Escuela Politécnica de Ingeniería
Grado de Ingeniería Informática en Tecnologías de la Información

Tecnologías Web

Tema 2

Tecnologías web de cliente

Índice

- HTML4/HTML5
- Diferencias entre xhtml y HTML 4
- Diferencias entre HTML 5 y HTML 4
- Estructura básica de un documento XHTML

HTML (1)

- HTML (***H**yper**T**ext **M**arkup **L**aguage*).
  - Es el principal lenguaje de marcado para la creación de páginas web que se pueden visualizar en un cliente web (p.e., en un un navegador).
  - Permite **describir la estructura y la información** de una página web en forma de texto. Tipo MIME: **`text/html`**.
  - Recomendaciones (últimas):
    - HTML (24-12-1999): HTML 4.01 Specification
    - XHTML (23-11-2010): XHTML 1.1 – Module-based XHTML – Second Edition
    - ¿ HTML 5 ?
      - Previsto, inicialmente, para el año 2022.
      - A finales de septiembre de 2012, el W3C anunció que habría varias versiones de HTML 5 y que para la primera, HTML 5.0, se dispondría de una recomendación a finales de 2014. La siguiente versión, HTML 5.1, dispondrá de recomendación a finales del 2016.

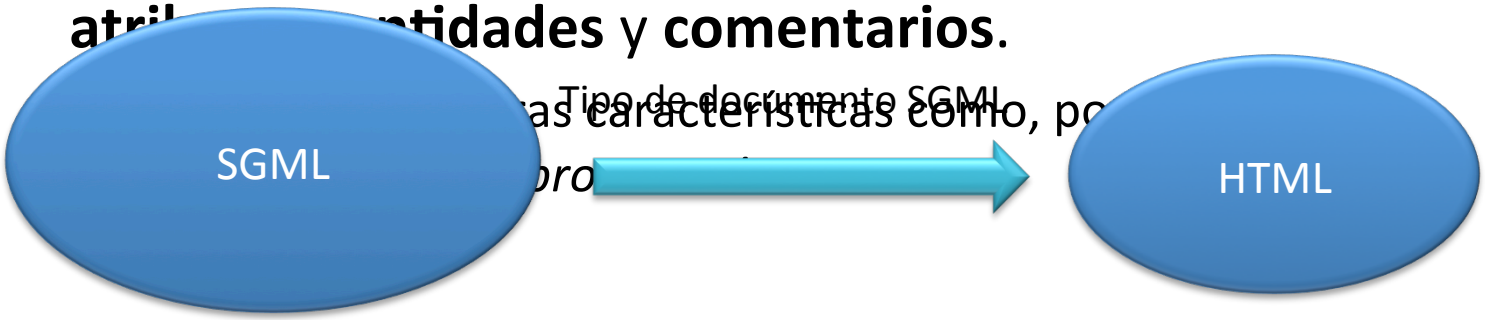
HTML (2)

- Antecedentes

Año	Versión
1990	Diseño del HTML
1991	HTML
1994	HTML 2
1995	HTML 3 (borrador)
1997	HTML 3.2
1998	HTML 4 (recomendación)
2000	XHTML 1 (recomendación)
2003	XHTML 2.0 (borrador, continuado en HTML 5)
2004	Web Applications 1.0 (HTML 5) WhatWG
2008	HTML 5 (borrador)

HTML (3)

- Basado en SGML (*Standard Generalized Markup Language*)
  - Un sistema que permite definir lenguajes para dar formatos a documentos.
    - Permite representar **información estructural, de presentación** y **semántica** junto con el **contenido**.
  - Estructuras de SGML utilizadas en HTML: **elementos**, **atri** **ntidades** y **comentarios**.
  - as características como, po
    - Tipo de documento SGML
    - *oro*



Componentes de HTML (1)

- **Elementos**
  - Representan estructuras o un comportamiento deseado. HTML incluye elementos que representan párrafos (`p`), cabeceras (o títulos, `h1` a `h6`), listas (`ul`, `ol`, `dl`), imágenes (`img`), tablas (`table`, `tr`, `td`), etc.
  - Habitualmente constan de tres partes: **etiqueta de inicio**, **contenido** y **etiqueta de fin**.
    - Sintaxis: **`<`*`nomb_elemento`*`>`** `contenido` **`</`*`nomb_elemento`*`>`**
    - Ejemplo: `<p>Esto es un párrafo.</p>`
  - Ciertos elementos carecen de contenido (por ejemplo, la representación de un salto de línea). Éstos constan de una única etiqueta de inicio y fin.
    - Ejemplo: `<br>`

Componentes de HTML (2)

- **Atributos**
  - Son propiedades asociadas a los elementos, a los que se les asigna (mediante el símbolo de igualdad, =) un valor delimitado por comillas dobles (o simples).
    - Los atributos se asignan en la etiqueta de inicio del elemento y pueden ser opcionales u obligatorios.
    - Sintaxis: <*nomb_elemento* ***atr1="v1" atr2="v2"*** …>
    - Ejemplo: `<img src="logo.jpg" alt="Logo de empresa">`
- **Entidades**
  - Son nombres simbólicos o numéricos. Referencian caracteres no imprimibles o utilizados de forma característica en HTML (como, por ejemplo, '<' y '>').
    - Sintaxis: ***&nombre; &#código_ascii; &#xcódigo_ascii;***
    - Ejemplos: '<' → `<` '>' → `>` '&' → `&` '©' → `&xA9;`

Componentes de HTML (3)

- **Comentarios**
  - Se utilizan para incorporar contenido que no tiene significado especial para HTML.
    - En los comentarios las etiquetas de elementos o las entidades no se interpretan como tales.
    - Pueden utilizarse con fines de documentación o para que el agente ignore otros contenidos que no son de información o HTML (como, por ejemplo, *scripts* o reglas de estilo).
    - Sintaxis: **`<!-- comentario ->`**

Estructura global de un documento HTML (1)

- La estructura global de un documento HTML consta de:
  1. La **definición del tipo de documento** (**DTD**).
     - Es la descripción de la estructura y sintaxis de un documento SGML.
       - *Describe los elementos*: etiquetas y tipo de contenido.
       - *Define los atributos de los elementos*: nombre, valores que pueden tomar y si éstos son opcionales u obligatorios.
       - *Establece cómo se pueden anidar los elementos y el orden* en que aparecen en el documento.
     - HTML 4.01 dispone de tres tipos de DTD, en un primer intento (fallido) de separar la *información de presentación* de la *información de la estructura* y *contenido* del documento: *transitional*, *frameset* y ***strict***.

Estructura global de un documento HTML (2)

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

Sólo utilizaremos la forma estricta

- XHTML (a partir de la versión 1.1) y HTML 5 [illegible] elementos y atributos de presentación. Así, la *información de presentación* queda completamente separada de la *información de la estructura* y *contenido* del documento. Habitualmente la información de presentación se incluye en un segundo documento: *hojas de estilo en cascada* (CSS).

Estructura global de un documento HTML (3)

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN""http://
www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
```

- HTML 5 no está basado en SGML por lo que la declaración DOCTYPE es más corta

```
<!DOCTYPE html>
```

- Para otros tipos de documentos puede consultarse la DTD recomendada en la URI:

  http://www.w3.org/QA/2002/04/valid-dtd-list.html

Estructura global de un documento HTML (4)

2. **Elemento raíz**: `html`
   - Contiene la **cabecera** (elemento head) y el **cuerpo** (elemento body) del documento.
4. **Cabecera**. Elemento head
   - Su contenido representa **información adicional sobre el documento**: su título, metadatos o documentos relacionados (por ejemplo, de *scripts* u hojas de estilo).
     - Como mínimo consta del título del documento, el elemento `title`.
     - Entre los posibles metadatos (autor, fecha de modificación, etc.) que se especifican con el elemento sin contenido `meta`, es especialmente conveniente (aunque no obligatorio) establecer la codificación de caracteres.
5. **Cuerpo**. Elemento body
   - Su contenido es la **descripción de la estructura** y **la información** que proporciona el documento.

Estructura global de un documento HTML (5)

Estructura global del documento

HTML 4.01	HTML 5
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN" "http://www.w3.org/TR/html4/strict.dtd"> <html> <head> <meta http-equiv="Content-Type" content="text/html; charset=utf-8"> <title>Título</title> <link rel="stylesheet" href="form.css" type="text/css" media="screeen"> </head> <body> . . . </body> </html>	<!DOCTYPE html> <html> <head> <meta charset="utf-8"> <title>Título</title> <link rel="stylesheet“ href="form.css“ type="text/css“ media="screeen"> </head> <body> . . . </body> </html>

Organización de elementos (1)

- Elementos de nivel superior
  - `html`, `head` y `body`
- Elementos de cabecera (`head`)
  - `link`, `meta`, `script`, `style` y `title`
- Elementos de cuerpo (`body`)
  - Elementos de bloque
    - Pueden contener otros elementos de bloque (con excepciones, como por ejemplo los párrafos), elementos de línea y texto.
    - `div`, `h1` a `h6`, `hr`, `noscript`, `p` y `pre`
  - Elementos en línea
    - Pueden contener otros elementos de línea y texto.
    - `br`, `a`, `img`, `map`, `area`, `script` y `span`

Organización de elementos (2)

- Listas
  - Desordenadas: `ul`, `li`
  - Ordenadas: `ol`, `li`
  - Descriptivas: `dl`, `dt`, `dd`
- Tablas
  - Elemento `table`
  - Descripción de la tabla: `caption`
  - Estructura de la tabla: `thead`, `tfoot` y `tbody`
    - Filas: `tr`
    - Celdas: `th` y `td`
  - Otros: `colgroup` y `col`
- Formularios
  - Elemento `form`
  - Agrupación de campos: `fieldset` y `legend`
  - Etiquetas de campos: `label`
  - Campos: `input`, `select`, `option`, `button` y `textarea`

Atributos principales

- Los atributos principales se pueden aplicar a la mayoría de los elementos.
  - `id`: permite asignar un identificador único a un elemento.
  - `class`: permite asignar una clase (o un conjunto de clases) a un elemento. A los elementos de una clase se les asocia la misma información de presentación mediante reglas de estilo.
  - `title`: permite aplicar un título a un elemento.
  - `style`: permite aplicar un estilo a un elemento.
  - `lang`: permite especificar el idioma de los valores de los atributos y del texto contenido en el elemento.
  - `dir`: permite especificar la dirección del texto.

Atributos de teclado

- Se pueden aplicar a los elementos `a`, `area` y campos de formulario.
  - `accesskey`: permite asignar una tecla de acceso rápido a un elemento.
  - `tabindex`: permite asignar un orden de tabulación a un elemento.

Eventos (1)

- Los eventos son atributos que permiten agregar interactividad. El valor del atributo es un *script* que se ejecuta ante determinas actuaciones sobre el elemento.
  - Eventos de formulario
    - `onfocus`: ocurre cuando un elemento recibe el enfoque.
    - `onblur`: ocurre cuando un elemento pierde el enfoque.
    - `onsubmit`: ocurre cuando se envía un formulario.
    - `onreset`: ocurre cuando se reestablece un formulario.
    - `onselect`: ocurre cuando se selecciona un texto en un campo de texto (`input` o `textarea`).
    - `onchange`: ocurre cuando el control pierde el enfoque y su valor ha sido modificado desde que recibió el enfoque por última vez.

Eventos (2)

- Eventos de ventana (sólo para el elemento body)
  - `onload`: ocurre cuando el agente de usuario termina de cargar.
  - `onunload`: ocurre cuando el agente de usuario retira el documento.
- Eventos de teclado. Éstos se producen si el elemento tiene el enfoque.
  - `onkeypress`: ocurre al presionar y soltar una tecla.
  - `onkeydown`: ocurre al presionar una tecla.
  - `onkeyup`: ocurre al soltar una tecla.

Eventos (3)

- Eventos de ratón
  - `onclick`: ocurre cuando se realiza un clic sobre el elemento.
  - `ondblclick`: ocurre cuando se realiza un doble clic sobre el elemento.
  - `onmousedown`: ocurre cuando el botón del ratón se presiona sobre el elemento.
  - `onmouseup`: ocurre cuando el botón del ratón se suelta sobre el elemento.
  - `onmouseover`: ocurre cuando el puntero del ratón se pone sobre el elemento.
  - `onmouseout`: ocurre cuando el puntero del ratón se quita del elemento.

Diferencias entre xhtml y HTML4

- En HTML4 no importan las mayúsculas/minúsculas en etiquetas, atributos y valores predefinidos de atributos
  - <BODY>, <Body> y <body> son equivalentes
  - <H1 ALIGN="…"> es equivalente a <H1 aLiGn="…">
  - <INPUT TYPE="TEXT"> es equivalente a <INPUT TYPE="text">
- En xhtml, se usan minúsculas para las etiquetas, atributos y valores predefinidos
  - <body>
  - <h1 align="…">
  - <input type="text"/>

Xhtml: uso de comillas

- En HTML 4, las comillas son opcionales si el atributo contiene sólo valores alfanuméricos
  - <H1 ALIGN="LEFT"> o <H1 ALIGN=LEFT>
- En xhtml, es obligatorio el empleo de comillas simples o dobles
  - <h1 align="left"> o <h1 align='left'>

Xhtml: Etiquetas de finalización

- HTML 4
  - Algunas etiquetas son contenedores
    - <H1>...</H1>, <A HREF...>...</A>
  - Otras son independientes
    -
, <HR>
  - Algunas etiquetas tiene finalizadores opcionales
    - <P>, <LI>, <TR>, <TD>, <TH>
- XHTML
  - Todas las etiquetas son contenedores. **El uso de fin es obligatorio**
    - ... , <li>...
  - Si la etiqueta no tiene contenido, pueden fusionarse las etiquetas inicio/fin
    -
</br> →
    - – Un bug de IE previene este uso para evitar <script/>. Se recomienda usar siempre <script ...></script>.
  - Algunas personas usan
 (espacio después del /) por compatiblidad con los navegadores muy antiguos. Pero estos navegadores no soportan XMLHttpRequest, de modo que no tendrá importancia en aplicaciones Ajax.

Xhtml: Atributos booleanos

- En HTML 4, se puede simplicar la notación para atributos booleanos
  - <option value="1" selected>...</option>
  - <dlcompact>–<dl compact>
- En xhtml, se deben escribir los valores de los atributos obligatoriamente:
  - <option value="1" selected="selected">...</option>
  - <dl compact="compact">

Formato xhtml

- Formato mínimo:

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

- Formato habitual

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN”
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="content-type" content="text/html; charset=UTF-8" />
<title>Some Title</title></head>
<body>...</body>
</html>
```

Los DTDs de xhtml

- Transicional
  - Se admiten etiquetas de formato no-CSS como <font> y <i>
  - <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd>
- Stricto
  - No se admiten etiquetas de formato no-CSS
    - –<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
    - "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd>
- Frames
  - Para la página padre que use frames
  - <!DOCTYPE html PUBLIC //W3C//DTD XHTML 1.0 Frameset//EN "http://www.w3.org/TR/xhtml1/DTD/xhtml1-frameset.dtd">
  - Téngase en cuenta que las páginas que constituyen los frams no usarán este DTD sólo la página padre de los frames.

Los elementos <head> y <body>

- Head
  - Debe contener obligatoriamente <title>
  - Y puede contener: <meta>, <script>, <style> y <base>
- body
  - Contiene la parte que se visualiza directamente en la ventana.
  - Attributos
    - bgcolor, backgrond, text, link, vlink y alink
      - E.g.: <body bgcolor="blue">
      - Normalmente se usan hojas de estilo en vez de atributos
      - onload, onunload, onfocus, onblur
        - » Para el manejo de eventos con JavaScript

HTML 5

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="utf-8"/>
<link href="css/some-stylesheet.css"
rel= stylesheet />
<script src="scripts/some-script.js"></script>
</head>
<body>
...
</body>
</html>
```

Introducción I

- HTML5 es un proyecto de cooperación entre World Wide Web Consortium (W3C) y el Web Hypertext Application Technology Working Group (WHATWG).
- WHATWG ha trabajado con los formularios web y las aplicaciones, y el W3C con XHTML 2.0. En 2006, decidieron cooperar para crear una nueva versión de HTML.
- Las nuevas reglas de HTML5 son:
  - Las nuevas características deben estar basadas en HTML, CSS, DOM, y JavaScript
  - Redución de la necesidad de plugins externos (como Flash)
  - Mejor manejo de los errores
  - Más etiquetas para sustituir los scripts
  - HTML5 debe ser independiente de los dispositivos
  - El proceso de desarrollo debe ser visible al públicos

Introducción II

- Nuevas características en HTML5
  - La etiqueta <canvas> para dibujo 2D
  - Las etiquetas <video> y <audio> para reproducción de medios
  - Soporte para almacenamiento local
  - Nuevas etiquetas de contenido específico, como <article>, <footer>, <header>, <nav>, <section>
  - Nuevos controles, como calendar, date, time, email, url, search

HTML5. Nuevos elementos (I)

- <canvas>. Empleado para dibujar «on the fly», empleando Javascript.
- Elementos media
  - <audio>. Define un contenido de tipo musical.
  - <video>. Define un contenido de tipo video.
  - <source>. Define fuentes de medios para ser usado en <video> y <audio>
  - <embed>. Define un contenedor para una aplicación externa o contenido interactivo (un plug-in)
  - <track>. Define un pista de testo para un elemento <video> o <audio>
- Elementos de formulario
  - <datalist>. Especifica una lista de opciones predefinidas para controles de de tipo ìnput.
  - <keygen>. Define un campo generador de claves (para formularios).
  - <output>. Define el resultado de un calculo.

HTML5. Nuevos elementos (2)

- Nuevos elementos semánticos
  - Los más relevantes se han introducido para reducir el uso abusivo de bloques `div` anidados. Práctica ésta que se ha extendido de forma habitual, aunque no siempre justificada, en HTML 4.01.
    - **`section`**. Representa una sección general dentro de un documento.
    - **`nav`**. Representa una sección dedicada a la navegación del sitio.
    - **`article`**. Representa un contenido independiente en un documento.
    - **`aside`**. Representa un contenido que está poco relacionado con el resto de la página.
    - **`header`**. Representa un grupo de artículos introductorios o de navegación.

HTML5. Nuevos elementos(3)

- **`footer`**. Representa el pie de una sección con información acerca de la página y no relacionada con su contenido.

– El abuso del elemento `div` en HTML 4 ha provocado que muchos documentos existentes en la web hayan perdido la *información estructural* (o prácticamente carezcan de ésta) al ser el div un elemento de carácter genérico.

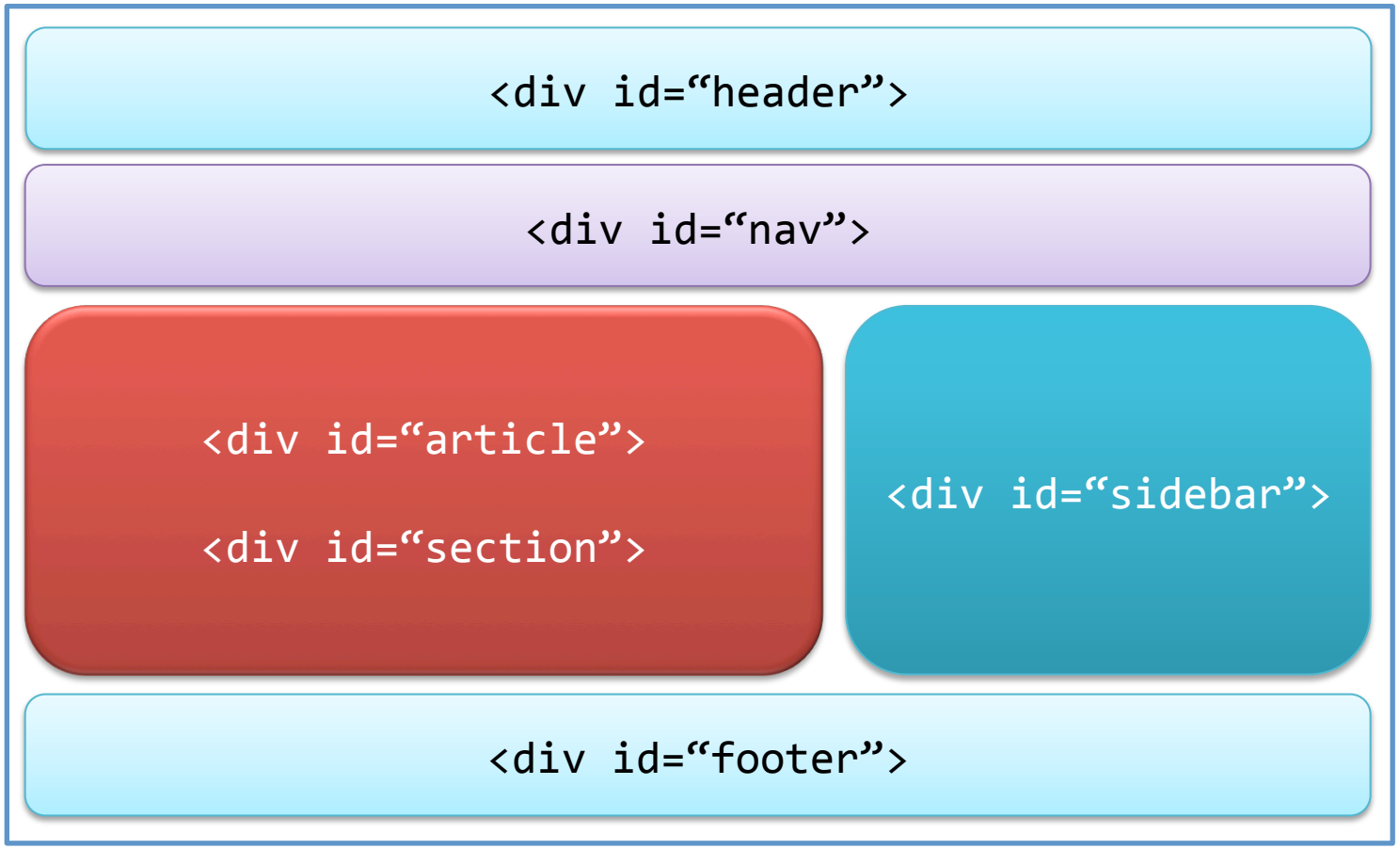
- Un buen documento de HTML 5 debe utilizar los nuevos elementos semánticos en lugar de elementos div.

Nuevos elementos en HTML5 (2)

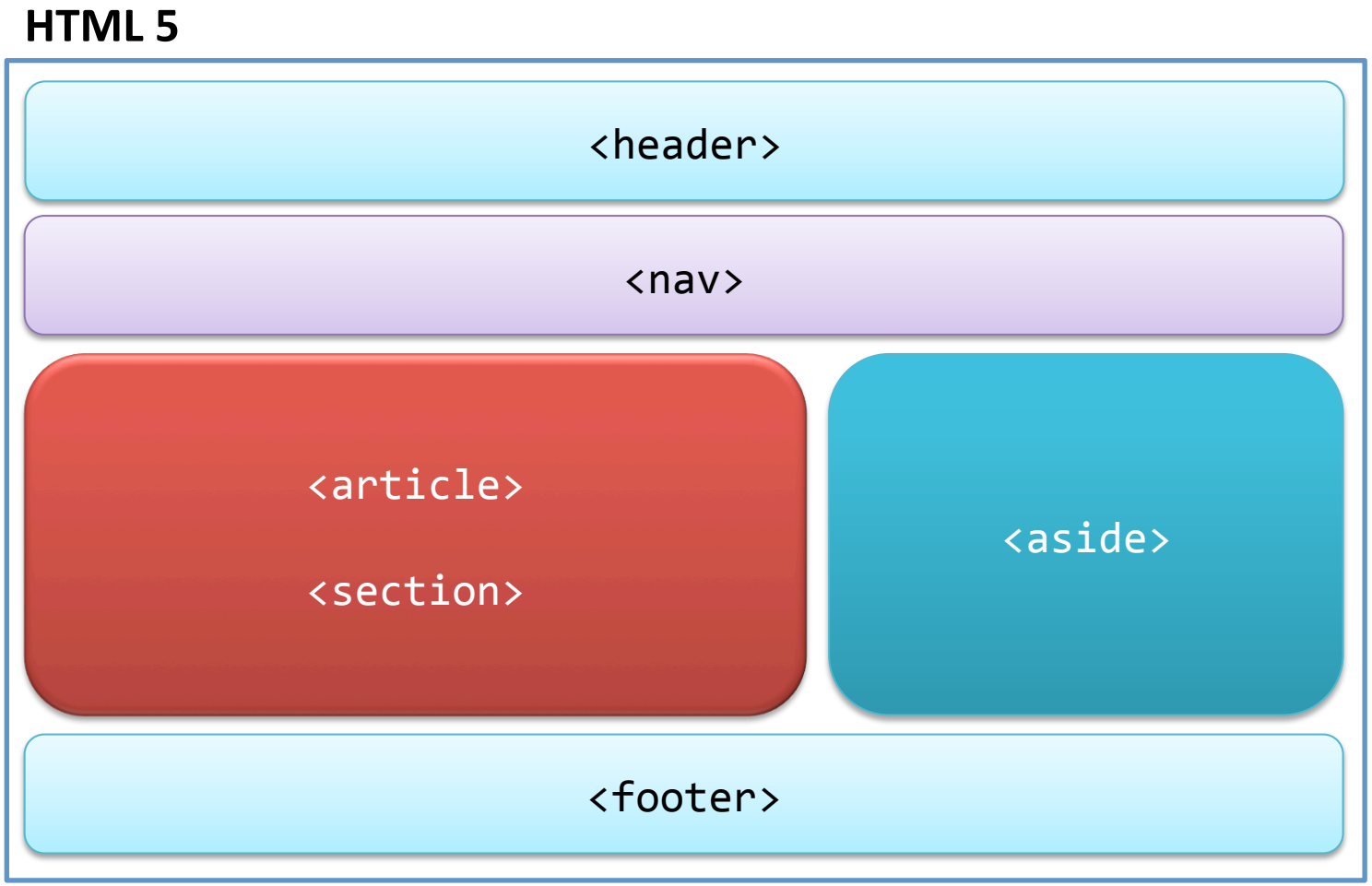
- Otros elementos
  - **`main`**. Representa el contenido principal del cuerpo de un documento o aplicación.
  - **`figure`**. Representa algún contenido de flujo con un subtítulo (`figcaption`) opcional. Puede contener ilustraciones, diagramas, tablas, código, etc.
  - **`embed`**. Representa un contenedor para una aplicación externa (típicamente no HTML) o contenido interactivo.

HTML5. Nuevos elementos(4)

HTML 4



HTML5. Nuevos elementos(5)



HTML 5. Nuevos atributos globales

- Nuevos atributos globales.
  - Se pueden aplicar a cualquier elemento.
    - **contextmenu**. Permite especificar un menú contextual para un elemento (éste se muestra al pulsar el botón derecho del ratón sobre el elemento).
    - **contenteditable**. Especifica cuando el contenido de un elemento es editable.
    - **draggable**. Especifica cuando un elemento se puede arrastrar o no.
    - **dropzone**. Especifica si los datos arrastrados se copian, mueven o vinculan
- No soportados por ningún navegador comercial a día de hoy

Nuevos elementos en HTML5 (II)

- Elementos estructurales y semánticos
  - <article> . Define un artículo.
  - <aside> . Define un contenido a parte de la página.
  - <bdi> . Aisla una parte de texto que puede ser formateada en una dirección diferente a otro texto externo a este.
  - <command> . Define un botón command que un usuario puede invocar.
  - <details> . Define detalles adicionales que un usuario puede visualizar u ocultar.
  - <dialog> . Define una caja de diálogo o ventana.
  - <summary> . Define un cabecera visible para un elemento <details>
  - <figure> . Espeifica un elemento autocontenido, como una ilustración, diagrama, foto, listas de códigos, etc.
  - <figcaption> . Define un título para un elemento <figure>.
  - <footer> . Define un pie para un documento o sección.
  - <header> . Define una cabecera para un documento o sección.
  - <mark> . Define un texto resultado.
  - <meter> . Define una medida escalar en un rango conocido (un calibre)
  - <nav> . Define enlaces de navegación.
  - <progress> . Representa el progreso de una tarea.
  - <ruby> . Define una anotación ruby (para tipografías del este asiático)
  - <rt> . Define una explicación/pronunciación de caracteres (Defines an explanation/pronunciation of characters (para tipografías del este asiático)
  - <rp> . Define que se muestra en los navegadores que no soportan anotaciones ruby.
  - <section> . Define una sección en un documento.
  - <time> . Define un fecha/hora.
  - <wbr> . Define un posible salto de linea.

HTML 5. Nuevos eventos

- Como resultado de la aparición de las nuevas características multimedia, elementos, atributos y características de edición y *drag & drop*, en HTML 5 se han incrementado de forma notable los eventos.
  - Fundamentalmente eventos de ventana, de ratón y de interacción con elementos multimedia.
  - También aparecen algunos nuevos eventos de formulario.

HTML 5. Elementos eliminados de HTML 4

- <acronym>
- <applet>
- <basefont>
- <big>
-
- <dir>
- <font>
- <frame>
- <frameset>
- <noframes>
- <strike>
- <tt>

HTML 5. formularios (1)

- Nuevos tipos input
  - HTML4: `text`, `password`, `checkbox`, `radio`, `submit`, `reset`, `file`, `hidden`, `image`, `button`.
  - HTML5: `email`, `url`, `date`, `time`, `datetime`, `month`, `week`, `number`, `range`, `tel`, `search`, `color`.
- Nuevos atributos
  - **`autofocus`**. Indica que elemento de formulario debe ganar el foco al cargar una página.
  - **`placeholder`**. Permite ubicar en un campo un texto informativo para el usuario que se elimina al ganar este el foco y que se restablece cuando el campo pierde el foco y está vacío.
  - **`required`**. Para indicar campos obligatorios.

HTML 5. formularios (2)

  - **`pattern`**. Permite especificar una expresión regular para validar la entrada de un campo input.
  - **`min`**, **`max`**, **`step`**. Permiten especificar un rango de valores y la granularidad de la entrada de un campo input.
  - **`form`**. Indica el formulario al que pertenece un control y permite poner un elemento de formulario en cualquier parte de una página.

- Nuevos elementos de formulario
  - **`datalist`**. Especifica una lista de opciones predefinidas para un control `input`.
  - **`output`**. Define el resultado de un cálculo.
  - **`keygen`**. Especifica un campo generador de pares de llaves. La llave privada se almacena localmente y la llave pública se envía al servidor.
  - **`progress`**. Representa el progreso de finalización de una tarea.

HTML5. formularios. Nuevos inputs (3)

- No todos los navegadores soportan los nuevos campos. En caso de no ser soportados se comportarán como campos de texto normales.
- Algunos ejemplos:
  - time
  - Date
  - email
  - Range
  - url

HTML5. formularios. Nuevos elementos (4)

- <datalist>
  - Especifica una lista de opciones predefinidas para un elemento <input>.
  - Se emplea para proveer de la característica de autocompletado a elementos <input> (en forma de control desplegable)
  - Se debe emplear el atributo “list” del elemento <input> para conectarlo con el elemento <datalist>.

```
<input list="browsers">

<datalist id="browsers">
  <option value="Internet Explorer">
  <option value="Firefox">
  <option value="Chrome">
  <option value="Opera">
  <option value="Safari">
</datalist>
```

HTML5. formularios. Nuevos elementos (5)

- <keygen>
  - Forma se segura de autenticar usuarios.
  - La etiqueta <keygen> especifica un campo que genera un par de claves en un formulario
  - Cuando se envia el formulario, las dos claves se genera: una privada y una pública
    - La privada se almacena de forma privada, y la pública es enviada al servidor. La clave pública puede emplearse para generar un certificado para autenticar al usuarios en el futuro.

```
<form action="demo_keygen.asp"
method="get">
Username: <input type="text"
name="usr_name">
Encryption: <keygen name="security">
<input type="submit">
</form>
```

HTML5. formularios. Nuevos elementos (6)

- <output>
  - Representa el resultado de un cálculo (como si fuera un script).

```
<form oninput="x.value=parseInt(a.value)
+parseInt(b.value)">0
<input type="range" id="a" value="50">100 +
<input type="number" id="b" value="50">=
<output name="x" for="a b"></output>
</form>
```

HTML5. Elementos multimedia <canvas> (1)

- El elemento <canvas> es sólo un contenedor para gráficos por lo que es necesario emplear scripts para dibujarlos.
- Canvas presenta varios métodos para dibujar:
  - paths, boxes, circles, caracteres y añadir imágenes.
- Por defecto un <canvas> no presenta borde ni contenido.
- Etiqueta:

```
<canvas id="myCanvas" width="200" height="100">
</canvas>
```

- Especificar siempre el atributo “id” (para emplearlo desde el script) y los atributos width y height para definir el tamaño.
- Puedes tener múltiples <canvas> en una página HTML.
- Para añadir el borde emplea el atributo “estilo” :

```
<canvas id="myCanvas" width="200" height="100"
style="border:1px solid #000000;">
</canvas>
```

Internet Explorer 9+, Firefox, Opera, Chrome, and Safari support the <canvas> element.

Note: Internet Explorer 8 and earlier versions, do not support the <canvas> element.

HTML 5. Elementos multimedia (2)

```
<!DOCTYPE html>
<html>
<head>
   <meta charset="utf-8" />
   <title>Canvas example</title>
   <script>
   function draw () {
      var context;
      context = document.querySelector("canvas").getContext("2d”);
      context.fillStyle = "rgb(255, 0 , 0)";
      context.fillRect(20, 20, 120, 120);            // first square
      context.fillStyle = "rgb(255, 255, 0)";
      context.fillRect(100, 100, 250, 250);    // second square
   }
   window.onload = draw;
   </script>
</head>
<body>
   <canvas width="400" height="300"></canvas>
</body>
</html>
```

HTML 5. Elementos multimedia. <audio> y <video> (3)

- Audio y vídeo
  - El elemento audio permite insertar audio. Formatos: mp3, wav y ogg.
  - El elemento video permite insertar video. Formatos: mp4, ogg y webm (Ejemplo):

```
<!DOCTYPE html>
<html>
<head>
   <meta charset="utf-8" />
   <title>Audio example</title>
</head>
<body>
   <audio controls>
      <source src="sample.ogg" type="audio/ogg">
      <source src="sample.mp3" type="audio/
mpeg">
      Your browser does not support the audio
tag.
   </audio>
</body>
</html>
```

HTML5. SVG (1)

- SVG = Scalable Vector Graphics
- SVG se emplea para definir gráficos vectoriales para la web
- SVG define los gráficos en formato XML
- SVG al ser vectoriales no pierden calidad y pueden ser ampliado y redimensionados
- Cada elemento y atributo de un archivo SVG puede animarse
- SVG es una recomendación de W3C (http://www.w3.org/TR/SVG/)

Internet Explorer 9+, Firefox, Opera, Chrome, and Safari support inline SVG.

HTML5. SVG (2)

```
<!DOCTYPE html>
<html>
<body>
<svg xmlns="http://www.w3.org/2000/svg" version="1.1"
height="190">
  <polygon points="100,10 40,180 190,60 10,60 160,180"
  style="fill:lime;stroke:purple;stroke-width:5;fill-
rule:evenodd;">
</svg>
</body>
</html>
```

HTML5. Drag & drop

- En HTML5, drag &drop es parte del estandar, y cualquier elemento es arastrable.
- Para hacer un elemento arastrable:
  - <img draggable="true">

```
…..
<script>
function allowDrop(ev) { ev.preventDefault(); }
function drag(ev) { ev.dataTransfer.setData("Text",ev.target.id); }
function drop(ev) { ev.preventDefault(); var data=ev.dataTransfer.getData("Text");
ev.target.appendChild(document.getElementById(data)); }
</script>
<div id="div1" ondrop="drop(event)" ondragover="allowDrop(event)"></div>
<img id="drag1" src="img_logo.gif" draggable="true" ondragstart="drag(event)" width="336"
height="69">
…..
```

- Especificar qué arrastramos (**ondragstart** y setData())
  - Indicar que debería ocurrir cuando se arastra el elemento
  - En el ejemplo, el atributo **ondragstart** invoca a la función, **drag**(event), la cual especifica que dato va a ser atrastrado.
  - El método dataTransfer.setData() indica el tipo de dato y valor de dato arrastrable:
    - En este caso el tipo es "Text" y el valor es el id del elemento atrastrable ("drag1").
- Donde se suelta (**ondragover**)
  - El evento **ondragover** indica donde puede ser soltado el elemento arastrado
  - Por defecto, datos/elementos no pueden solterase sobre otros elementos. Para permitirlo debes espeficiar el manejador por defecto de prevención del elemento
    - Esto se hace invocando al evento event.preventDefault() para el evento ondragover.
- Ejecutar el proceso de soltar. (**ondrop**)
  - Cuando el objeto arastrado se suelta se ejecuta el evento drop
  - En el ejemplo el atributo ondrop llama a la función **drop**(event)

Almacenamiento Web (1)

- En HTML5 las páginas puedes almacenar datos locales al navegador
- Antes esto se hacía con cookies. El “Almacenamiento Web” de HTML5 es más seguro y rápido.
- Los datos almacenados no son enviados con cada solicitud al servidor, sino que **sólamente** se envián a demanda**.**
- Es posible almacenar grandes cantidades de información sin afectar al rendimiento del navegador.
- Los datos se almacenan como pares clave/valor y la página web sólo puede acceder a los datos almacenados por ella misma

Web storage is supported in Internet Explorer 8+, Firefox, Opera, Chrome, and Safari.

Note: Internet Explorer 7 and earlier versions, do not support web storage.

Almacenamiento Web (2)

- Existe dos objetos nuevos para almacenar datos en el navegador:
  - localStorage – Almacena datos sin fecha de caducidad
  - sessionStorage – Almacena datos durante una sesión
- Antes de usar el “ALWEB”, comprueba que tu navegador soporta localStorage y sessionStorage:

```
if(typeof(Storage)!=="undefined")
  {  // Yes! localStorage and sessionStorage support!
     // Some code.....
  }
else
  {
  // Sorry! No web storage support..
  }
```

Almacenamiento Web (3)

- El objeto localStorage no se borra cuando se cierra el navegador y está disponible “a-infinitum”

```
localStorage.lastname="Smith";
  document.getElementById("result").innerHTML="Last name:” + localStorage.lastname;
```

  - Se crea el par localStorage key/value = "lastname”/"Smith”
  - Se obtiene el valor de lastname y se escribe en el elemento “result”
  - Los pares Key/value se almacenan como strings. Recuerda convertirlos a otro formato si lo necesitas como el siguiente ejemplo:

```
if (localStorage.clickcount)
  {
  localStorage.clickcount=Number(localStorage.clickcount)+1;
  }
else
  {
  localStorage.clickcount=1;
  }
document.getElementById("result").innerHTML="You have clicked the
button " + localStorage.clickcount + " time(s).";
```

Almacenamiento Web (4)

- El objeto sessionStorage es igual a localStorage salvo que se borra al finalizar la sesión del navegador

```
localStorage.lastname="Smith";
  document.getElementById("result").innerHTML="Last name:” + localStorage.lastname;
```

  - Se crea el par localStorage key/value = "lastname”/"Smith”
  - Se obtiene el valor de lastname y se escribe en el elemento “result”
  - Los pares Key/value se almacenan como strings. Recuerda convertirlos a otro formato si lo necesitas como el siguiente ejemplo:

```
if (sessionStorage.clickcount)
  {
  sessionStorage.clickcount=Number(sessionStorage.clickcount)+1;
  }
else
  {
  sessionStorage.clickcount=1;
  }
document.getElementById("result").innerHTML="You have clicked the
button " + sessionStorage.clickcount + " time(s) in this
session.";
```

HTML5. Web workers (1)

- Cuando se ejecuta un script in una página HTML, la página no responde hasta que el script finaliza.
- Un web workers es un hilo de ejecución Javascript ejecutándose en background sin que afecte al rendimiento de la página: se puede continuar clickando, seleccionando cosas, etc.
- Existen dos tipos de web workers:
  - Dedicados. Están vinculados a su creador, es decir sólo se pueden comunicar con scripts del mismo cread
  - Compartidos. Se pueden comunicar con scripts (workers) de otros creadores en el mismo dominio.
- Antes usar web workers comprueba que tu navegador los soporta :

```
if(typeof(Worker)!=="undefined")
  {
  // Yes! Web worker support!
  // Some code.....
  }
else
  {
  // Sorry! No Web Worker support..
  }
```

HTML5. Web workers (2)

- Creación de un fichero de un web worker
  - Vamos a crear un web worker en un script Javascript externo.
  - Se trata de un script que cuenta. Almacenado en "demo_workers.js" :

```
var i=0;

function timedCount()
{
i=i+1;
postMessage(i);
setTimeout("timedCount()",500);
}
timedCount();
```

- El método postMessage() se emplean para enviar mensajes de respuesta a la página HTML.
- Nota: Normalmente los web workers se emplean para tareas de mayor carga de CPU que esta.

HTML5. Web workers (3)

- Ahora que tenemos el archivo Javascript, necesitamos invocar al web worker desde HTML.
- En el siguiente código comprobamos si el worker se ha creado previamente, y sino es así se crea un objeto nuevo web worker y ejecuta el código: "demo_workers.js":

```
if(typeof(w)=="undefined")
  {
  w=new Worker("demo_workers.js");
  }
```

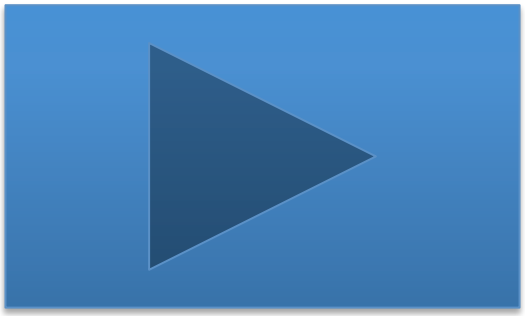
- Ahora podemos enviar y recibir mensajes del web worker
- Añade el manejador de eventos "onmessage" al web worker.

```
w.onmessage=function(event){
document.getElementById("result").innerHTML=event.data;
};
```

- Cuando el web worker responde un mensaje, se ejecuta el código del manejador onmessage. Los datos enviados por el web worker se almacenan en event.data.

HTML5. Web workers (4)

- Finalización de un Web Worker
  - Cuando se crea un web worker, continua escuchando mensajes (incluso cuando el script externo finaliza) hasta que éste termina.
  - Para finalizar un worker y liberar los recursos que ocupa en el navegador, se emplea el método terminate().
- Web workers y acceso a DOM
  - Ya que los workers son archivos externos, no tiene acceso a los siguientes objetos Javascript:
    - The window object
    - The document object
    - The parent object



CSS (1)

- Hojas de estilo en cascada
  - Las hojas de estilo (o CSS, por Cascading StyleSheets) es un lenguaje utilizado para describir **la semántica de la presentación** (aspecto y formato) de un documento escrito en un lenguaje de marcas.
  - Su aplicación más común es la de proporcionar estilo a las páginas web escritas en (X)HTML, aunque se puede aplicar también a cualquier documento XML.
  - Fue diseñado para separar la *información de presentación de un documento* (X)HTML de la *información de la estructura y contenido* de éste.
    - Facilitan la presentación del documento en distintos dispositivos y la mejora de la accesibilidad.
    - Permiten un mayor control sobre el aspecto final del documento.

CSS (2)

- Reglas generales
  - CSS se expresa mediante un conjunto de reglas en un fichero de texto plano. Tipo MIME: **`text/css`**.
  - Cada regla consta de dos partes: **selector** y **declaración**.
    - El **selector** determina que elementos estarán afectados por la regla.
    - La **declaración** establece que pares propiedad-valor se aplicarán a los elementos seleccionados.
  - CSS especifica un esquema de prioridades para determinar qué reglas de estilo se aplican si más de una regla coincide en contra de un elemento en particular.
    - En este esquema, denominado **cascada**, se calculan unas prioridades o pesos que se asignan a las reglas, de modo que los resultados siempre son predecibles.

CSS. Sintaxis

- CSS tiene una sintaxis sencilla y utiliza una serie de palabras clave en inglés para especificar los nombres de varias propiedades de estilo.
- Sintaxis de una regla:

```
selector1 [, selector2, ...] {
    prop-1: val-1;
    . . .
    prop-k: val-k;
}
```

opcional

comentario

```
body {
        color : #FFFFFF;  /* blanco */
        backgroung-color: rgb(90%, 90%, 90%);
        font-family: “Liberation Sans”, sans-serif;
}
```

CSS. Uso

- Métodos para especificar estilos en HTML
  - En documentos de texto independientes relacionados en el documento HTML mediante el elemento de cabecera `link`.
  - Estilos incrustados en el documento HTML mediante el elemento de cabecera `style`.
  - Estilos en línea aplicados a elementos específicos mediante el atributo `style` (desaconsejado, da lugar a documentos menos accesibles).

```
<head>
        . . .
        <link rel="stylesheet" type="text/css"
        href="estilo.css" media="screen">
</head>
```

CSS. Codificación de caracteres

- La regla `@charset`
  - Permite especificar la codificación de caracteres utilizada por el autor de una hoja de estilo en cascada. Esta regla debe ubicarse al principio de la hoja de estilo, sin estar precedida de otros caracteres. Sintaxis:

    `@charset “codificación”;`

```
@charset “ISO-88591”;

@charset “UTF-8”;
```

  - En HTML4 se puede especificar la codificación de caracteres en el atributo `charset` del elemento `link`, pero este atributo no existe en HTML5.

CSS. Tipos de medios (1)

Medio	Descripción
`all`	Todos los medios definidos
`braille`	Dispositivos táctiles que emplean el sistema braille
`embosed`	Impresoras braille
`handheld`	Dispositivos de mano
`print`	Impresoras y navegadores en el modo *imprimir*
`projection`	Proyectores y dispositivos para presentaciones
`screen`	Pantallas de ordenador
`speech`	Sintetizadores para navegadores de voz
`tty`	Dispositivos textuales limitados (teletipos y terminales)
`tv`	Televisores y dispositivos de resolución baja

CSS. Tipos de medios (2)

- Medios definidos con reglas de tipo `@media`
  - Las reglas `@media` permiten indicar de forma directa el medio o medios en los que se aplicarán los estilos incluidos en la regla. Sintaxis:

```
@media medio1 [, medio2, ...] {
    sel-1 { declar-1 }
    . . .
    sel-k { declar-k }
}
```

opcional

```
@media screen, print {
        body { line-height: 1.25em; }
}
```

CSS. Tipos de medios (3)

- Medios definidos con reglas de tipo `@import`
  - Las reglas de tipo `@import` permiten importar reglas de otros archivos CSS (excepto la regla `@charset`, si existe) y, opcionalmente, permiten especificar el medio al que se aplican los estilos. Sintaxis:

```
@import url("arch_css.css") [medio1, ...];
```

opcional

```
@import url("estilo1.css");
@import url("estilo2.css") print;
@import url("estilo3.css") projection, tv;
```

Id y Class

Estilos

- Backgrounds
- Texts
- Fonts
- Links
- Lists
- Tables

Box Model

- Box Model
- Border
- Outline
- Margin
- Padding

Avanzados

- Agrupamiento y anidamiento
- Dimension
- Display
- Posicionamiento
- Flotantes
- Image gallery
- Image sprites

CSS3: Novedades

- Borders
- Backgrounds
- Text effects
- Fonts
- Transformaciones 2D
- Transformaciones 3D
- Transiciones
- Animaciones
- Múltiples columnas
- Interface de usuario