea.

Por su parte la etiqueta <A> es un elemento en línea. Si defines dos <A> seguidos verás que se colocan en la misma línea, uno seguido del otro.

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

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

Las siguientes etiquetas también se considera que son de bloque: dd, dt, frame-set, li, tbody, td, tfoot, th, thead, tr.

RECUERDA:

- 1) Los elementos en bloque incluyen un salto de línea final.
- 2) Los elementos en línea no contendrán dentro de ellos otro elemento en línea del mismo tipo. Ejemplo: dentro de la etiqueta <a> no habrá otra etiqueta <a>.



3.- Restricciones del XHTML.

Tú ya has aprendido a escribir código HTML 4.01. Ahora para escribir en XHTML sólo tendrás que utilizar las etiquetas descritas atendiendo a una serie de restricciones básicas:



3.1.- XHTML es case-sensitive:

Los nombres de las etiquetas y atributos siempre se escriben en minúsculas.

* Ejemplo correcto en XHTML:

`<p>Este es un párrafo con un enlace</p>`



* Ejemplo incorrecto en XHTML (pero correcto en HTML):

`<P>Este es un párrafo con un enlace</P>`

3.2.- El orden pasa a ser importante:

En XHTML será muy importante cerrar las etiquetas en orden inverso al de apertura. El HTML es más permisivo con esto, aunque en clase ya hemos pedido que se respete esta circunstancia.

* Ejemplo correcto en XHTML:

`<p>Este es un párrafo con <a>un enlace</p>`



* Ejemplo incorrecto en XHTML (pero correcto en HTML):

`<p>Este es un párrafo con <a>un enlace</p>`

3.3.- Atributos siempre encerrados entre comillas:

Ahora será obligatorio poner los valores de los atributos siempre entre comillas.

* Ejemplo correcto en XHTML:

`<p>Este es un párrafo con un enlace</p>`



* Ejemplo incorrecto en XHTML (pero correcto en HTML):

`<p>Este es un párrafo con un enlace</p>`

3.4.- Los atributos no se comprimen:

Esta característica poco frecuente en HTML. En ocasiones hay un atributo cuyo valor puede coincidir con el propio nombre del atributo. En este caso es frecuente que en HTML colocamos sólo el nombre del atributo, sin valor.

En XHTML todo atributo tiene un valor. No está permitido comprimir la escritura. Mira el siguiente ejemplo:

* Ejemplo correcto en XHTML:

`<input type="text" readonly="readonly" value="Hola">`



* Ejemplo incorrecto en XHTML (pero correcto en HTML):

`<input type="text" readonly value="Hola">`

3.5.- Todas la etiquetas se cierran:

En XHTML no puede haber ni una etiqueta que no tenga su correspondiente etiqueta de cierre. Sin embargo en HTML vimos que existen etiquetas que no llevan su cierre porque no era necesario. Es el ejemplo de `
`. Basta con poner un `
` para que se produzca un salto de línea.

Bien, en XHTML no habrá etiquetas sin su cierre. Luego podríamos hacer `
</br>`.

En este caso de etiquetas “vacías”, que no llevan nada en su interior, se define una nueva forma de escritura. Observa la norma general para estos casos:

`<etiqueta/>`



Observa que la marca de cierre se pone al final.

Ejemplos:

```
<input type="text" value="Juan"/>
<br/>
```



3.6.- El atributo name es obsoleto:

En XHTML sustituimos el atributo name por el atributo id. En todas las etiquetas donde usábamos name ahora usaremos id.

* Ejemplo correcto en XHTML:

```
<input type="text" id="nombre" value="Juan Pérez"/>
```



* Ejemplo incorrecto en XHTML (pero correcto en HTML):

```
<input type="text" name="nombre" value="Juan Pérez">
```

3.7.- Otras restricciones:

Existen dos restricciones más que están ligadas a los lenguajes de programación como el JavaScript. Dado que no hemos visto estos lenguajes las incluimos aquí para tener conocimiento de ellas.

1.- Antes de acceder al valor de un atributo, se eliminan todos los espacios en blanco que se encuentran antes y después del valor. Además, se eliminan todos los espacios en blanco sobrantes dentro del valor de un atributo. En otras palabras, si en el interior de un atributo se incluyen varios espacios en blanco seguidos, se eliminan todos salvo un único espacio en blanco utilizado para separar las diferentes palabras.

Ejemplo: `<input type="text" value=" La casa Roja">`

El valor del atributo será “La casa Roja” → eliminamos espacios.





2.- El código escrito en JavaScript, que en HTML se escribe dentro de una etiqueta `<script>`, debe encerrarse entre unas etiquetas especiales: (`<![CDATA[` y `]]>`) para evitar que el navegador interprete de forma errónea caracteres como `&` y `<`.

Ejemplo:

```
<script language="javascript">
  <![CDATA[
    alert("Hola, este código está en JavaScript");
  ]]>
</script>
```



PRACTICA 01.- Realiza una página web con sintaxis xhtml que defina un formulario para recoger datos de un alumno:

Datos personales: nombre, apellidos, dirección, código postal, isla (es un select).

Datos de contacto: teléfono, móvil, fax, email.

4.- La cabecera XHTML, metadatos y doctype.

4.1.- Estructura de la cabecera:

```
<head profile="perfil" lang="idioma" >

  <!-- zona para metadatos -->
  <meta name="author" content="Adolfo Hernández"
    ... resto de etiquetas meta ...

  <!-- zona para el título -->
  <title>Aprendiendo XHTML</title>
  ...

  <!-- zona para enlazar ficheros (CSS, javascript, RSS,...) -->

  <!-- ejemplo para enlazar hoja de estilos en CSS -->
  <link rel="stylesheet" href="estilos.css" type="text/css" media="screen" />
  <!-- ejemplo para enlazar canal de noticias en RSS -->
  <link type="application/rss+xml" title="RSS 2.0" href="canal.rss" />
  <!-- ejemplo para enlazar fichero Javascript -->
  <script src="micodigo.js" type="text/javascript"></script>
</head>
```



Como ves seguimos manteniendo la misma estructura que en HTML. En la etiqueta head puedes definir el atributo profile que es muy poco utilizado y especifica una URL donde hay un documento que aclara que especifica más información sobre las

etiquetas meta. Por su parte el atributo lang especifica el idioma en que está escrita esta página. Recuerda que ambos atributos son opcionales.

4.2.- Etiquetas meta:

Recuerda que las puedes definir como quieras. Siempre se coloca un nombre con el atributo **name** y luego una descripción de la etiqueta con el atributo **content**. En XHTML estas etiquetas mantienen el atributo name (recuerda que dijimos que en el resto usaríamos el atributo id en su lugar). En algunas ocasiones en lugar de usar el atributo name se puede usar el atributo **http-equiv** que puede ser usado por los servidores web para conocer aspectos generales de la página, como la codificación de caracteres utilizada. Ejemplo:

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

Esta etiqueta informa al servidor que la página está codificada en UTF-8.

Hay una serie de metadatos que se utilizan tanto que han pasado casi a ser un estándar usado por los buscadores web. Mira los siguientes ejemplos:

* Definir el autor del documento:

```
<meta name="author" content="Juan Pérez" />
```



* Definir el programa con el que se ha creado el documento:

```
<meta name="generator" content="WordPress 2.8.4" />
```

* Definir la codificación de caracteres del documento:

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

```
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1" />
```

* Definir el copyright del documento:

```
<meta name="copyright" content="librosweb.es" />
```

* Definir el comportamiento de los buscadores:

```
<meta name="robots" content="index, follow" />
```

* Definir las palabras clave que definen el contenido del documento:

```
<meta name="keywords" content="xhtml, web, programacion" />
```

* Definir una breve descripción del sitio:

```
<meta name="description" content="página para aprender XHTML" />
```

4.3.- El título:

En XHTML es obligatorio que cada página tenga su etiqueta title. Y además muchos buscadores utilizan el contenido de esta etiqueta para realizar su búsquedas, con lo que es importante que el título sea descriptivo. En cada página pon un título corto, único y descriptivo sólo de esta página.



4.4.- DOCTYPE:

El estándar XHTML deriva de XML, por lo que comparte con él muchas de sus normas y sintaxis. Uno de los conceptos fundamentales de XML es la utilización del DTD o Document Type Definition ("Definición del Tipo de Documento").

Un DTD es un documento que recoge el conjunto de normas y restricciones que deben cumplir los documentos de un determinado tipo **para estar bien escrito y ser un documento válido**. Veremos la potencia de los DTD cuando trabajemos con XML.

El conjunto de normas, obligaciones y restricciones que se deben seguir al crear un documento de un determinado tipo, se recogen en su correspondiente DTD. El estándar XHTML define el DTD que deben seguir las páginas y documentos XHTML.

En este documento se definen las etiquetas que se pueden utilizar, los atributos de cada etiqueta y el tipo de valores que puede tener cada atributo.

En realidad, la versión 1.0 del estándar de XHTML define tres DTD diferentes. Para indicar el DTD utilizado al crear una determinada página, se emplea una etiqueta especial llamada **doctype**. La etiqueta doctype es el único elemento que se incluye fuera de la etiqueta <html> De hecho, la declaración del doctype es lo primero que se debe incluir en una página web, antes incluso que la etiqueta <html>.

Como se verá más adelante, para que una página XHTML sea correcta y válida es imprescindible que incluya el correspondiente doctype que indica el DTD utilizado.

A continuación se muestran los tres DTD que se pueden utilizar al crear páginas XHTML:

XHTML 1.0 Estricto:

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



Se trata de la variante con las normas más estrictas y las restricciones más severas. Las páginas web que incluyan este doctype, no pueden utilizar atributos relacionados con el aspecto de los contenidos, por lo que requiere una separación total de código HTML y estilos CSS.

XHTML 1.0 Transitorio:

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



Se trata de una variante menos estricta que la anterior, ya que permite el uso de algunos atributos HTML relacionados con el aspecto de los elementos.

XHTML 1.0 Frameset:

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



Esta última variante la utilizan las páginas que están formadas por marcos, una práctica completamente desaconsejada y que hoy en día sólo utilizan los sitios web obsoletos.

Si no tienes claro el DTD que más te conviene, deberías utilizar el XHTML 1.0 transitorio, ya que es más fácil crear páginas web válidas. Si tienes conocimientos avanzados de XHTML, puedes utilizar XHTML 1.0 estricto.

4.5.- Espacios de nombres (namespace):

XHTML proviene del XML. Pero XML soporta un conjunto de tecnologías o lenguajes derivados, y no únicamente el XHTML. Muchos de estos lenguajes derivados tienen sintaxis similar, etiquetas similares, etc. Y lo que es peor, pueden utilizarse combinados en una misma página web.

Es necesario por tanto que cuando el navegador encuentre una etiqueta sepa exactamente a qué lenguaje nos estamos refiriendo.

Un **namespace** en un documento XML permite diferenciar las etiquetas y atributos que pertenecen a cada lenguaje. Si en un mismo documento se mezclan etiquetas de dos o más lenguajes derivados de XML (XHTML y SVG por ejemplo) y que tienen el mismo nombre, no se podría determinar a qué lenguaje pertenece cada etiqueta y por tanto, no se podría interpretar esa etiqueta o ese atributo.

Los namespaces se indican mediante una URL. El namespace que utilizan todas las páginas XHTML (independientemente de la versión y del DOCTYPE) es <http://www.w3.org/1999/xhtml> y se indica de la siguiente manera:

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



De esta forma, es habitual que las páginas XHTML comiencen con el siguiente código:

```
<!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" lang="es" xml:lang="es">

... contenido de la página ...

</html>
```



Observa que agregamos el lenguaje de escritura de la página con `xml:lang="es"`.



Aunque el código anterior es mucho más complicado que una simple etiqueta `<html>`, es imprescindible para que las páginas XHTML creadas sean correctas y superen satisfactoriamente **el proceso de validación** que veremos en el siguiente apartado.

Afortunadamente, si utilizas un editor avanzado como Dreamweaver para crear las páginas, todo el código anterior se incluye de forma automática. Si creas las páginas a mano, sólo tienes que copiar y pegar ese código en cada nueva página.



PRACTICA 02.- Con la práctica anterior agrega una cabecera:

- Se trata de código xhtml transicional.
- Debes incluir metainformación con el autor de la página.
- Agrega el namespace por defecto para xhtml




PRACTICA 03.- Realiza una página como la siguiente:

- Utiliza una etiqueta `<fieldset>` para la parte de Datos básicos y otra para la de Datos Económicos.
- Utiliza etiqueta `<legend>` para el texto “Datos básicos” y “Datos económicos”.
- Utiliza etiquetas `<label>` para el resto de textos antes de un control de formulario.

Busca la información necesaria para usar `<fieldset>`, `<legend>` y `<label>`

5.- Validación de documentos.

La validación es el proceso que asegura que un documento escrito en un determinado lenguaje cumple con las normas y restricciones de ese lenguaje.

La **validación** no es obligatoria. Una página puede verse bien sin ser válida 

No obstante es **recomendable** que nuestras páginas sean válidas, ya que eso asegura que se verán correctamente en cualquier navegador.

Hay programas de edición de páginas web que incluyen sus propios validadores. Por ejemplo el Dreamweaver tiene un validador propio, que puede configurarse en el menú Edición → preferencias → validador.

Una vez definido el validador que se aplicará en el menú preferencias, escribimos nuestro código, y a continuación en el menú archivo podemos escoger la opción **validar** para obtener una lista de posibles errores y advertencias.

La **validación** consiste en utilizar el DTD definido en la etiqueta DOCTYPE. 

Por otra parte el W3C ha creado una herramienta gratuita para la validación de las páginas. Para utilizarlo sólo tienes que abrir la página web <http://validator.w3.org/>

- Si la página web está publicada en un servidor web utilizar la pestaña “validate by URI”.
- Si la página es un fichero .html que tienes en tu equipo utiliza “validate by upload”.
- Si lo que quieres es validar un único trozo de código utiliza “validate by input”.

Existen muchos otros validadores. Uno de los mejores es el que incluye la herramienta de diseño web **Dreamweaver**. Incluye validación para HTML y para xhtml, se puede configurar en el menú edición → preferencias → validador.



PRACTICA 04.- Valida el ejercicio anterior de la página del formulario con el validador de la w3c. Revisa y corrige los errores hasta que la validación sea perfecta.

6.- Uso de DIV y SPAN.

DIV y SPAN son dos etiquetas que ya existían en HTML pero las dejamos para este tema porque con XHTML su uso se ha incrementado de forma notable. Como ya habrás visto para posicionar elementos en la pantalla con HTML era necesario crear



tablas sin borde. Para hacer páginas complejas era necesario anidar muchas tablas dentro de otras, haciendo que el mantenimiento de la página se volviera complicado.

Div significa en inglés *Division* y es una etiqueta que se usa para establecer una región de la página web. Imagina que es como si definieras una caja dentro del documento. Dentro de esa caja podrás colocar todos los elementos que desees.

Es frecuente que a estos DIV se les llame “Capas”, dado que más adelante usando CSS verás que es posible colocar unos div sobre otros, ocultar y mostrar algunos, etc. No obstante, para realizar capas es necesario usar CSS, y en concreto una propiedad llamada z-index.

Actualmente las páginas web profesionales creadas con XHTML suelen estar formadas por multitud de etiquetas <div>. Aunque esta etiqueta soporta atributos básicos como align, valign, etc, los más utilizados son los siguientes:

```
<div id="identificador" class="estilo en CSS">  
... contenido de la región ....  
</div>
```



Una vez presentada la etiqueta div veremos que más adelante, utilizando CSS, es donde le sacaremos partido a la misma. No obstante verás que muchas páginas profesionales definen una serie de div básicos para referirse a zonas de la página. Ejemplos de estas regiones son:

- contenedor (wrapper) suele encerrar la mayor parte de los contenidos de la página y se emplea para definir las características básicas de la página: su anchura, sus bordes, imágenes laterales, si se centra o no respecto de la ventana del navegador, etc.
- cabecera (header) que incluye todos los elementos invariantes de la parte superior de la página (logotipo, imagen o banner, cuadro de búsqueda superior, etc.)
- contenido (content) engloba el contenido principal del sitio (la zona de noticias, la zona de artículos, la zona de productos, etc. dependiendo del tipo de sitio web).
- menu (menu) se emplea para agrupar todos los elementos del menú lateral de navegación de la página.
- pie (footer) que incluye todos los elementos invariantes de la parte inferior de la página (aviso de copyright, política de privacidad, términos de uso, etc.).
- lateral (sidebar) se emplea para agrupar los elementos de las columnas laterales y secundarias de la página.

OJO: esto es sólo una propuesta general. En tus páginas podrás crear los div que quieras con los nombres que quieras.

La etiqueta span tiene la misma forma y atributos que div:

```
<span id="identificador" class="estilo en CSS">  
... contenido de la región ....  
</span>
```



No hay, por tanto diferencias. La única cuestión a tener en cuenta es que span es una región pequeña, habitualmente de unos pocos caracteres o una única línea.



PRACTICA 05.- Instalación y pruebas del validador de Dreamweaver.

I.E.S. DOMINGO PÉREZ MINIK



CICLO FORMATIVO DE GRADO SUPERIOR:
*"ADMINISTRACIÓN DE SISTEMAS
INFORMÁTICOS EN RED (L.O.E.)"*



MÓDULO:

Lenguajes de marcas y sistemas de
gestión de información

(4 horas semanales)