

HTML, CSS y JavaScript

Programa: Web Fundamental

Ámbito: Programación de páginas web

Keywords: HTML, XHTML, CSS, browser, style, JavaScript, DOM

- 1. introducción**
- 2. conceptos básicos**
- 3. HTML estático**
- 4. estilos con CSS**
- 5. tablas**
- 6. JavaScript**
- 7. formularios**
- 8. evolución a HTML5**

1 **introducción**

2. conceptos básicos

3. HTML estático

4. estilos con CSS

5. tablas

6. JavaScript

7. formularios

8. evolución a HTML5

2 conceptos básicos

- 3. HTML estático
- 4. estilos con CSS
- 5. tablas
- 6. JavaScript
- 7. formularios
- 8. evolución a HTML5

conceptos

internet

- Sistema global de redes de computadores interconectadas.
- Es la **red de redes**.
- Conecta redes de distintos ámbitos: negocios, universidades, sector público, ...
- Utiliza el protocolo de comunicaciones **TCP/IP**.
- Utiliza distintas tecnologías de comunicaciones: cable, óptica, wi-fi, satélite, 3G/4G (móvil/celular).
- Nació en los años 60, creció en los 80 y se masificó mundialmente en los 90.
- Actualmente se utiliza para todo tipo de comunicaciones masivas: **World Wide Web**, telefonía, e-mail, compartir archivos, streaming, ...



conceptos

world wide web

La **World Wide Web** (www), también conocida como "la web", es un sistema de documentos en formato de hipertexto, que se acceden a través de internet.

- Utilizando un **navegador web**, se pueden obtener páginas web que contienen texto con formato, imágenes, y vínculos que permiten navegar hacia otras páginas.
- Fue desarrollada entre 1989 y 1990.
- Utiliza una arquitectura **cliente-servidor**. El cliente es el navegador, que recibe la información desde un servidor a través de internet.

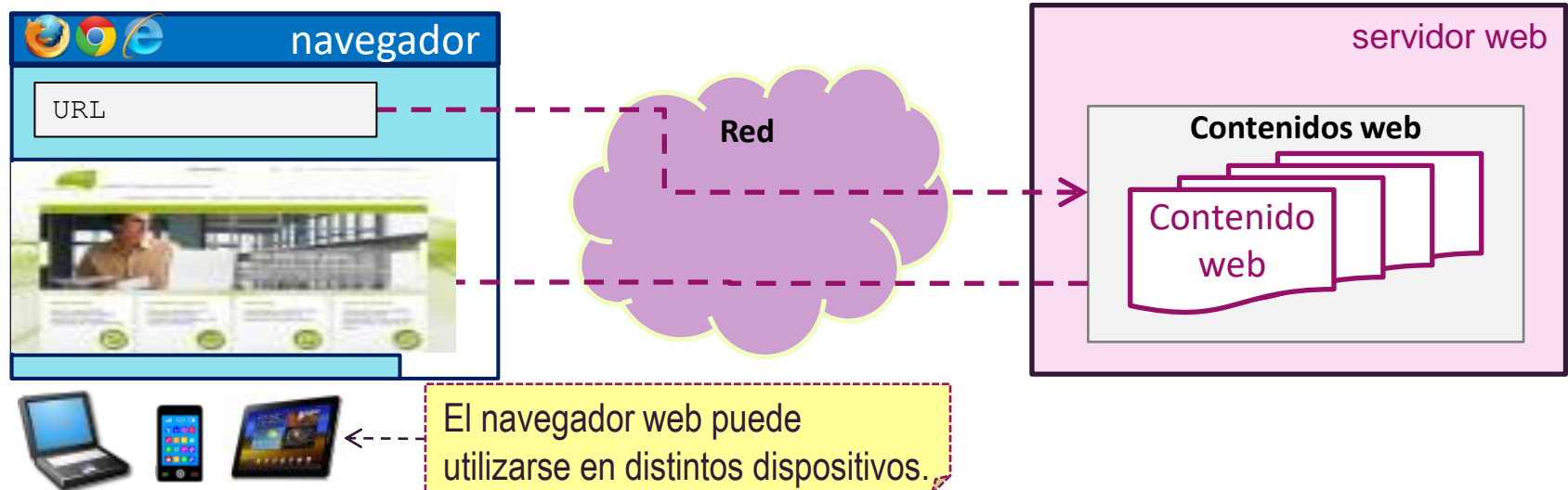


conceptos

servidor web

Un **servidor web** se refiere al software y hardware necesario para distribuir **contenidos web** a través de la red.

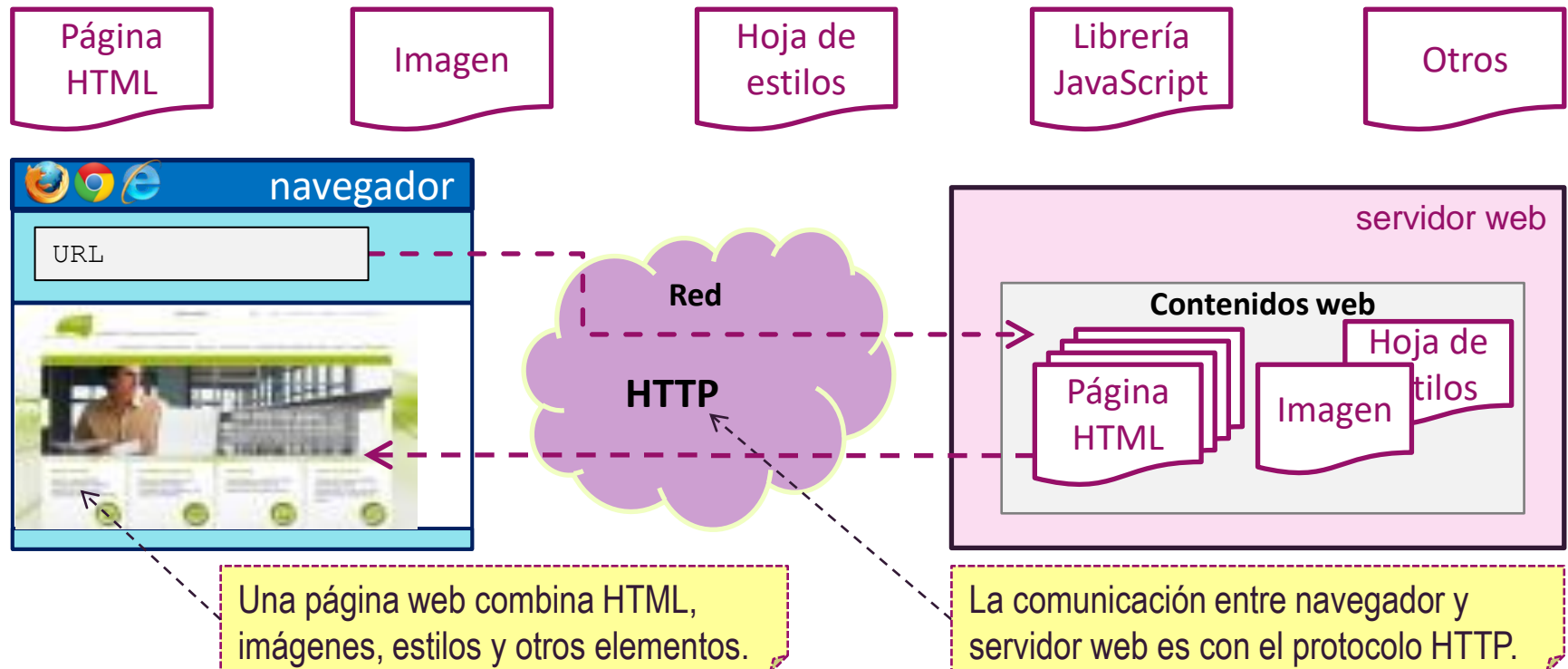
- Desde un navegador web, que equivale al "cliente", se hace la petición al servidor, a través de una **URL** (Uniform Resource Locator).
- El servidor responde a la petición con el contenido web, el cual es retornado al navegador, quien lo interpreta y muestra.



conceptos

tipos de contenido

Un servidor web contiene distintos tipos de contenidos web, almacenados en archivos, que son obtenidos y utilizados por un navegador:



conceptos

¿qué es HTML?

HTML (Hyper Text Markup Language), es un lenguaje informático diseñado para estructurar textos de acuerdo al formato estándar de las páginas web.

```
<!doctype html ...>
<html id="ctl00_Html1"
...><head>...
<title>everis</title>
...
<body>...</body>
</html>
```

interpretado

Lenguaje HTML,
formado por etiquetas.

El navegador interpreta
HTML, y muestra el
resultado.



conceptos

HTML

Características de HTML:

- Representa en forma de **hipertexto** el formato estándar de las páginas web.
- Consta de una serie de marcas o **etiquetas** (tags en inglés), que son interpretadas por el navegador, el cual presenta gráficamente los elementos (texto, imágenes,...) que contiene el documento.
- Un documento HTML es un **archivo de texto**, por lo que en principio se puede utilizar cualquier editor para escribirlo.
- Si la extensión de la página es .html o .htm, el navegador la reconoce e interpreta.

conceptos

alcance del curso

- En este curso, se detalla el lenguaje HTML desde el punto de vista de **programación de la parte cliente**, es decir, contenido, estilos gráficos y comportamiento específico en el navegador. Las páginas sólo se prueban **localmente**, no desde un servidor web.

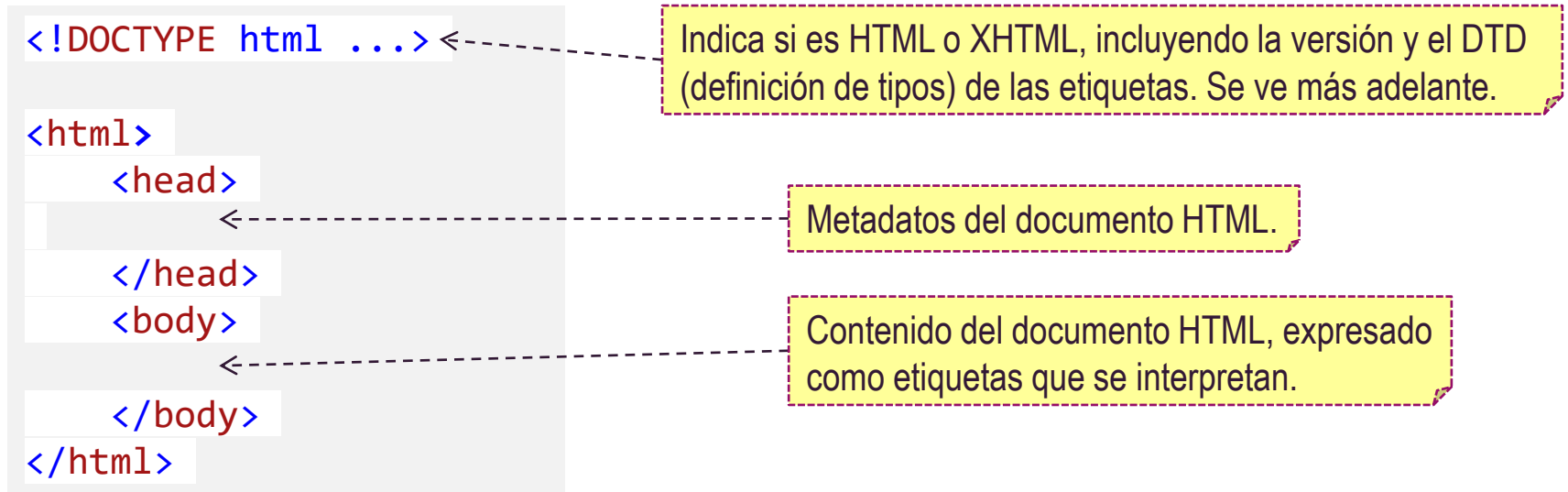


- En cursos de las tecnologías web (Java, .NET) se describe cómo se generan las páginas HTML como parte de aplicaciones empresariales, y cómo se accede a éstas por una URL de un servidor u otra forma de navegación. Para edición de páginas, cada tecnología usa sus propias herramientas (Eclipse, Visual Studio, ...). En este curso se utiliza un **editor genérico HTML**.

conceptos

estructura básica

Un documento HTML tiene la siguiente estructura básica:



conceptos

navegador, definición

Un **navegador web** o **browser** es un software que se utiliza para buscar, recuperar y mostrar información, principalmente desde internet.

- Los documentos se obtienen a través de un identificador, llamado genéricamente **URI** (Uniform Resource Identifier).
- Como se ha dicho anteriormente, los documentos pueden ser páginas web, imágenes, videos, hojas de estilo, o cualquier otro archivo interpretable.
- Los navegadores más conocidos son:

Chrome
(Google)



Firefox
(Mozilla)



Internet Explorer
(Microsoft)



Opera
(Opera Software)



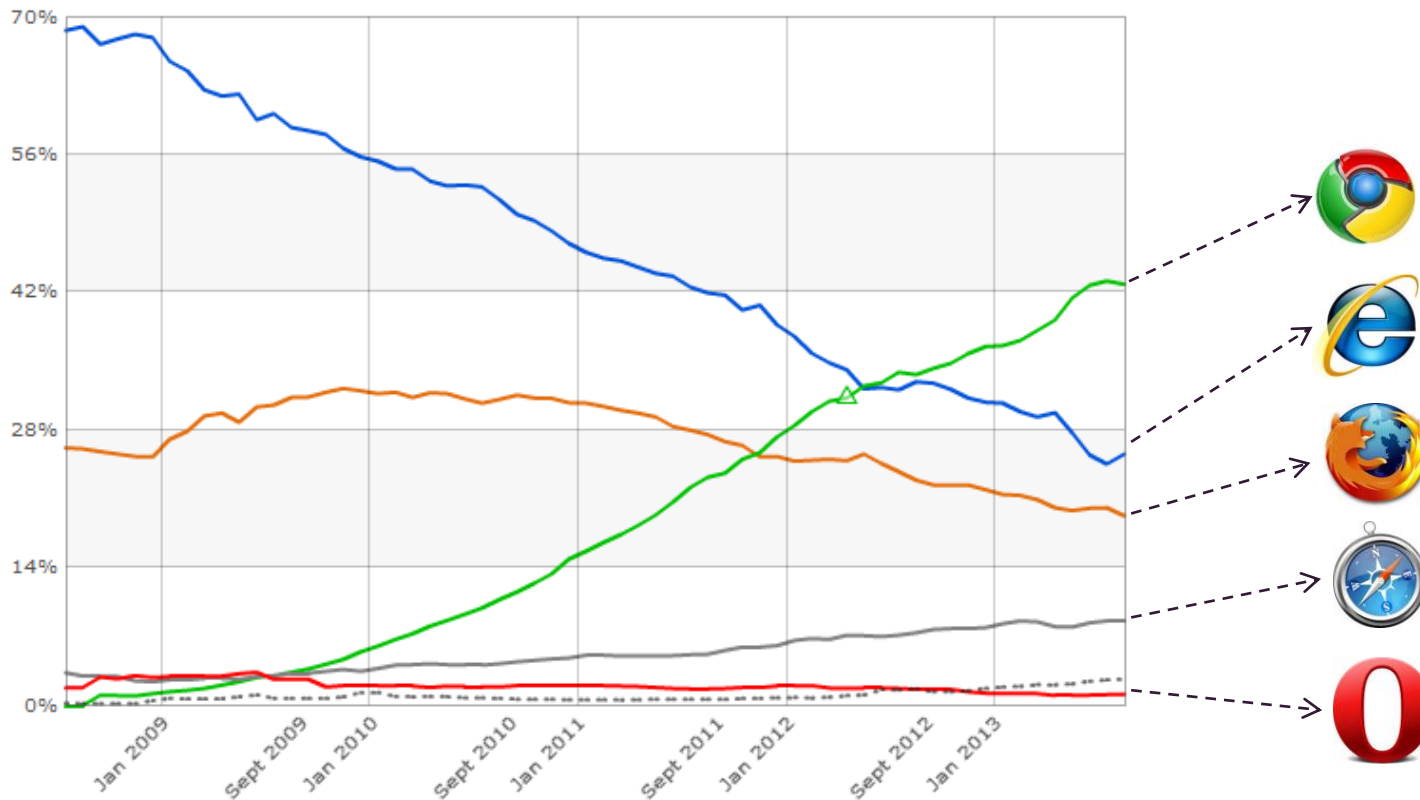
Safari
(Apple)



conceptos

navegador, comparativa

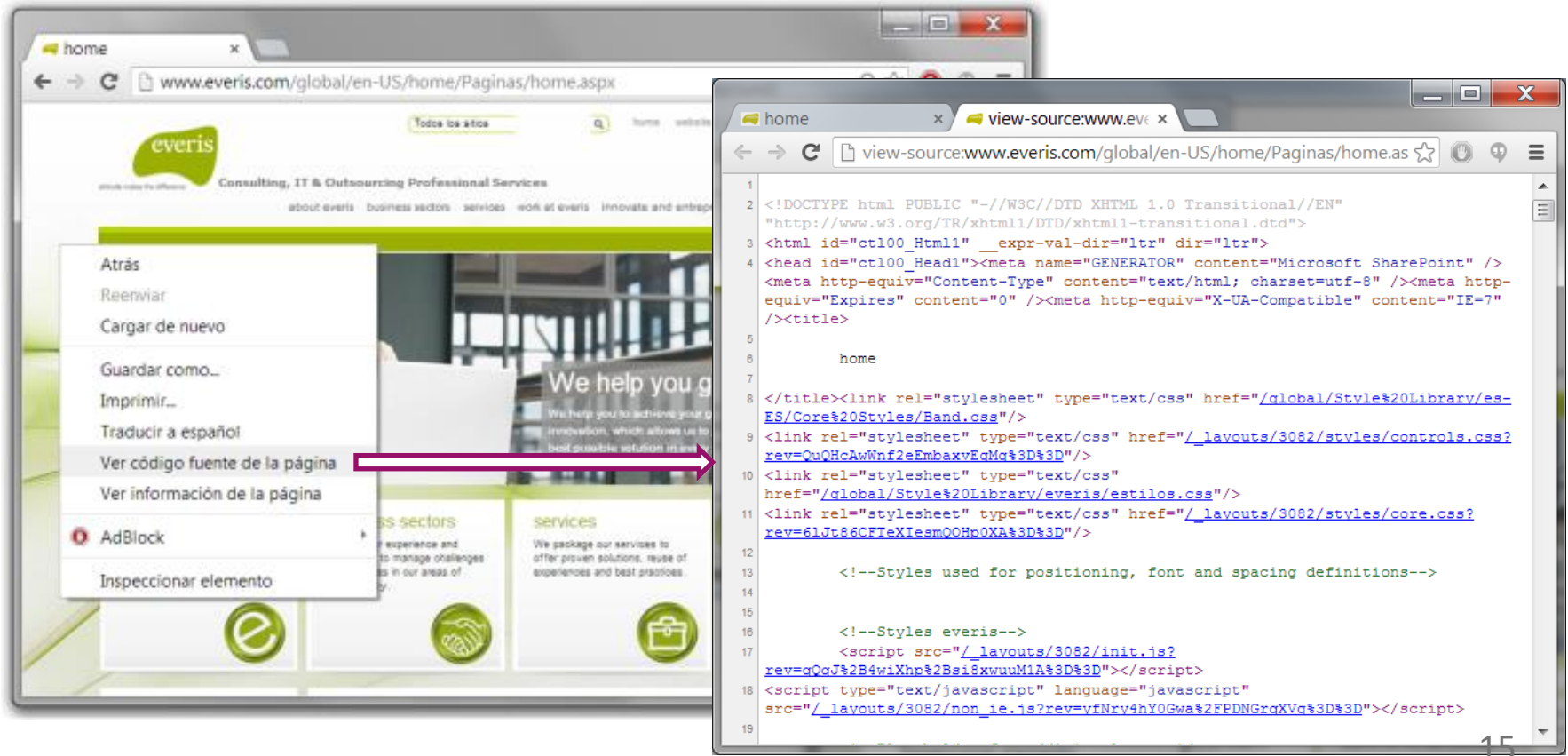
Las tendencias en el uso de los navegadores han cambiado mucho a lo largo de los años. En los años 2009-2013, la utilización es la siguiente:



conceptos

código fuente

Desde un navegador se puede ver el código fuente HTML de la página.



conceptos básicos

protocolo HTTP

HTTP (HyperText Transfer Protocol), protocolo de transporte de hiper texto.

- Es el que normalmente se utiliza para comunicar un navegador con un servidor.
- En términos generales, es un protocolo para intercambiar o transferir "hypertext", que son estructuras lógicas de información en formato de texto.
- Ejemplo de mensajes de navegación web con HTTP (simplificados):

request

```
GET /app/pag1.html HTTP/1.1  
Host: www.everis.com
```

response

```
HTTP/1.1 200 OK  
Server: ...  
Content-Type: ...  
  
<html>  
  <head>...</head>  
  <body>...</body>  
</html>
```


conceptos

URL

Una **URL** (**U**niform **R**esource **L**ocator) es una cadena de texto que corresponde a un recurso web.

El formato de una URL es:

`protocol://path/to/resource`

Cuando la URL se refiere a un archivo del disco local, tiene la forma:

`file://path/to/resource`

Ejemplo en Windows:

`file://C:/BpE/ws-html/html-basic/catalogo.html`

Cuando la URL es remota, en una ubicación en un servidor y utilizando el protocolo HTTP, tiene la forma:

`http://servidor:puerto/application/path/to/resource`

Protocolo más
utilizado en web.

El puerto permite que desde una misma máquina se puedan
publicar varios servicios. Con http, si se omite es el 80.

conceptos

Caso práctico 1: Primera aproximación al www

Resumen del ejercicio:

- Abrir en un navegador web un sitio web.
- Ver el código fuente.
- Observar y reconocer sus partes principales.

3 HTML estático

- **contenido básico**
 - reglas básicas
 - listas
 - navegación
 - agrupamiento
 - head
 - inspección de elementos

4. estilos con CSS

5. tablas

6. JavaScript

7. formularios

8. evolución a HTML5

contenido básico

uso de etiquetas

HTML define su contenido a través de **etiquetas (tags)**, que son los **elementos** del documento. La sintaxis de una etiqueta depende de si tiene contenido o no.

- En el caso de las que tienen contenido, la nomenclatura es:

```
<nombre-elemento [atributo="valor"]>Contenido</nombre-elemento>
```

Mismo nombre al abrir y cerrar etiqueta

- En el caso de las que no tienen contenido, la nomenclatura es:

```
1) <nombre-elemento [atributo="valor"]></nombre-elemento>
```

```
2) <nombre-elemento [atributo="valor"]/>
```

- Los **atributos** son utilizados por algunas etiquetas para especificar propiedades del contenido.

A continuación, se presentan las etiquetas básicas de HTML.

contenido básico

encabezados

Para agregar un encabezado (header) al contenido de un documento, se utilizan los tags `<h1>` a `<h6>`, dependiendo de la importancia.

```
<h1>Encabezado 1</h1>  
<h2>Encabezado 2</h2>  
<h3>Encabezado 3</h3>  
<h4>Encabezado 4</h4>  
<h5>Encabezado 5</h5>  
<h6>Encabezado 6</h6>
```

Son de tipo *block*, pues después de cada uno se agrega un salto de línea.

Encabezado 1

Encabezado 2

Encabezado 3

Encabezado 4

Encabezado 5

Encabezado 6

contenido básico

párrafo

Para especificar un párrafo, se utiliza la etiqueta `<p>`:

```
<h2>HTML, CSS y JavaScript</h2>
```

```
<p>El contenido incluye  
HTML, CSS y JavaScript</p>
```

```
<p>Primero se describe el  
lenguaje HTML</p>
```

HTML, CSS y JavaScript

El contenido incluye HTML, CSS y JavaScript

Primero se describe el lenguaje HTML

El espacio entre párrafos es doble.

No se aplica salto de línea del código fuente.

Es de tipo *block*, pues después de cada uno se agrega un salto de línea.

contenido básico

párrafo

Al párrafo se le puede especificar la alineación:

```
<h3>Detalles</h3>
```

```
<p align="justify">El atributo  
align del párrafo define su  
alineación. Las opciones  
principales son left, center,  
right y justify (horizontal),  
y existen otras como top, middle  
y bottom para la alineación  
vertical.</p>
```

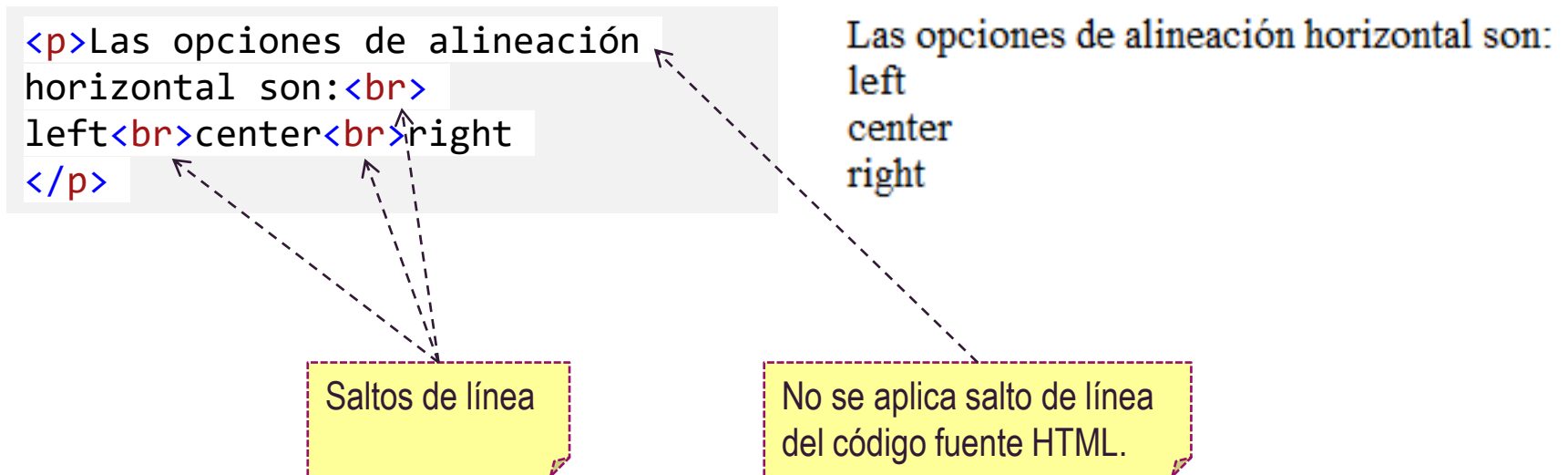
Detalles

El atributo align del párrafo define su alineación. Las opciones principales son left, center, right y justify (horizontal), y existen otras como top, middle y bottom para la alineación vertical.

contenido básico

break

El salto de línea del código fuente no tiene efecto sobre el HTML. Para agregar un salto de línea, o *break*, se utiliza `
`:



contenido básico

énfasis de texto

A un texto se le puede cambiar el aspecto, utilizando:

- **negrita:**

Conocer HTML es
muy importante.

Conocer HTML es **muy importante**.

- *cursiva:*

El atributo <i>align</i>
especifica alineación.

El atributo *align* especifica alineación.

- subrayado:

Más adelante se resuelve esto
con <u>estilos gráficos</u>

Más adelante se resuelve esto con estilos gráficos.

- *énfasis:*

Este es el contenido
básico.

Este es el *contenido básico*.

Etiqueta en desuso. Se reemplaza
por estilos gráficos.

contenido básico

imagen

Se puede agregar una imagen utilizando la etiqueta ****:

```

```

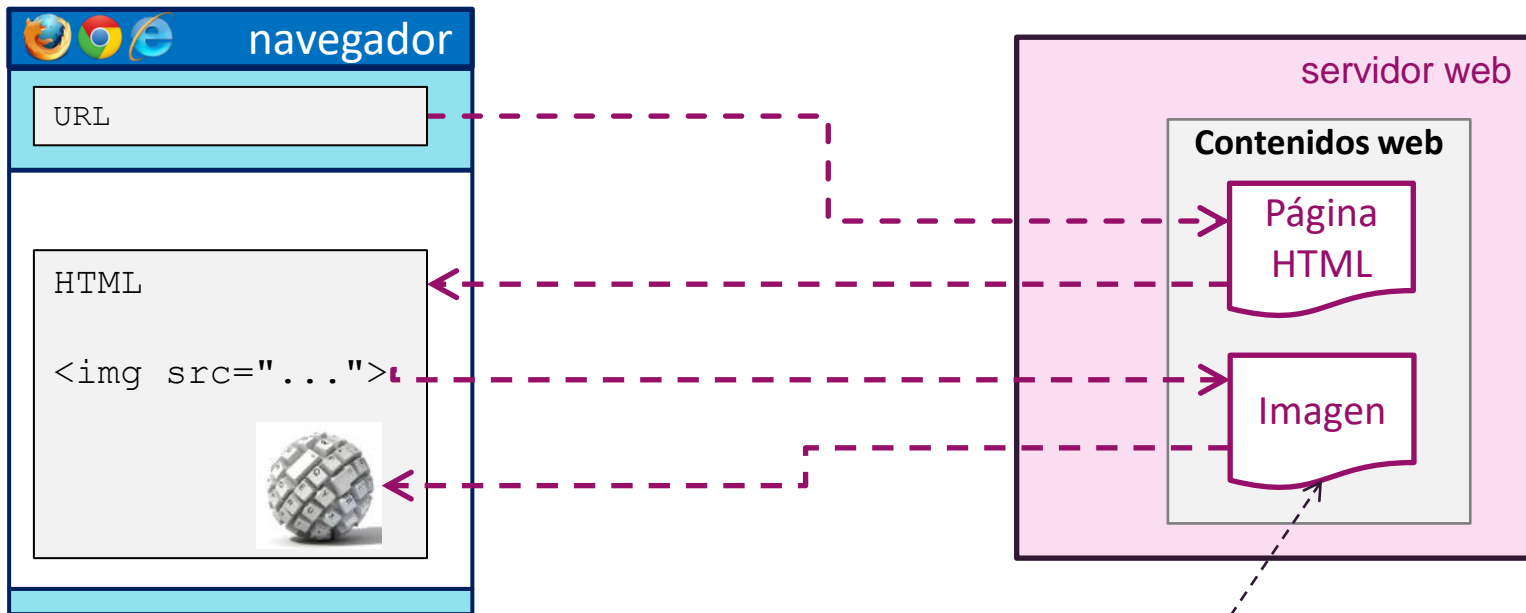
fuelle: URL o ruta (relativa o absoluta) de la imagen.



contenido básico

imagen

A diferencia de los elementos anteriores, que se interpretan directamente, la imagen debe cargarse previamente desde la dirección indicada en el atributo src.



La imagen podría estar en otro servidor.

contenido básico

imagen

Se puede especificar el alto y ancho de una imagen, la que se ajusta:

```

```

```

```

```

```



Si se omite uno de los dos atributos, se mantiene la relación de aspecto.

No es conveniente cambiar mucho el tamaño:

- Si se reduce mucho, se transfiere una imagen mucho más grande de lo que se muestra. Conviene ajustar su tamaño en el origen.
- Si se amplía mucho, se pueden observar problemas de nitidez.

contenido básico

imagen, tipo

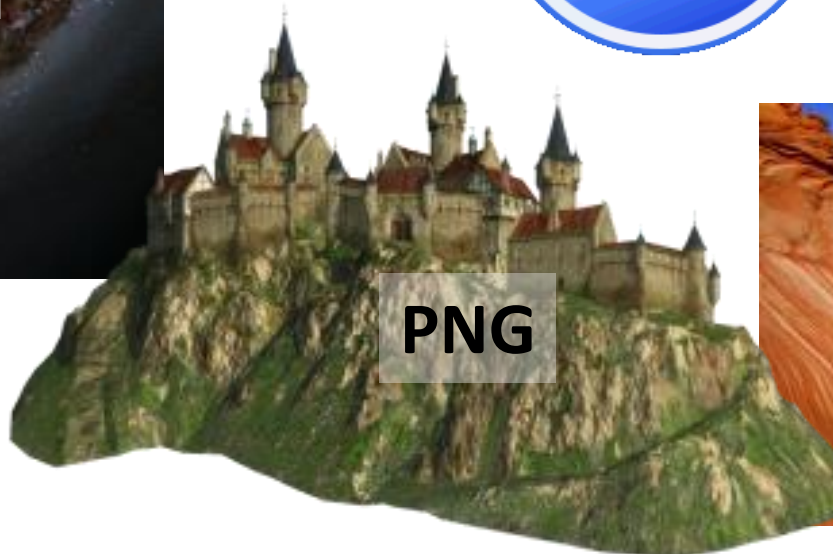
Los navegadores soportan varios tipos de imagen:

- **GIF** (Graphics Interchange Format): Para imágenes de pocos colores, que no deben perder nitidez, como los logos, o textos en formato imagen. Soporta transparencia.
- **JPG** (de JPEG, Joint Photographic Experts Group): es un formato comprimido, que a cambio tiene una pérdida de calidad ajustable en función de cuánto se comprime. Es adecuado para fotografías grandes, donde la pérdida es poco perceptible. No soporta transparencia.
- **PNG** (Portable Network Graphics): Tiene tamaño intermedio entre GIF y PNG. Soporta transparencia. Permite utilizar más colores que GIF dependiendo del número de bits (PNG-8, 24 o 32). Se utiliza en los mismos casos que GIF, para imágenes que requieren nitidez como logos o textos en imágenes.

contenido básico

¿GIF, JPG o PNG?

¿Cuál tipo de imagen utilizar? Depende del tipo de imagen.



contenido básico

imagen, atributos

Una imagen puede tener una representación alternativa en caso que no logre cargarse, con el atributo **alt**:

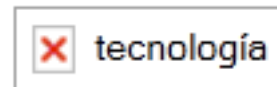
```

```

Si el archivo de la imagen no se encuentra, se ve lo siguiente en función del navegador:



tecnología



contenido básico

tooltip

En general, los elementos HTML tienen un atributo llamado **title**, que despliega un **tooltip** (texto flotante) cuando se pasa el puntero del mouse sobre éste.

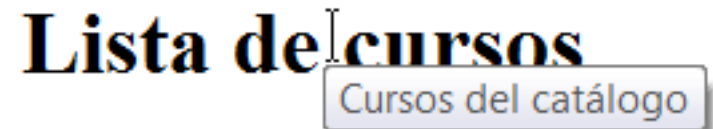
Ejemplos:

```

```



```
<h1 title="Cursos del catálogo">  
  Lista de cursos</h1>
```



contenido básico

Caso práctico 1: Reconocimiento visual de elementos

Resumen del ejercicio:

Dado el código fuente de un documento HTML, se pide:

- Abrirlo en un navegador web.
- Asociar las partes del documento con el código HTML.
- Realizar modificaciones con un editor de texto.

Una URL (*Uniform Resource Locator*) es una cadena de texto que corresponde a un recurso web.

3 HTML estático

- **reglas básicas**
- listas
- navegación
- agrupamiento
- head
- inspección de elementos

4. estilos con CSS

5. tablas

6. JavaScript

7. formularios

8. HTML5

9. resumen y conclusiones

10.anexos

reglas básicas

espacios, tabulaciones y saltos de línea

HTML sólo considera un espacio, aunque hayan más, e ignora las tabulaciones y saltos de línea:

```
<p>Los múltiples    espacios, tabulaciones y saltos  
de línea se interpretan como un solo espacio</p>
```

Los múltiples espacios, tabulaciones y saltos de línea se interpretan como un solo espacio

Para insertar más de un espacio, se debe utilizar " ". Ejemplo:

```
<p>Múltiples&nbsp;&nbsp;&nbsp;espacios entre palabras</p>
```

Tres espacios

Múltiples espacios entre palabras

reglas básicas

block vs inline

En HTML existen etiquetas del tipo *block* y del tipo *inline*:

- Las de tipo *block* agregan un salto de línea después de su utilización:

```
<h2>Encabezado 2</h2>  
<h3>Encabezado 3</h3>
```

Encabezado 2

Encabezado 3

Salto de
línea

- Las de tipo *inline* no lo agregan:

```
  

```



No hay salto
de línea

reglas básicas

anidamiento de etiquetas

En HTML, las etiquetas se pueden anidar, como se ha observado en los ejemplos:

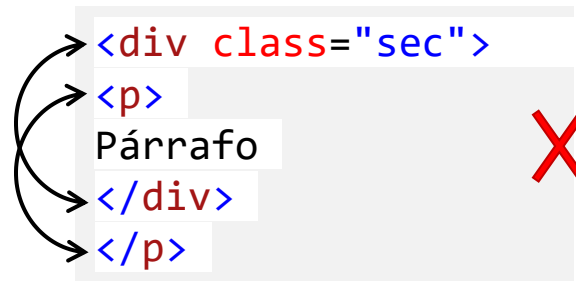
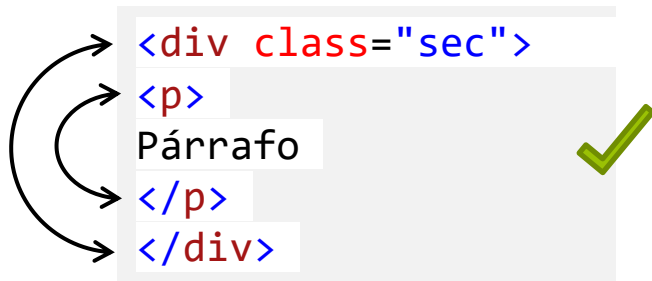
```
<html xmlns="http://www.w3.org/1999/xhtml">  
  <body>  
    <div class="frame">  
      <div class="brd">  
        <div class="sec">Sección</div>  
      </div>  
    </div>  
    <p>Párrafo</p>  
  </body>  
</html>
```

Esto implica que un documento HTML tiene una estructura tipo árbol. Más adelante se detalla.

reglas básicas

cierre de etiquetas

Al estar anidadas, las etiquetas deben cerrarse de manera adecuada, respetando el orden de la jerarquía:



reglas básicas

comentarios

En HTML, la nomenclatura de comentarios es la siguiente:

```
<!-- bloque comentado -->
```

El bloque comentado puede tener una o más líneas, y cualquier contenido. Se utilizan para facilitar el entendimiento del código HTML:

```
<!--  
Inicio sección de datos  
-->  
<h1>Catálogo de cursos</h1>
```

Rigurosamente hablando, los comentarios HTML deben cumplir con el estándar XML, por lo que no deben incluir en su contenido dos guiones seguidos, aunque los navegadores ignoran esta restricción:

```
<!-- No -- permitido aunque funciona -->
```

reglas básicas

caracteres especiales

En HTML, los caracteres especiales (acentos, eñes y otros) se pueden escribir directamente:

```
<p>la antigüedad en la compañía</p>
```

No obstante, si el **juego de caracteres** de la página no corresponde con el del archivo, se pueden producir conflictos que hacen que no se vean correctamente. Por ejemplo, si el archivo HTML utiliza UTF-8 y la página declara ISO-8859-1, se ve así:

la antigÃ¼edad en la compaÃ±a

Nota: los detalles de los juegos de caracteres se ven en cursos posteriores.

reglas básicas

caracteres especiales

Para evitar los efectos por diferencias de juego de caracteres entre el archivo físico y el declarado en el contenido HTML, existe una nomenclatura para los caracteres especiales. En el ejemplo:

```
<p>la antiguedad en companiacuta</p>
```

Nomenclatura para las vocales acentuadas:

&[letra][tipo modificador];

a – e

A – E

En algunos idiomas,
otras letras no vocales.

Distintos tipos usados en algunos idiomas:

- acute (agudo): á, Á
- grave (grave): à, À
- circ (cincunflejo): â, Â
- tilde (tilde): ã, Ã
- uml: ä, Ä
- ring (anillo): å, Å

reglas básicas

caracteres especiales

Además de los acentos, hay otros caracteres especiales, algunos de los cuales se describen en la siguiente tabla:

Caracter	Nombre entidad	Descripción
&	&	símbolo & (<i>ampersand</i>)
(espacio)	 	espacio en blanco (<i>non breaking space</i>)
"	"	comillas
'	'	apóstrofe (las llamadas comillas simples)
<	<	menor que (<i>less than</i>)
>	>	mayor que (<i>greater than</i>)
¿	¿	abre signo de interrogación
!	¡	abre signo de exclamación/admiración
©	©	copyright

reglas básicas

caracteres especiales

También se utiliza la nomenclatura "&#[código decimal];" para los caracteres especiales. Ejemplos:

Caracter	Nombre entidad	Descripción
(espacio)	 	Espacio en blanco
&	&	símbolo &
"	"	comillas
'	'	apóstrofe (las llamadas comillas simples)
<	<	menor que
>	>	mayor que
¿	¿	abre signo de interrogación
¡	¡	abre signo de exclamación/admiración

reglas básicas

Caso práctico 2: Overview de herramientas open source

Resumen del ejercicio:

- Dada una definición de página HTML simple, escribirla utilizando algunas herramientas, para conocer las características de cada una.
- Herramientas a utilizar:
 - Amaya
 - SeaMonkey
 - AptanaStudio
- Analizar las diferencias.

3 HTML estático

- **listas**
- navegación
- agrupamiento
- head
- inspección de elementos

4. estilos con CSS

5. tablas

6. JavaScript

7. formularios

8. evolución a HTML5

9. resumen y conclusiones

10.anexos

listas

descripción

Existen dos tipos de **listas**, las ordenadas (****) y no ordenadas (****). La lista ordenada coloca un índice en cada ítem:

- Lista ordenada con **números** (opción ordenada default):

```
<h3>Cursos línea Java:</h3>
<ol>
  <li>Java Básico</li>
  <li>Java Intermedio</li>
  <li>Spring Core</li>
  <li>Hibernate-JPA</li>
  <li>Web Services</li>
</ol>
```

ordered list

list item

Cursos línea Java:

1. Java Básico
2. Java Intermedio
3. Spring Core
4. Hibernate-JPA
5. Web Services

listas

descripción

- Ordenada con **letras** minúsculas:

```
<ol type="a">  
  <li>Java Básico</li>  
  <li>Java Intermedio</li>  
  <li>Spring Core</li>  
  <li>Hibernate-JPA</li>  
  <li>Web Services</li>  
</ol>
```

- a. Java Básico
- b. Java Intermedio
- c. Spring Core
- d. Hibernate-JPA
- e. Web Services

- Ordenada con **números romanos** en mayúsculas:

```
<ol type="I">  
  <li>Java Básico</li>  
  <li>Java Intermedio</li>  
  <li>Spring Core</li>  
  <li>Hibernate-JPA</li>  
  <li>Web Services</li>  
</ol>
```

- I. Java Básico
- II. Java Intermedio
- III. Spring Core
- IV. Hibernate-JPA
- V. Web Services

Nota: el atributo type está en desuso.
Más adelante se utilizan estilos gráficos.

listas

descripción

En la lista no ordenada, los elementos aparecen con *bullets*:

```
<h3>Cursos línea .NET:</h3>
<ul>
  <li>C# Básico</li>
  <li>C# Avanzado</li>
  <li>everNExT</li>
  <li>Persistencia</li>
  <li>Servicios Web</li>
</ul>
```

unordered list

list item

Cursos línea .NET:

- C# Básico
- C# Avanzado
- everNExT
- Persistencia
- Servicios Web

3 HTML estático

- **navegación**
- agrupamiento
- head
- inspección de elementos

4. estilos con CSS

5. tablas

6. JavaScript

7. formularios

8. evolución a HTML5

9. resumen y conclusiones

10.anexos

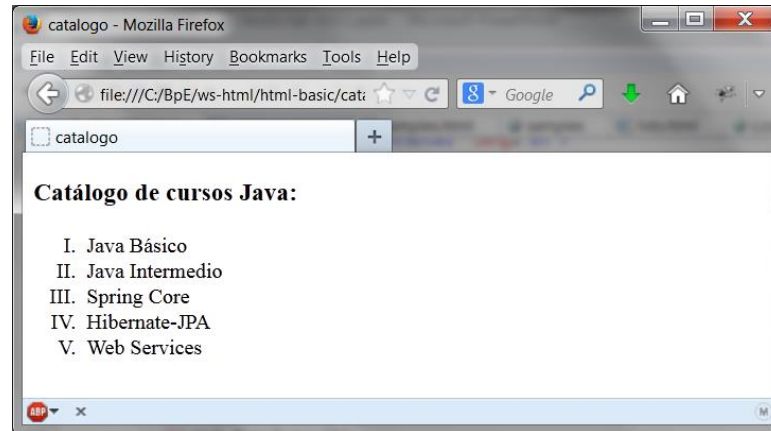
navegación

hyperlink

Para navegar desde una página HTML a otra, se puede utilizar un **hyperlink**:

```
<a href="catalogo.html">  
  Ir a catálogo</a>
```

[Ir a catálogo](#)



Un link, una vez visitado, por default cambia su aspecto gráfico:

[Ir a catálogo](#)

A través de estilos gráficos, se puede controlar el aspecto de un hyperlink.

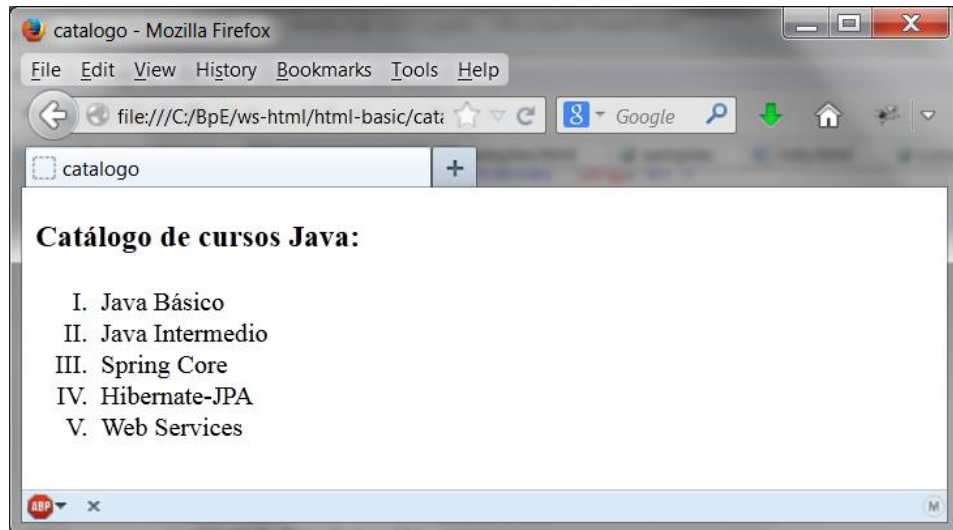
navegación

hyperlink

En un hyperlink, aparte de texto, se puede colocar una **imagen**:

```
<a href="catalogo.html"></a>
```

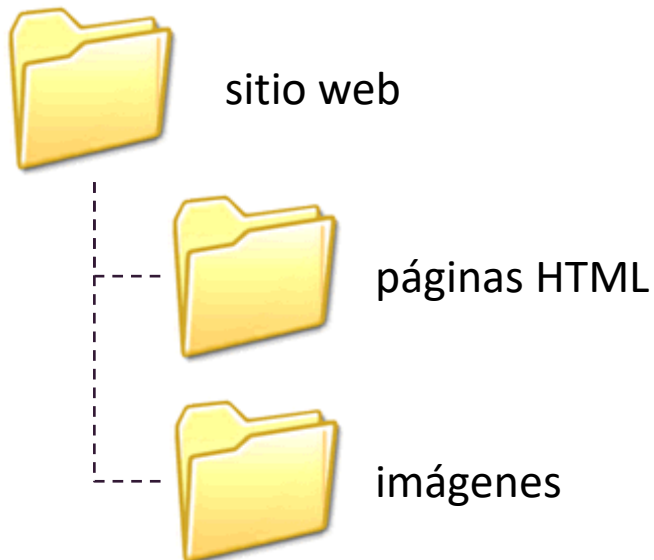
Nótese que no hay espacios entre
<a> e , ni antes ni después



navegación

estructura de una aplicación

Las pantallas e imágenes se pueden organizar en directorios, los que conforman un **sitio web**:



Podrían haber varios directorios de páginas o imágenes

Más adelante se presentan otros tipos de archivo, que forman parte de la estructura de un sitio web.

navegación

hyperlink

En el atributo **href** se puede incluir:

- Una URL de una página de destino:

```
<a href="http://www.everis.com">everis Home</a>
```

- Una ruta relativa o absoluta:

```
<a href="../pages/catalogo.html">Ir a catálogo</a>
```

```
<a href="C:/BpE/html/pages/catalogo.html">Ir a catálogo</a>
```

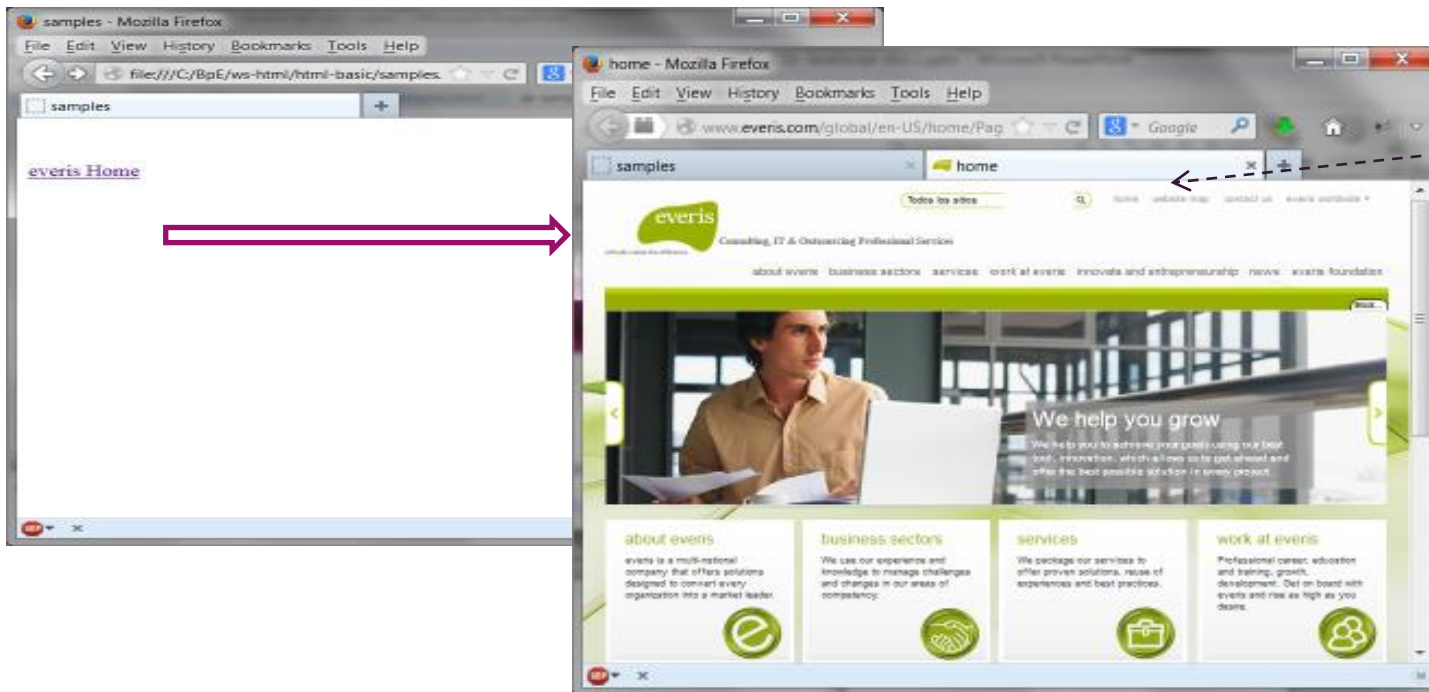
Nótese que la ruta relativa o absoluta es manejada también como una URL.

navegación

hyperlink

En el atributo **target** permite definir como destino otra ventana o pestaña:

```
<a href="http://www.everis.com" target="_blank">everis Home</a>
```



Nueva
pestaña

navegación

anchor

La etiqueta `` es un **anchor** (ancla), que permite definir posiciones en una pantalla para desplazarse, desde la propia pantalla o una externa:

Nomenclatura de hyperlink a un anchor: # seguido del id del anchor de destino

Anchor, con su id

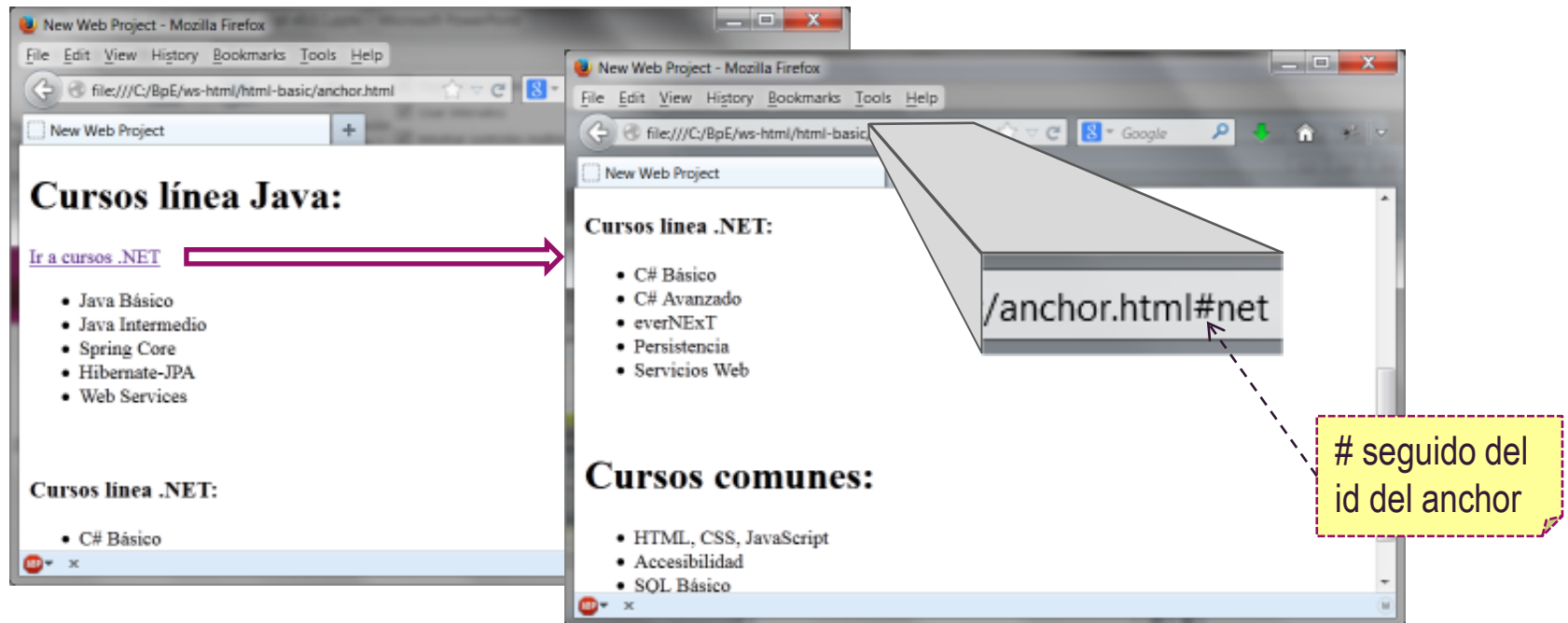
```
<h1>Cursos línea Java:</h1>
<a href="#net">Ir a cursos .NET</a>
<ul>
  <li>Java Básico</li>
  <li>Java Intermedio</li>
  ...
</ul>
<a id="net"><h3>Cursos línea .NET:</h3></a>
<ul>
  <li>C# Básico</li>
  <li>C# Avanzado</li>
  ...
</ul>
<h1>Cursos comunes:</h1>
...
```

Nota: la inicial de anchor es la que da el nombre a la etiqueta `<a>`

navegación

anchor

Se observa el desplazamiento de la pantalla:



Para ir a un anchor de otra pantalla, es su URL seguida de #id

3 HTML estático

- **agrupamiento**
- head
- inspección de elementos

4. estilos con CSS

5. tablas

6. JavaScript

7. formularios

8. evolución a HTML5

9. resumen y conclusiones

10.anexos

agrupamiento

fieldset

La etiqueta **<fieldset>** permite agrupar elementos de una página HTML:

```
<h3>Cursos línea Java:</h3>
<fieldset>
  <ul>
    <li>Java Básico</li>
    <li>Java Intermedio</li>
    ...
  </ul>
</fieldset>
<h3>Cursos línea .NET:</h3>
<fieldset>
  <ul>
    <li>C# Básico</li>
    <li>C# Avanzado</li>
    ...
  </ul>
</fieldset>
```

Cursos línea Java:

- Java Básico
- Java Intermedio
- Spring Core
- Hibernate-JPA
- Web Services

Cursos línea .NET:

- C# Básico
- C# Avanzado
- everNExT
- Persistencia
- Servicios Web

agrupamiento

legend

Dentro de un `<fieldset>`, la etiqueta `<legend>` permite especificar un texto al grupo:

```
<h3>Cursos:</h3>
<fieldset>
  <legend>Línea Java</legend>
  <ul>
    <li>Java Básico</li>
    <li>Java Intermedio</li>
    ...
  </ul>
</fieldset>
<fieldset>
  <legend>Línea .NET</legend>
  <ul>
    <li>C# Básico</li>
    <li>C# Avanzado</li>
    ...
  </ul>
</fieldset>
```

Cursos:

Línea Java

- Java Básico
- Java Intermedio
- Spring Core
- Hibernate-JPA
- Web Services

Línea .NET

- C# Básico
- C# Avanzado
- everNExT
- Persistencia
- Servicios Web

agrupamiento

sección

Una página puede ser dividida en secciones, las cuales pueden tener un tratamiento particular. Para ello, se utilizan las etiquetas **<div>** y ****.

```
<div>  
  <h3>Título</h3>  
  <p>Contenido</p>  
</div>
```

```
<p>Curso <span>(1)</span>: HTML</p>
```

La diferencia entre `<div>` y `` es que el primero es de tipo *block*, es decir, agrega un salto de línea, mientras que el segundo es *inline*, y no lo agrega.

Utilidad del manejo de secciones:

- Definir características comunes, principalmente a través de estilos gráficos, como se ve más adelante.
- Definir comportamientos comunes, a través de JavaScript, como se ve más adelante.

3 HTML estático

- **head**
- inspección de elementos

4. estilos con CSS

5. tablas

6. JavaScript

7. formularios

8. evolución a HTML5

head

descripción

El **head** (**cabecera**) contiene información del documento que no se muestra directamente al usuario. Está delimitada por la etiqueta **<head>**, y la información es:

- El título, que se observa en la barra de título del navegador o en la pestaña.
- Vínculos a hojas de estilo
- Vínculos a scripts
- Instrucciones “meta”

Toda esta información es opcional.

```
<!DOCTYPE html ...>  
  
<html>  
  <head>  
  </head>  
  <body>  
  
  </body>  
</html>
```

head

título

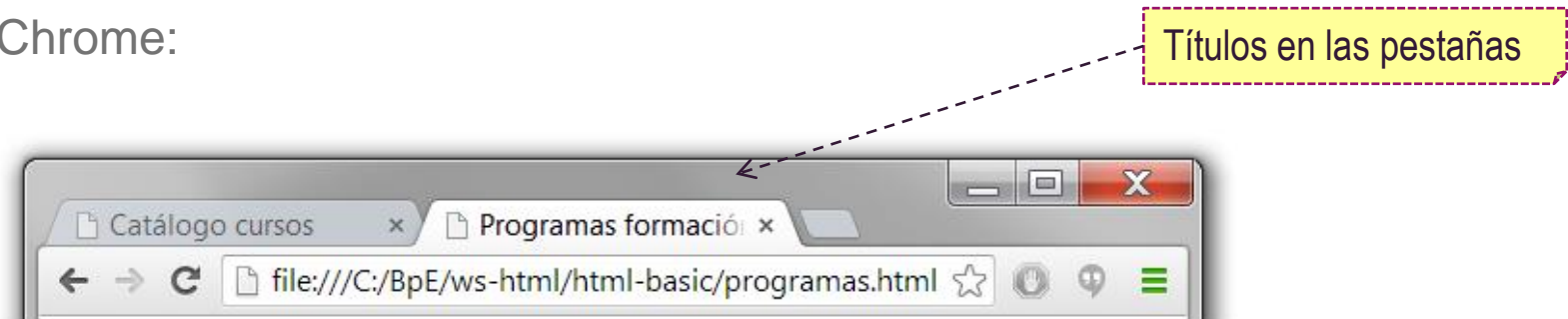
La etiqueta **<title>** se encarga de poner el título del documento, que se suele mostrar en la parte superior del navegador o en la pestaña, dependiendo del navegador utilizado. También es utilizado como identificador en las listas de favoritos o *bookmarks*.

Por ejemplo, se tienen dos páginas con los siguientes títulos:

```
<head>  
  <title>Catálogo cursos</title>  
</head>
```

```
<head>  
  <title>Programas formación</title>  
</head>
```

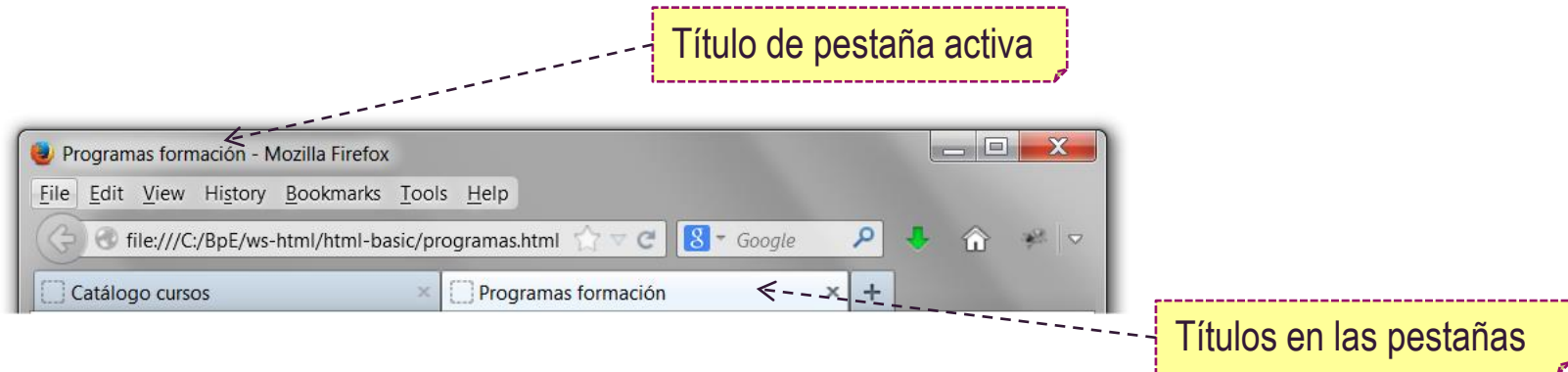
- En Chrome:



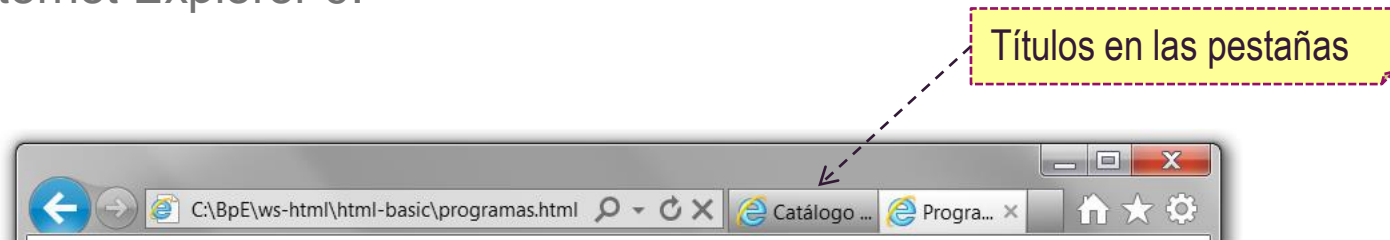
head

título

- En Firefox (depende de modo de visualización):



- En Internet Explorer 9:



head

meta

La etiqueta **<meta>** (metadata) permite especificar información general de la página. Por ejemplo:

- Especificar valores equivalentes a respuestas HTTP de cabecera, para establecer el tipo MIME y juego de caracteres de un documento HTML:

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

Indica que el content tiene como valor el tipo de contenido.

Indica que el contenido es HTML, utilizando nomenclatura MIME

Juego de caracteres del archivo de la página

Nota: en HTML5, esta sentencia se simplifica, como se describe más adelante.

head

meta

- Información sobre refresco de la página, que permite que recargue otra página (o la misma) después de cierto tiempo:

```
<meta http-equiv="refresh" content="30" />
```

Refresca la misma pantalla cada 30 segundos.

```
<meta http-equiv="refresh" content="5, url=programas.html" />
```

Después de 5 segundos, carga otra pantalla según la URL

- Información como el nombre del autor, fecha de expiración, programa con el que se ha diseñado la página, etc. Esto último se hace escribiendo pares nombre/valor:

```
<meta name="author" content="everis" />
```

```
<meta name="description" content="Detalle de programas" />
```

head

link

La etiqueta **<link>** define la relación entre dos documentos vinculados. Se utiliza para enlazar páginas de estilo externas al documento HTML.

```
<link type="text/css" rel="stylesheet" href="styles.css" />
```

Atributos:

- **rel**: Define la relación entre el documento enlazado con el actual. Valores: `stylesheet` para hojas de estilo CSS, y otros como `alternate`, `bookmark`, `chapter`, `help` y `glossary`.
- **type**: Especifica el tipo MIME del documento vinculado. Se utiliza `text/css` para hojas de estilo.
- **href**: La URL del documento vinculado.

3 HTML estático

- inspección de elementos

4. estilos con CSS

5. tablas

6. JavaScript

7. formularios

8. evolución a HTML5

inspección de elementos

descripción

Desde algunos navegadores se pueden **inspeccionar** elementos, lo que permite analizar su contenido y propiedades, verificando si son los que corresponden.

Esto tiene los siguientes beneficios:

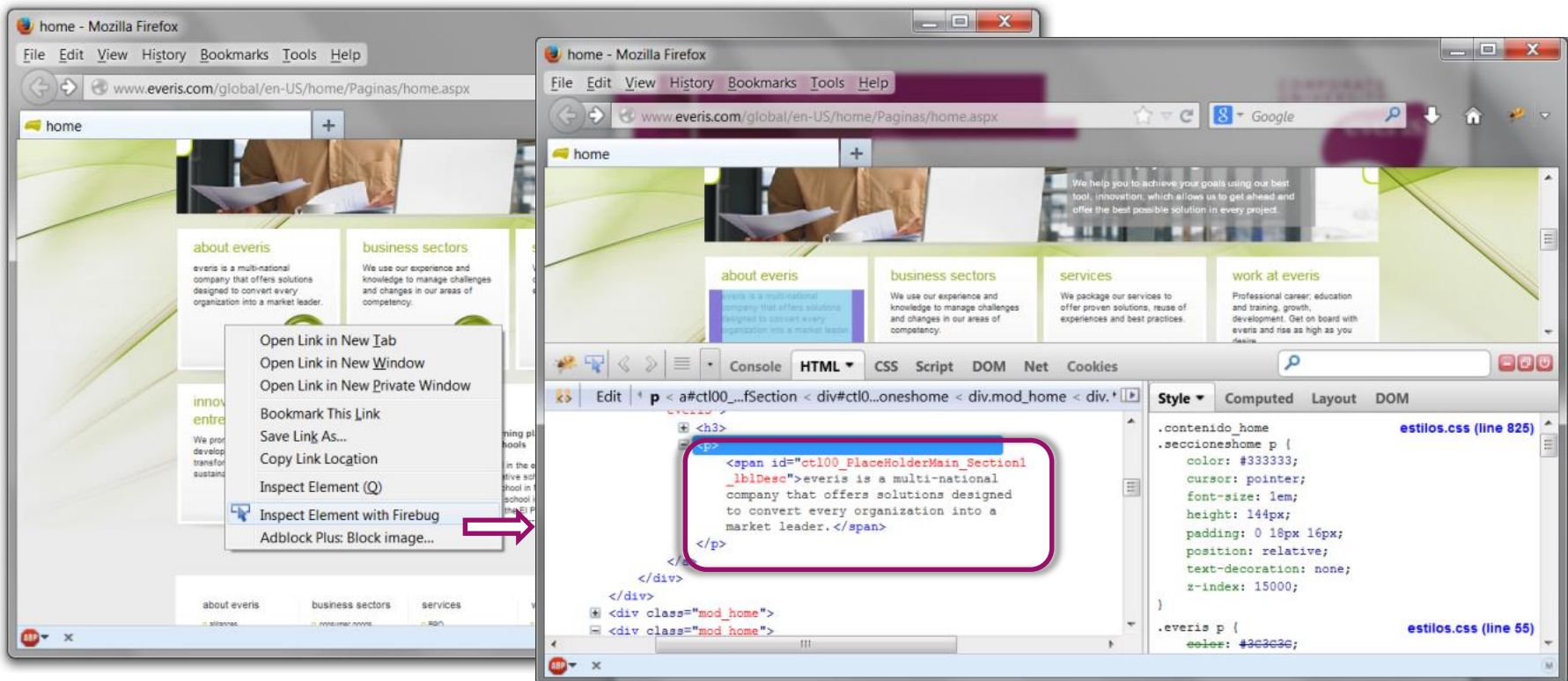
- Verificar si la estructura HTML corresponde a la definida. En páginas estáticas esto se verifica en la programación, pero en las que se generan dinámicamente (HTML dinámico y cursos posteriores) es importante.
- Comprobar si hay errores en la estructura, los cierres de etiquetas, uso de atributos.



inspección de elementos

inspección con Firefox

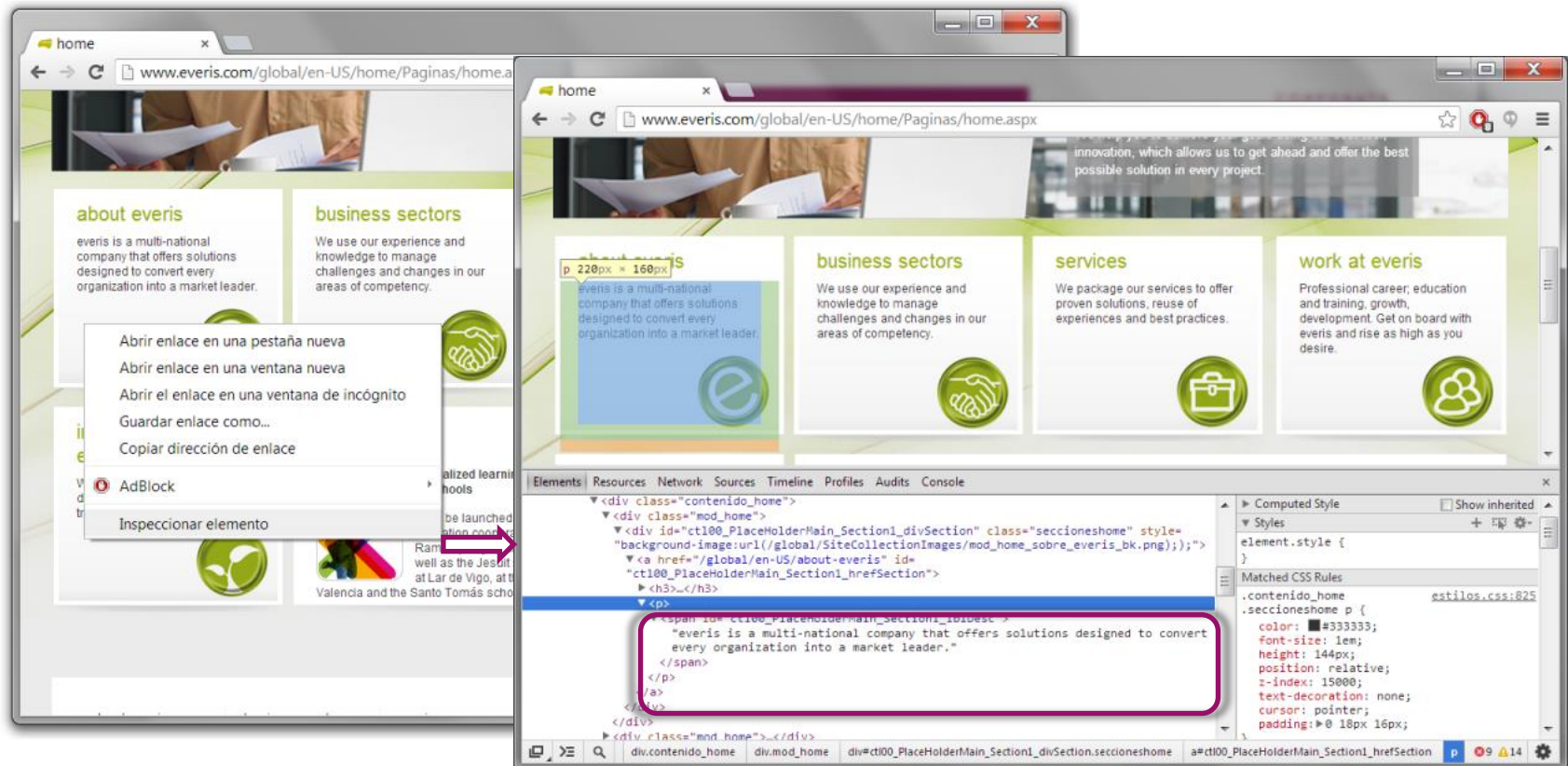
En **Firefox**, se puede inspeccionar elementos en forma nativa, aunque lo recomendable es utilizar el *add-on* llamado Firebug, pues da más posibilidades:



inspección de elementos

inspección con Chrome

En **Chrome**, se pueden inspeccionar elementos de la misma manera:



inspección de elementos

inspección con Internet Explorer 9

Internet Explorer 9 también permite inspeccionar, pero no apuntando a un elemento, sino la página completa. El elemento a inspeccionar se debe marcar posteriormente:



inspección de elementos

caso práctico

Ingresar al sitio:

Obtener el código HTML del elemento:

services

- BPO
- business consulting
- outsourcing
- SAP&ES
- technology

4 estilos con CSS

- **motivación**
- declaración de estilos
- clases
- alguno estilos
- combinación y herencia
- inspección de estilos

5. tablas

6. JavaScript

7. formularios

8. evolución a HTML5

motivación

formato en HTML

En el capítulo anterior, se presentan etiquetas para dar formato al texto contenido en una página HTML:

- Alineación justificada de párrafo.
- Texto en negrita, cursiva, subrayado.

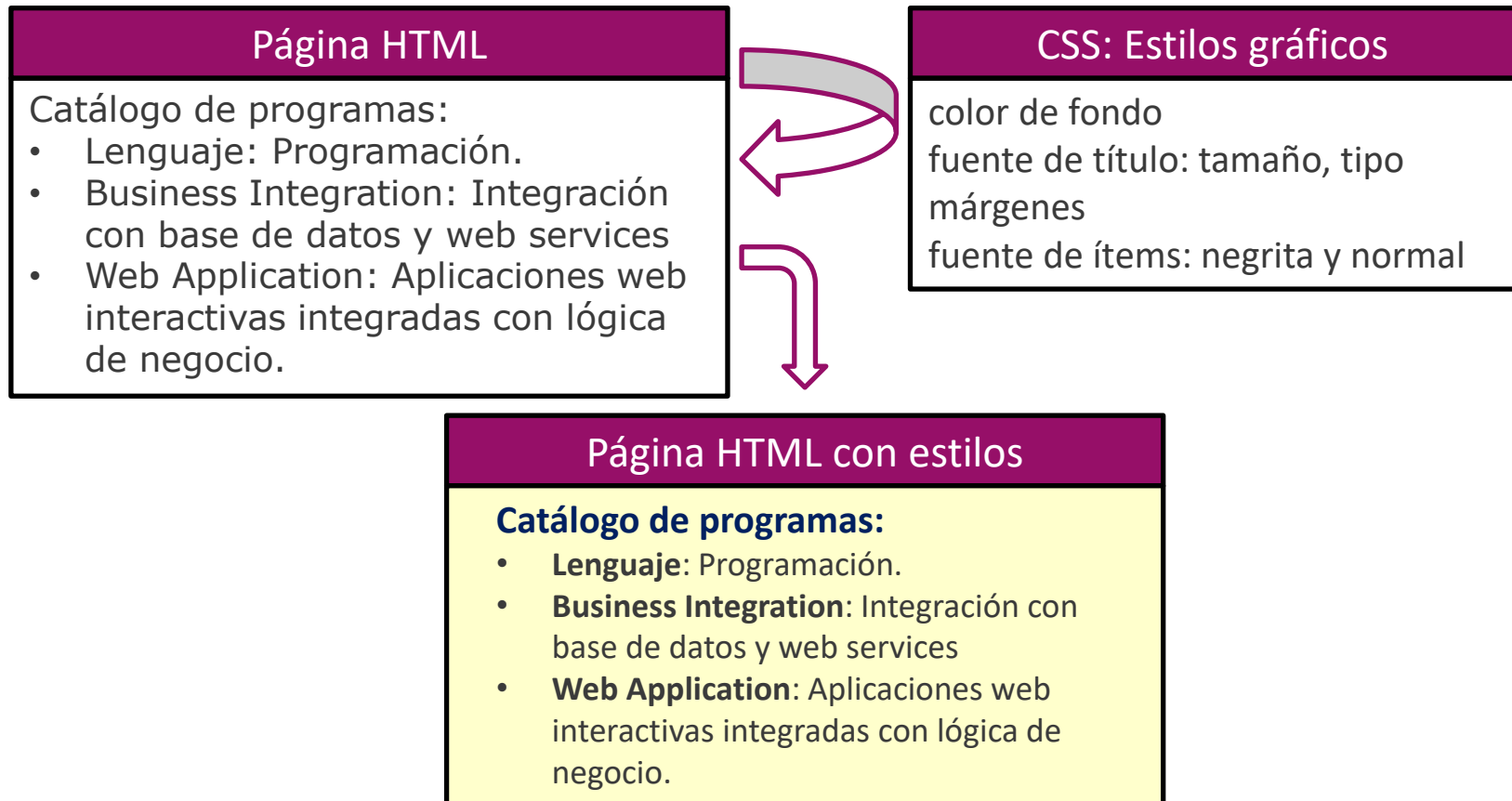
Esta idea para dar formato es de las primeras versiones de HTML, y tiene algunos problemas:

- **Limitado**: no existen suficientes etiquetas para todos los tipos de formato.
- **Poco flexible**: Mezcla en un mismo archivo el contenido y el aspecto gráfico, con lo cual no se puede adaptar para distintos formatos dependiendo de la situación.
- **No reutiliza**: No permite utilizar los mismos formatos en otras partes de la página o en otras páginas, por lo que se tienen que copiar.

motivación

estilos gráficos

Para resolver los problemas anteriores, se definen los estilos gráficos con **CSS**:



4 estilos con CSS

- **declaración de estilos**
- clases
- alguno estilos
- combinación y herencia
- inspección de estilos

5. tablas

6. JavaScript

7. formularios

8. evolución a HTML5

declaración de estilos

ejemplo inicial

Como ejemplo inicial, se utiliza el color del texto. En una página se puede agregar lo siguiente:

```
<head>
```

```
  <style>
```

```
    h1 {
```

```
      color: blue;
```

```
      font-family: Verdana;
```

```
    }
```

```
  </style>
```

```
</head>
```

```
<body>
```

```
  <h1>Título con estilo</h1>
```

```
  <p style="color: maroon">En style se  
    pueden definir estilos gráficos<br>  
    para los elementos HTML</p>
```

```
</body>
```

Nombre del elemento HTML.
Todos los h1 utilizan el estilo.

Se define para los h1 el color azul y la
familia (tipo de letra) Verdana para la fuente.

<p> tiene definido localmente
el color de la fuente

Título con estilo

En style se pueden definir estilos gráficos
para los elementos HTML

declaración de estilos

ejemplo inicial

Nótese la forma como se declaran los estilos en el ejemplo:

Utiliza la etiqueta <style>, a nivel de página.

```
<head>
  <style>
    h1 {
      color: blue;
      font-family: Verdana;
    }
  </style>
</head>
<body>
  <h1>Título con estilo</h1>
  <p style="color: maroon">En style se
    pueden definir estilos gráficos<br>
    para los elementos HTML</p>
</body>
```

La nomenclatura utiliza {llaves} y separación por ';' entre estilos, y ':' para la asignación.

Utiliza el atributo style localmente.

Como se ve a continuación, el uso de <style> es **no recomendado**.

declaración de estilos

dónde se declaran

En el ejemplo, la declaración de los estilos gráficos se realiza en una etiqueta `<style>`, declarada en el head. Esta es una opción, que tiene dos problemas:

- No es reutilizable en otras páginas.
- Junta en un mismo archivo los estilos gráficos y el contenido HTML.

Para evitar esto, se definen los estilos en un archivo separado, llamado hoja de estilos, o **Cascade Style Sheet (CSS)**. La declaración de una hoja de estilos se realiza en el head de la página:

```
<link type="text/css" rel="stylesheet" href="nombre.css" />
```

Referencia a archivo CSS con la hoja de estilos. Puede incluir ruta relativa o URL.

declaración de estilos

ejemplo hoja de estilos

Aplicándolo al ejemplo inicial:

```
<head>
  <link type="text/css"
        rel="stylesheet"
        href="styles.css" />
</head>
<body>
  <h1>Título con estilo</h1>
  <p style="color: maroon">En style se
    pueden definir estilos gráficos<br>
    para los elementos HTML</p> </body>
```

Nótese que se pueden utilizar estilos locales para casos particulares.

styles.css

```
h1 {
  color: blue;
  font-family: Verdana;
}
```

Título con estilo

En style se pueden definir estilos gráficos para los elementos HTML

4 estilos con CSS

- **clases**
- algunos estilos
- combinación y herencia
- inspección de estilos

5. tablas

6. JavaScript

7. formularios

8. evolución a HTML5

clases

selector

En el ejemplo inicial, se define el estilo para las etiquetas <h1>. Cuando se definen estilos generales para las etiquetas, se habla de "selector". Otros ejemplos para etiquetas:

```
p {  
  color: maroon;  
}  
  
a {  
  text-decoration: none;  
}  
  
fieldset {  
  padding: 20px;  
}
```

Todos los párrafos con color marrón

Hyperlink sin texto subrayado

Espacio entre el borde y el contenido del fieldset

clases

definición

Si se quiere que sólo algunos `<h1>` tengan un estilo definido, no se puede utilizar un selector, pues los define para todos. En ese caso, se debe utilizar una **clase**:

Nombre de clase. Se marca con un punto

```
.hred {  
    color: red;  
}  
  
.hblue {  
    color: blue;  
}
```

Nombre de clase.

```
<h3 class="hred">Catálogo de cursos Java</h3>  
<h3 class="hblue">Catálogo de cursos .NET</h3>
```

Catálogo de cursos Java

Catálogo de cursos .NET

clases

utilización de id

Otra opción, utilizada en situaciones específicas, es asociar un estilo a un identificador del elemento HTML. En el ejemplo anterior, queda:

Estilo asociado a id.
Se marca con #.

```
#hjava {  
    color: red;  
}  
  
#hnet {  
    color: blue;  
}
```

id del elemento

```
<h3 id="hjava">Catálogo de cursos Java</h3>  
<h3 id="hnet">Catálogo de cursos .NET</h3>
```

Catálogo de cursos Java

Catálogo de cursos .NET

clases

comentarios

En CSS, la nomenclatura de comentarios es la siguiente:

```
/* bloque comentado */
```

El bloque comentado puede tener una o más líneas, y cualquier contenido. Se utilizan para proporcionar información adicional que facilita el entendimiento:

```
/*  
  Cuando un link es visitado recupera el color original  
*/  
a:visited { text-decoration: none; }
```

No existen restricciones en el contenido de los comentarios en CSS.

4 estilos con CSS

- **algunos estilos**
- combinación y herencia
- inspección de estilos

5. tablas

6. JavaScript

7. formularios

8. evolución a HTML5

algunos estilos

color

Se pueden definir colores en distintos ámbitos, utilizando atributos de estilo:

- **color**: Color de fuente (texto)
- **background-color**: Color de fondo
- **border-color**: Color de bordes

La definición de un color se puede hacer de varias maneras:

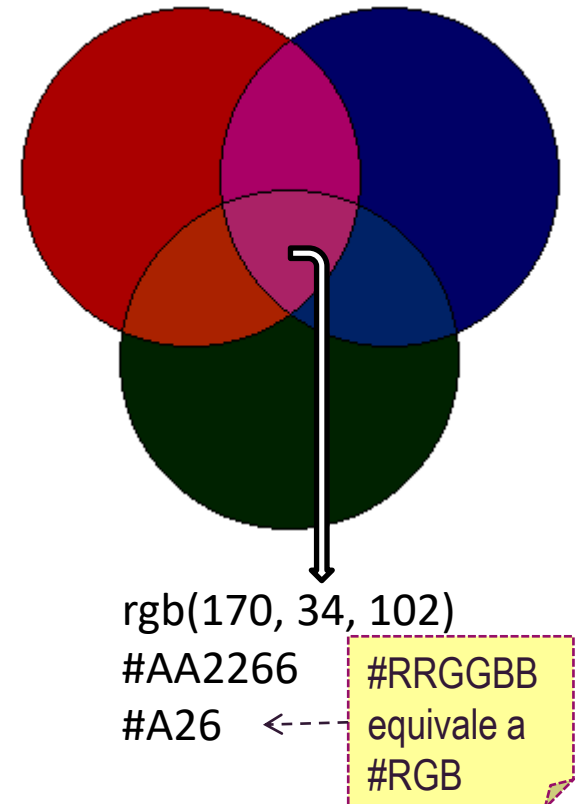
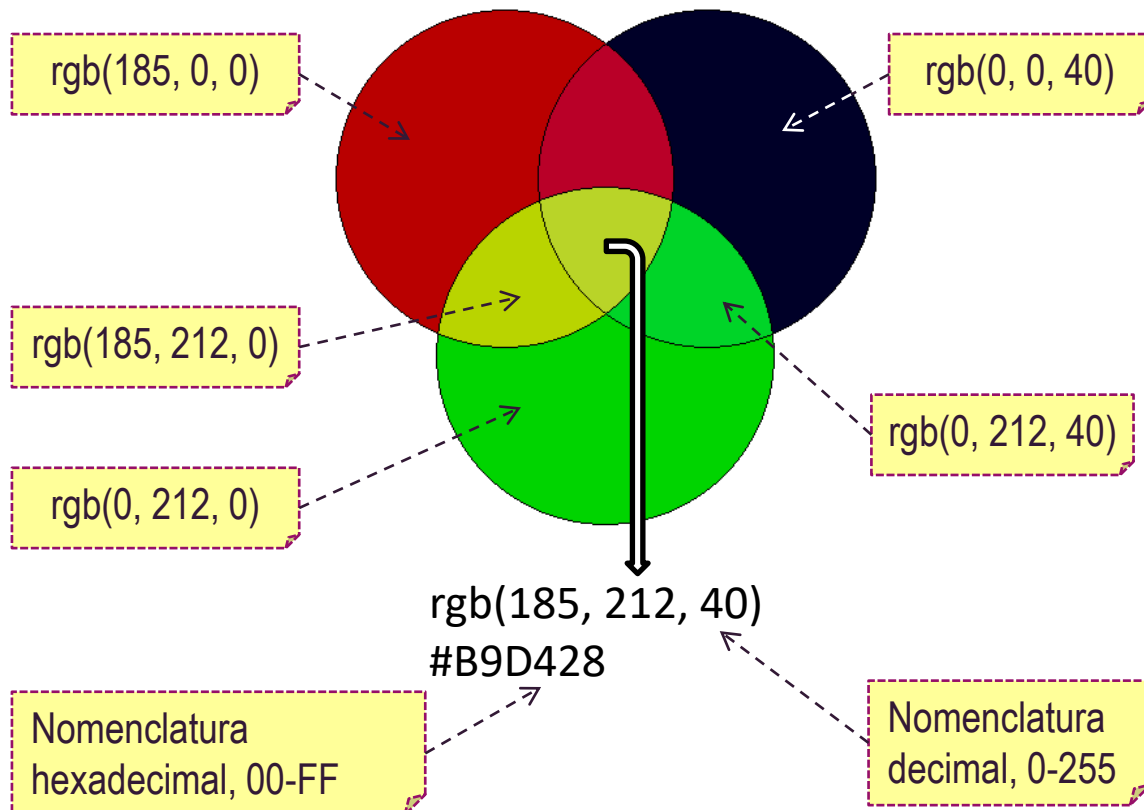
- Directamente con el nombre (en inglés), definido para algunos colores:



algunos estilos

color

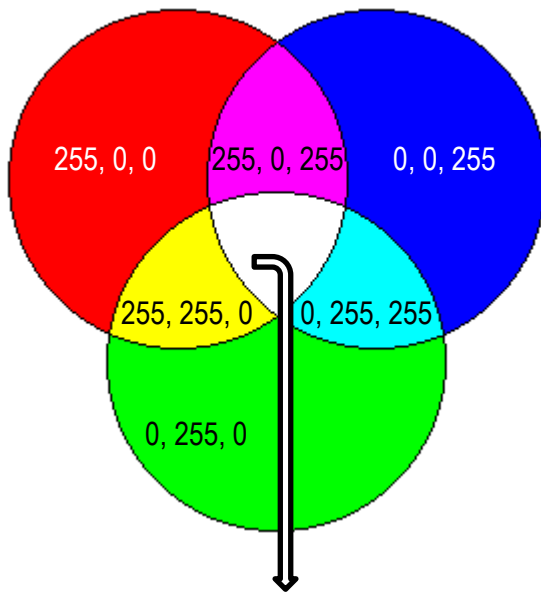
- A través de una combinación de los colores básicos RGB (Red-Green-Blue):



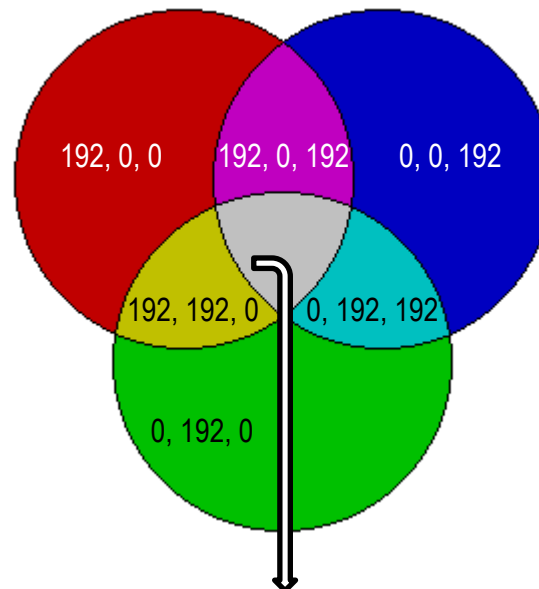
algunos estilos

color

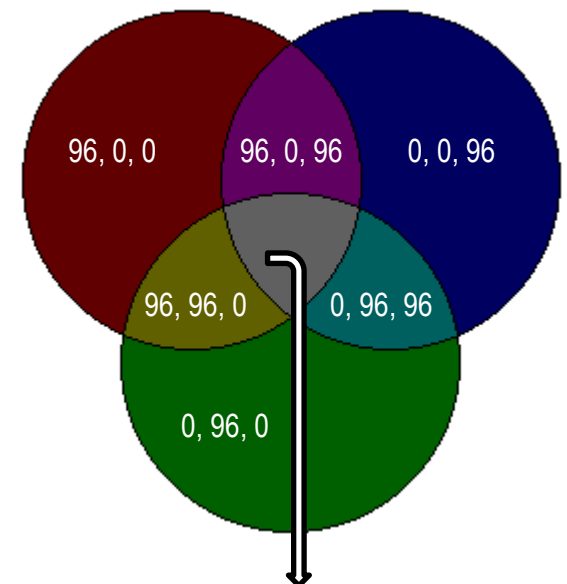
Si el nivel de RGB (Red-Green-Blue) es el mismo en los tres colores, equivale a niveles de grises (el negro es #000000):



`rgb(255, 255, 255)`
`#FFFFFF`
`#FFF`



`rgb(192, 192, 192)`
`#C0C0C0`



`rgb(96, 96, 96)`
`#606060`

algunos estilos

body

Cuando se definen estilos para `<body>`, se aplican a **todo el contenido HTML**.

- Por ejemplo, si se quiere definir que todo el documento utilice la fuente Arial, y de color "Navy", la definición en la hoja de estilos es:

```
body {  
    background-color: #F8F8F8;  
    font-family: Arial;  
}
```

- Aplicándolo al caso inicial, el párrafo `<p>` utiliza el estilo de font-family. El encabezado no lo utiliza, ya que tiene estilo propio:

Título con estilo

En style se pueden definir estilos gráficos para los elementos HTML

Utiliza font-family definida para `<body>`, pues no hay una para `<p>`.

algunos estilos

body

Para el `<body>`, y para cualquier otro elemento que pueda contener a otros, se pueden definir el color de fondo o una imagen de fondo:

```
body {  
    background-color: #EAEAEAC;  
}
```

Catálogo de cursos Java

Catálogo de cursos .NET

```
body {  
    background-image:  
        url(img_home_bk.jpg);  
    background-repeat: no-repeat;  
}
```

Catálogo de cursos Java

Catálogo de cursos .NET

algunos estilos

fuentes

Para un párrafo `<p>`, o para otro elemento que contenga texto, se pueden definir estilos para la fuente:

font-family	font-size	font-weight	font-style	font-variant
Arial	0.9em 9pt	400 normal	normal	normal
Verdana	1.0em 11pt	700 bold	<i>italic</i>	SMALL-CAPS
Sans-serif	1.4em 15pt	lighter	<i>oblique</i>	

algunos estilos

textos

Para un párrafo `<p>`, o para otro elemento que contenga texto, se pueden definir estilos para decorar, alinear o transformar el texto:

text-align	text-decoration	text-indent	text-shadow	text-transform
left	<u>underline</u>	0px	0em 0.13em gray	none
center	blink	5px	0.13em 0em gray	lowercase
right	line-through	10px	0.12em 0.12em gray	UPPERCASE
A small text with align justify	<u>overline</u>	20px	0.12em 0.12em red	Capitalize First Letter

algunos estilos

hyperlink

Para el hyperlink `<a>`, se pueden definir estilos en función de su estado:

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

Estilos para el hyperlink.
En este caso, se suprime el subrayado.

```
a:link { }
```

Estilos cuando aún no ha sido visitado.

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

Estilos cuando ya ha sido visitado. En este caso, se elimina el subrayado y recupera el color original.

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

Estilos cuando se pasa el puntero sobre el hyperlink.

```
a:focus, a:active { text-decoration: underline; }
```

Estilos cuando tiene el foco (focus), o cuando se presiona (active).

algunos estilos

lista

En HTML se presentan las listas, con el atributo type que está en desuso. A través de estilos, se pueden manejar el aspecto de la numeración o bullets de las listas. Ejemplos de listas ordenadas:

```
<ol style="list-style: upper-roman">
```

```
  <li>Java Básico</li>
  <li>Java Intermedio</li>
  <li>Spring Core</li>
  <li>Hibernate-JPA</li>
  <li>Web Services</li>
</ol>
```

- I. Java Básico
- II. Java Intermedio
- III. Spring Core
- IV. Hibernate-JPA
- V. Web Services

```
<ol style="list-style: lower-alpha">
```

```
  <li>Java Básico</li>
  <li>Java Intermedio</li>
  <li>Spring Core</li>
  <li>Hibernate-JPA</li>
  <li>Web Services</li>
</ol>
```

- a. Java Básico
- b. Java Intermedio
- c. Spring Core
- d. Hibernate-JPA
- e. Web Services

```
<ol style="list-style: decimal">
```

```
  <li>Java Básico</li>
  <li>Java Intermedio</li>
  <li>Spring Core</li>
  <li>Hibernate-JPA</li>
  <li>Web Services</li>
</ol>
```

1. Java Básico
2. Java Intermedio
3. Spring Core
4. Hibernate-JPA
5. Web Services

algunos estilos

lista

En HTML se presentan las listas, con el atributo type que está en desuso. A través de estilos, se pueden manejar el aspecto de la numeración o bullets de las listas. Ejemplos de listas ordenadas:

```
<ul style="list-style: square">
```

```
  <li>C# Básico</li>
  <li>C# Avanzado</li>
  <li>everNExT</li>
  <li>Persistencia</li>
  <li>Servicios Web</li>
</ul>
```

- C# Básico
- C# Avanzado
- everNExT
- Persistencia
- Servicios Web

```
<ul style="list-style: circle">
```

```
  <li>C# Básico</li>
  <li>C# Avanzado</li>
  <li>everNExT</li>
  <li>Persistencia</li>
  <li>Servicios Web</li>
</ul>
```

- C# Básico
- C# Avanzado
- everNExT
- Persistencia
- Servicios Web

```
<ul style="list-style: url(go.png)">
```

```
  <li>C# Básico</li>
  <li>C# Avanzado</li>
  <li>everNExT</li>
  <li>Persistencia</li>
  <li>Servicios Web</li>
</ul>
```

- ▶ C# Básico
- ▶ C# Avanzado
- ▶ everNExT
- ▶ Persistencia
- ▶ Servicios Web

algunos estilos

sección

Si se requiere aplicar estilos a una sección en particular de una página, una opción bastante utilizada es a través de un `<div>` o ``.

Por ejemplo, una sección cuya fuente es de tipo Arial, y con color de fondo "wheat" (uno de los que no son principales):

```
.secred {  
  font-family: Arial;  
  background-color: wheat;  
}
```

```
<div class="secred">  
  <h3>Título</h3>  
  <p>Contenido</p>  
</div>
```

Título

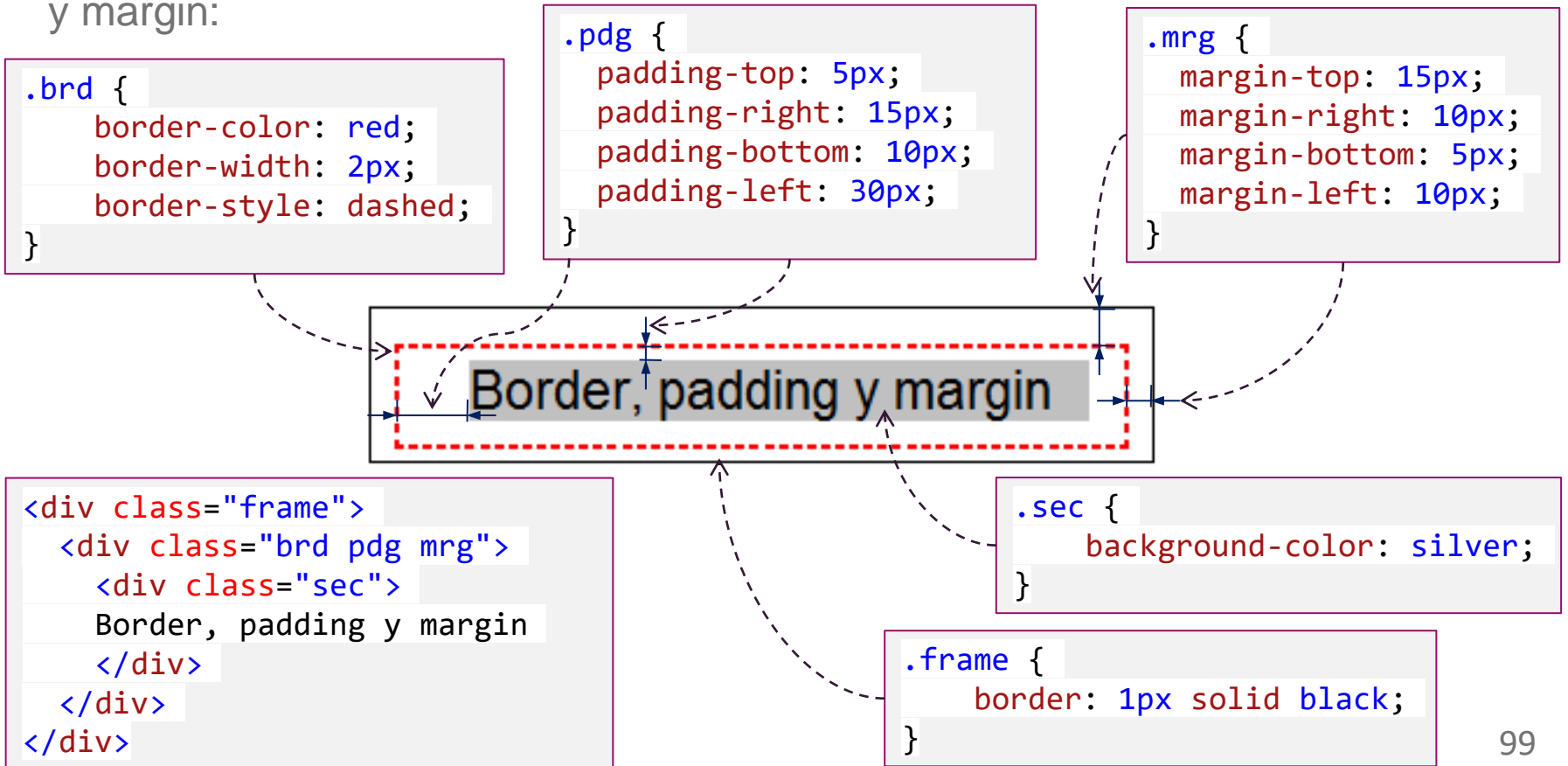
Contenido

A continuación, se presentan los detalles de bordes, márgenes y otras características donde normalmente se utilizan las secciones de página.

algunos estilos

border, padding, margin

A un elemento HTML se le pueden definir los atributos gráficos border, padding y margin:



algunos estilos

border, padding, margin

Aplicando distintas combinaciones:

```
<div class="frame">
  <div class="brd">
    <div class="sec">Border</div>
  </div></div><br/>
<div class="frame">
  <div class="brd pdg">
    <div class="sec">Border y padding</div>
  </div></div><br/>
<div class="frame">
  <div class="brd pdg mrg">
    <div class="sec">Border, padding y margin</div>
  </div></div><br/>
<div class="frame">
  <div class="pdg mrg">
    <div class="sec">Padding y margin</div>
  </div></div><br/>
<div class="frame">
  <div class="mrg">
    <div class="sec">Margin</div>
  </div></div>
```

Border

Border y padding

Border, padding y margin

Padding y margin

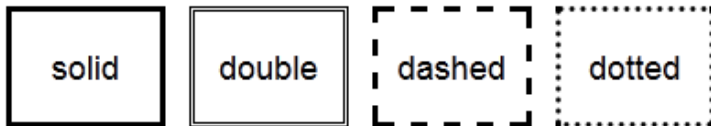
Margin

algunos estilos

border

Opciones para border:

- style:



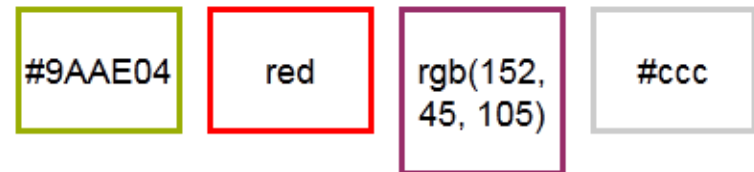
```
border-style: solid;
```

```
border-style: double;
```

```
border-style: dashed;
```

```
border-style: dotted;
```

- color:



```
border-color: #9AAE04;
```

```
border-color: red;
```

```
border-color: rgb(152, 45, 105);
```

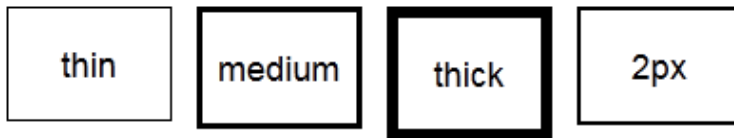
```
border-color: #ccc;
```

algunos estilos

border

Opciones para border:

- width:



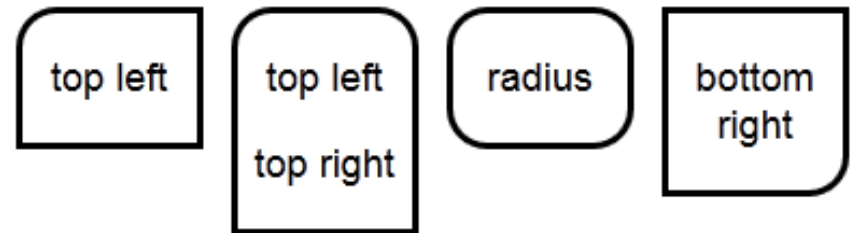
```
border-width: thin;
```

```
border-width: medium;
```

```
border-width: thick;
```

```
border-width: 2px;
```

- radius:



```
border-top-left-radius: 15px;
```

```
border-top-left-radius: 15px;  
border-top-right-radius: 15px;
```

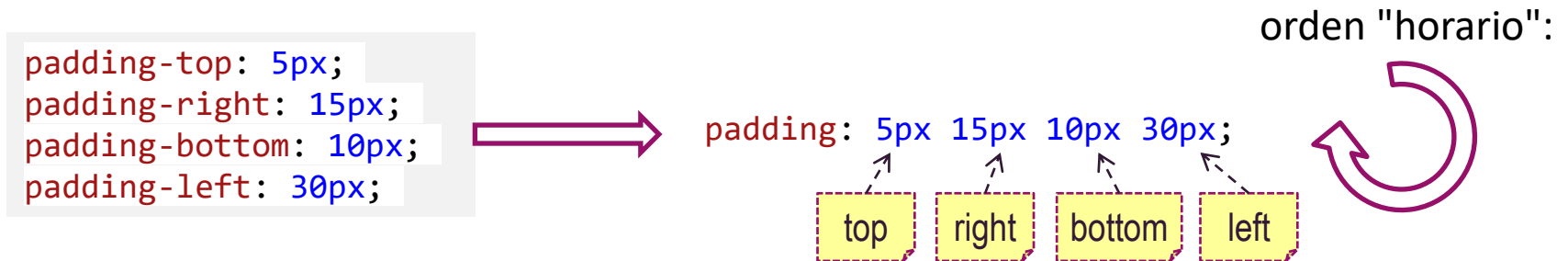
```
border-radius: 15px;
```

```
border-bottom-right-radius: 15px;
```

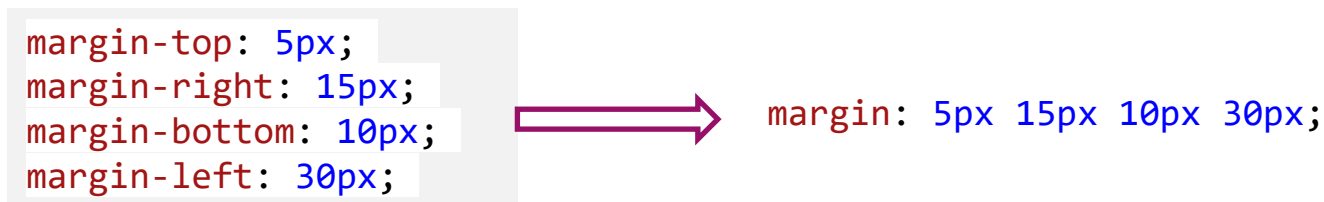
algunos estilos

shorthand

Algunas definiciones de estilos tienen una nomenclatura más compacta, llamada **shorthand**. Por ejemplo, el padding:



El caso del margin es similar:




algunos estilos

shorthand

Si los cuatro valores son iguales, se puede colocar uno:


```
padding-top: 10px;  
padding-right: 10px;  
padding-bottom: 10px;  
padding-left: 10px;
```




```
padding: 10px;
```

Similar es el caso de top-bottom y de left-right:

```
margin-top: 5px;  
margin-right: 15px;  
margin-bottom: 5px;  
margin-left: 15px;
```



```
margin: 5px 15px;
```



algunos estilos

shorthand

El caso del border también tiene un *shorthand*:

```
border-color: red;  
border-width: 2px;  
border-style: dashed;
```



```
border: red 2px dashed;
```



El orden no importa.

En el caso de las fuentes, también:

```
font-size: 10pt;  
font-family: Arial, Helvetica, sans-serif;  
line-height: 1.2em;
```



```
font: 10pt/1.2em Arial, Helvetica, sans-serif;
```



font: font-style font-variant font-weight font-size/line-height font-family

algunos estilos

cursor

A una sección del documento HTML se le puede cambiar la forma del cursor, utilizando estilos. Esto mejora la interacción con el usuario, indicando por ejemplo que debe esperar, o donde debe presionar:

`cursor: pointer;`



`cursor: e-resize;`



`cursor: help;`



`cursor: n-resize;`



`cursor: crosshair;`



`cursor: move;`



`cursor: wait;`



`cursor: progress;`



algunos estilos

Caso práctico 4-1: aplicación de estilos

- Resumen del ejercicio:
- Utilizar el proyecto "cp-curso".
- Crear una página HTML vacía, utilizando la plantilla HTML 4.01 transitional.
- Crear una carpeta "img", dentro de proyecto de la página.
- Ingresar al sitio:
- <http://www.everis.com>
- Utilizando la inspección de elementos, obtener la información de la URL de la imagen de fondo.
- Extraer y guardar la imagen de fondo en el directorio "img".
- Utilizando CSS, colocar como imagen de fondo de la pantalla la imagen obtenida.
- Agregar un hyperlink a la página, utilizando el logo de everis, que redirija al sitio de everis antes mencionado, en una nueva pestaña.

4 estilos con CSS

- **combinación y herencia**
- inspección de estilos

5. tablas

6. JavaScript

7. formularios

8. evolución a HTML5

combinación y herencia

combinación

Si se quiere que un elemento HTML tenga los estilos gráficos definidos en varias clases, se pueden combinar declarándolas juntas:

```
<element class="class1 class2 class3">...</element>
```

Las clases de estilo se separan por espacio.

combinación y herencia

combinación

Ejemplo:

```
.fnt1 {  
    color: blue;  
}  
.fnt2 {  
    font-family: Arial, Helvetica, sans-serif;  
    font-size: 14pt;  
}
```

```
<p class="fnt1">Fuente 1</p>  
<p class="fnt2">Fuente 2</p>  
<p class="fnt1 fnt2">Fuente 1 2</p>
```

Fuente 1

Fuente 2

Fuente 1 2

Combina atributos de las dos clases de estilo.

combinación y herencia

Caso práctico:

Combinar estilos para definir propiedades comunes y particulares de tres secciones de una página.

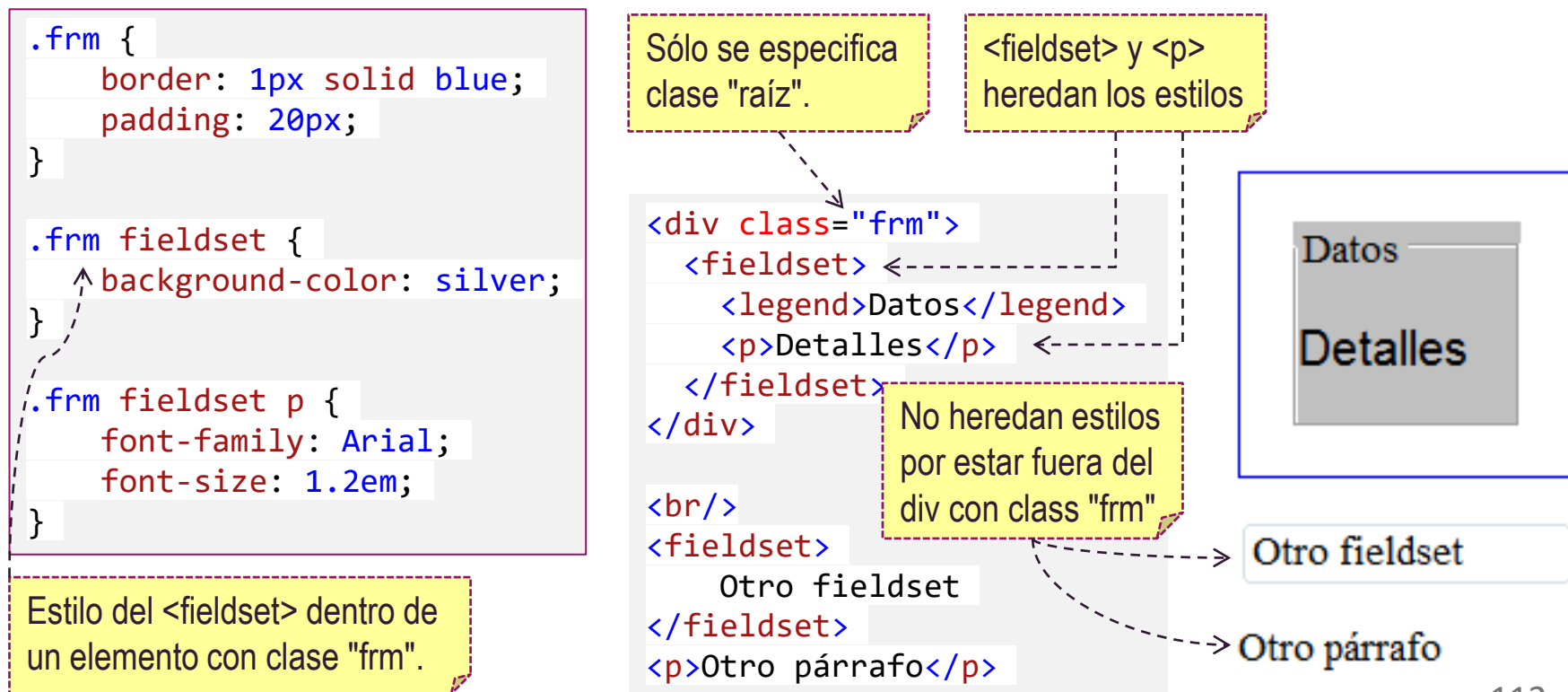
Utilizar la herramienta de edición HTML y CSS y averiguar lo siguiente:

- Si dos clases de estilo se llaman igual, y tienen distintos atributos definidos, qué sucede.
- Si dos clases de estilo se llaman igual, y tienen el mismo atributo con diferentes valores, determinar cuál vale.
- Verificar si los navegadores Chrome, Firefox e Internet Explorer tienen diferencias en los puntos anteriores.

combinación y herencia

herencia

En una estructura de etiquetas HTML, a través de CSS se pueden definir sus estilos, de modo que sólo se asigna la clase del elemento raíz. Ejemplo:



combinación y herencia

herencia

También se pueden especificar estilos de subelementos dado su *id*, utilizando la nomenclatura con #:

```
.frm {  
    border: 1px solid blue;  
    padding: 20px;  
}  
.frm fieldset {  
    background-color: silver;  
}  
.frm fieldset #instr {  
    font-family: Verdana;  
    font-size: 1.2em;  
    color: navy;  
}  
.frm fieldset #details {  
    font-family: Arial;  
    font-size: 1.0em;  
    color: red;  
}
```

```
<div class="frm">  
  <fieldset>  
    <legend>Datos</legend>  
    <p id="instr">Instrucciones</p>  
    <p id="details">Detalles</p>  
  </fieldset>  
</div>
```



combinación y herencia

toolbar con lista

Para las listas no ordenadas ``, se pueden definir estilos que permiten tener un aspecto gráfico similar a una barra de herramientas (toolbar). Aplicando herencia y los estilos vistos, resulta:

```
ul { list-style: none; }
.tbar {
  width: 28%
}
.tbar ul li {
  float: left;
  padding: 2%;
  border: 2px solid #fff;
}
.tbar ul li:hover {
  border: 2px solid #ccc;
}
.tbar ul li a {
  padding: 2px; width: 100%;
}
.tbar ul li a img {
  border: 0;
}
```

Elimina bullets

Distribución horizontal de los

Mismo ancho


Paso de puntero sobre elemento

Quita el borde (puesto por IE)

```
<div class="tbar">
  <ul>
    <li><a href="#" title="View"></a></li>
    <li><a href="#" title="Export xls"></a></li>
    <li><a href="#" title="New"></a></li>
  </ul>
</div>
```

tooltip

Hyperlink a imagen. Simula un botón.



4 estilos con CSS

- inspección de estilos

5. tablas

6. JavaScript

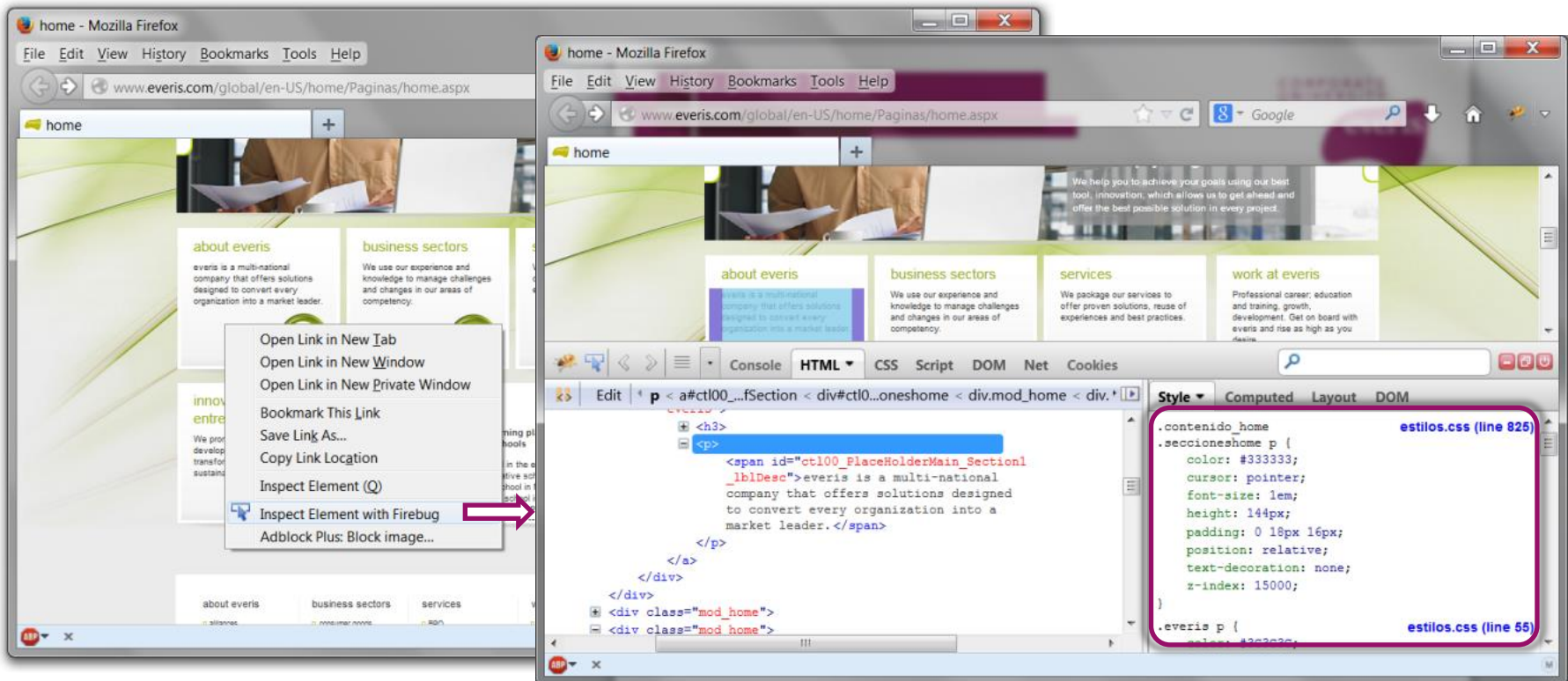
7. formularios

8. evolución a HTML5

inspección de estilos

inspección con Firefox

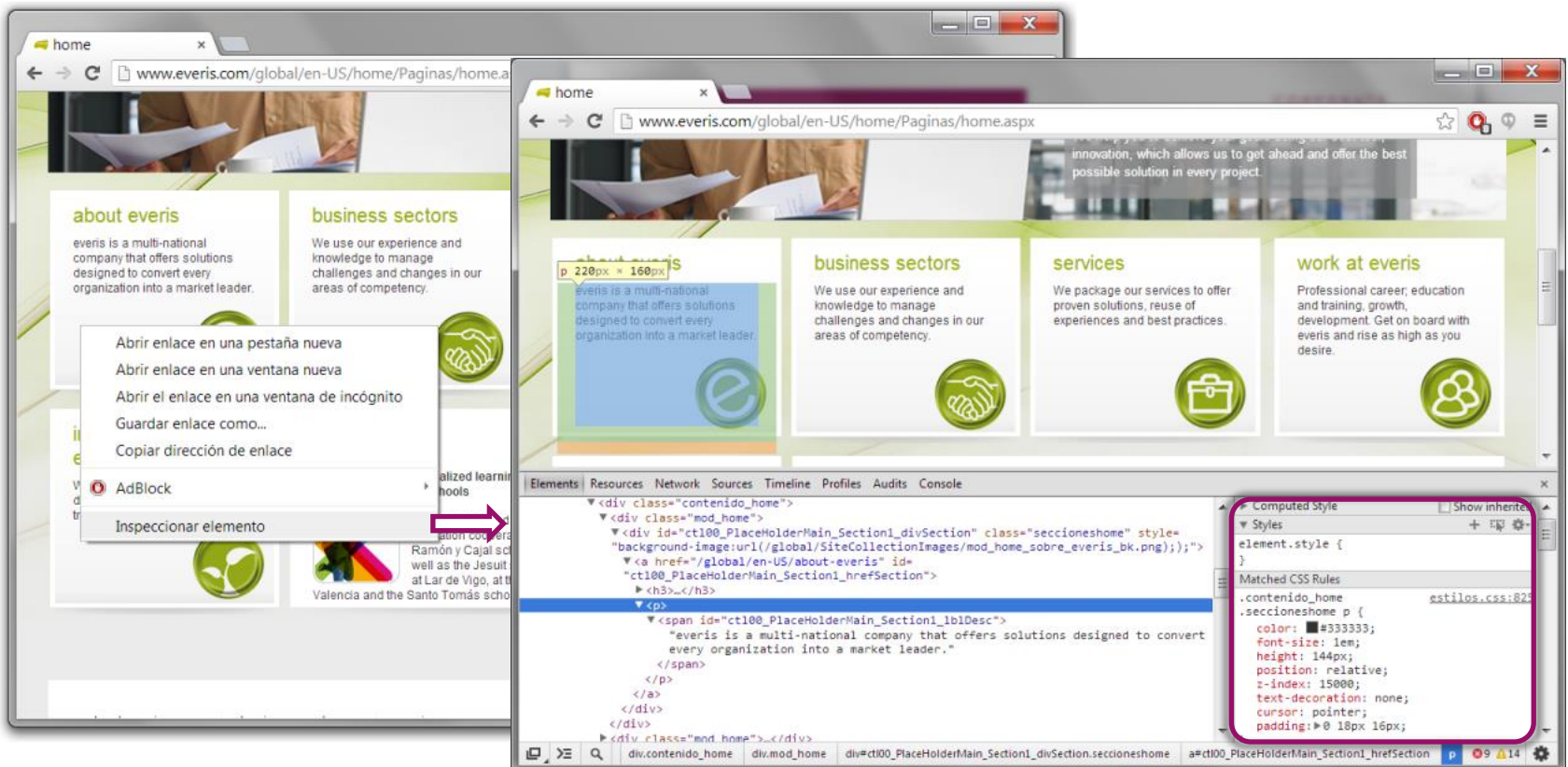
En **Firefox**, se puede inspeccionar elementos en forma nativa, aunque lo recomendable es utilizar el *add-on* llamado Firebug, pues da más posibilidades:



inspección de estilos

inspección con Chrome

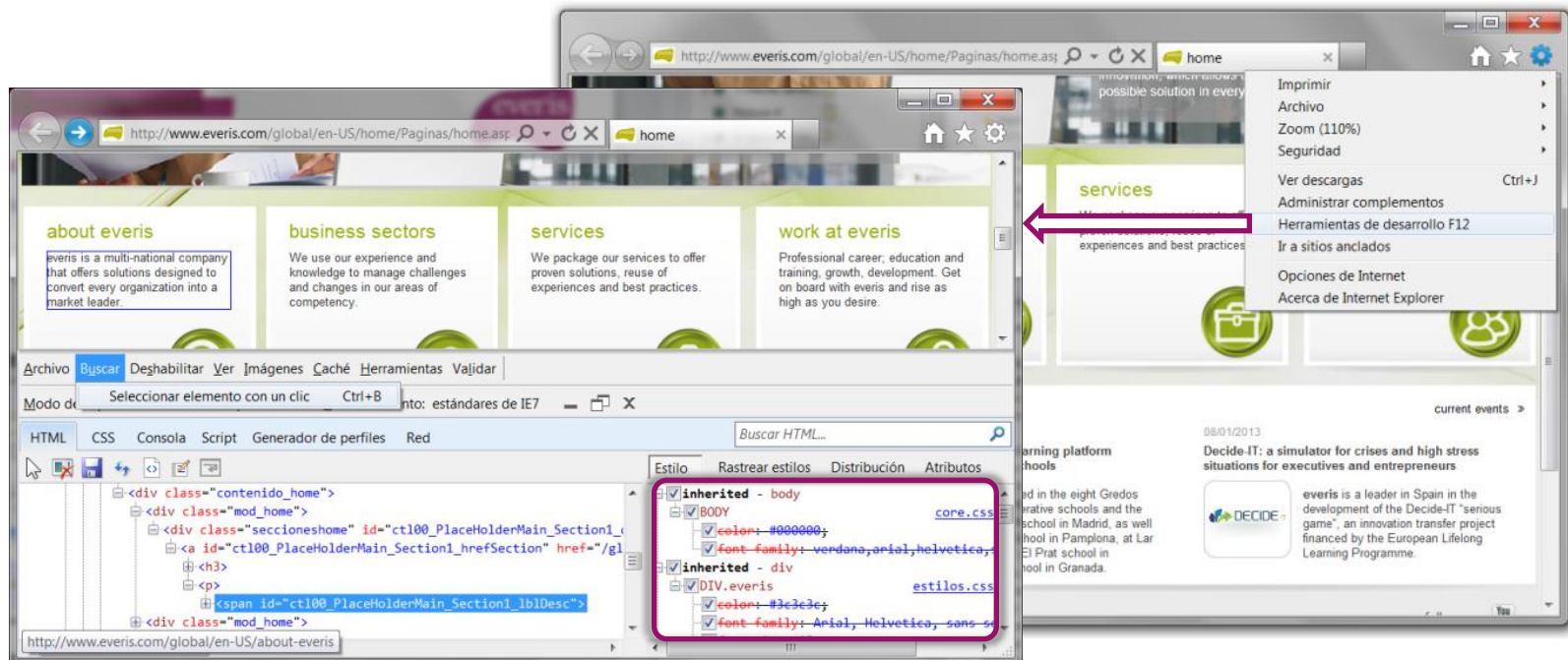
En Chrome, se pueden inspeccionar elementos de la misma manera:



inspección de elementos

inspección con Internet Explorer 9

Internet Explorer 9 también permite inspeccionar, pero no apuntando a un elemento, sino la página completa. El elemento a inspeccionar se debe marcar posteriormente:




inspección de estilos

análisis

En las herramientas de inspección, se observan todos los elementos que participan en la definición del estilo del elemento, y cuáles agregan o sobrescriben atributos. El estilo final es el resultado de la combinación de todos (*merge*):

about everis

everis is a multi-national company that offers solutions designed to convert every organization into a market leader.



Ejemplo con Chrome

```
.contenido_home .seccioneshome span {  
  font-weight: normal !important;  
}  
  
.everis p span {  
  font-weight: bold;  
}  
  
.everis html, .everis body, .everis span, .everis  
object, .everis iframe, .everis h1, .everis h2, .everis h3, .everis  
h4, .everis h5, .everis h6, .everis p, .everis blockquote, .everis  
pre, .everis a, .everis abbr, .everis acronym, .everis address,  
.everis code, .everis del, .everis dfn, .everis img, .everis q,  
.everis dl, .everis dt, .everis dd, .everis ol, .everis ul, .everis  
li, .everis fieldset, .everis form, .everis label, .everis legend,  
.everis table, .everis caption, .everis tbody, .everis tfoot, .everis  
thead, .everis tr, .everis th, .everis td {  
  margin: 0;  
  padding: 0;  
  border: 0;  
  font-weight: inherit;  
  font-style: inherit;  
  font-size: 100%;  
  font-family: inherit;  
}  
  
Inherited from p  
.contenido_home .seccioneshome p {  
  color: #333333;  
  font-size: 1em;  
  cursor: pointer;  
}  
  
.everis p {  
  font-size: 1em;  
  color: #333333;  
}
```

font-weight utilizado

font-weight ignorado

font-size utilizado

font-size ignorado

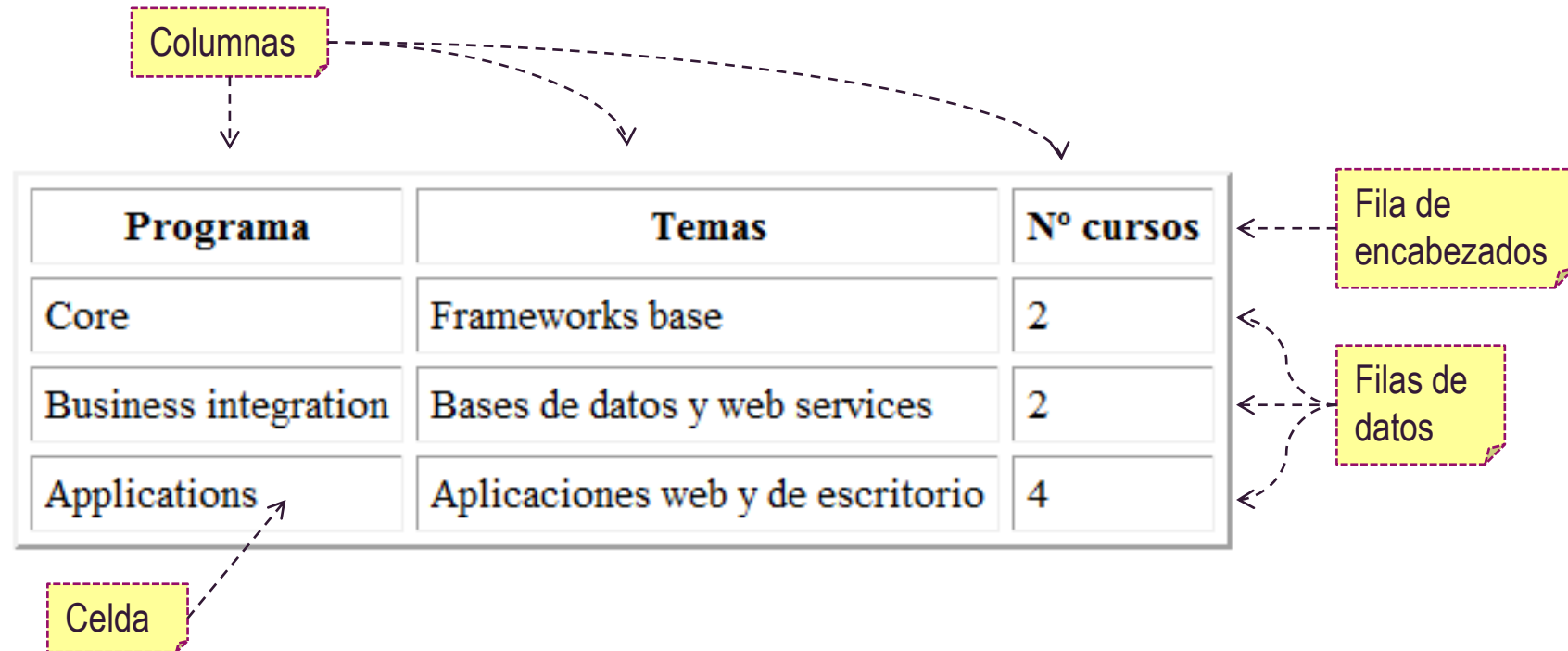
5 tablas

- 6. JavaScript
- 7. formularios
- 8. evolución a HTML5

tablas

estructura

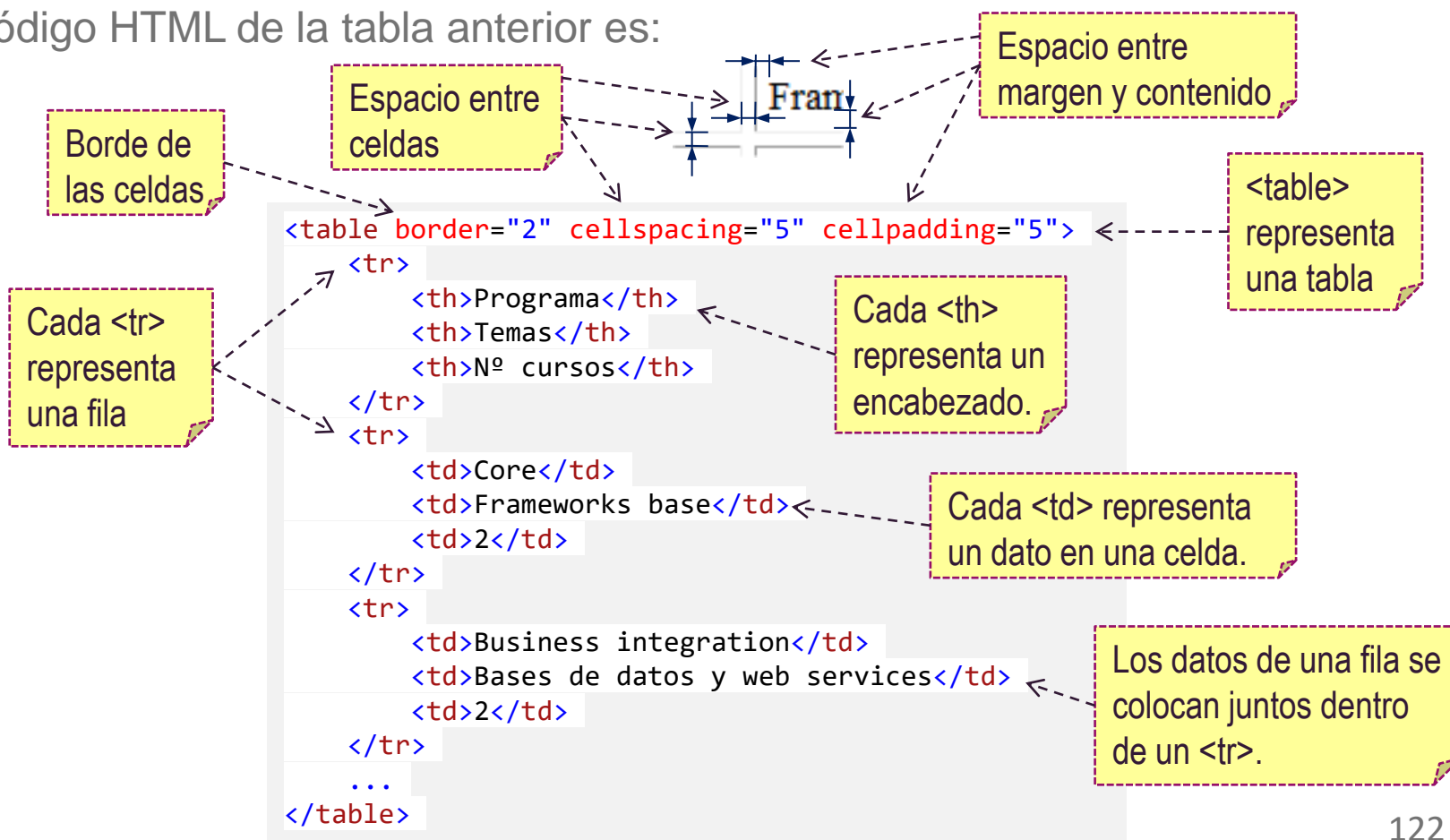
Una tabla es un tipo de elemento HTML que tiene los siguientes componentes:



tablas

estructura

El código HTML de la tabla anterior es:



tablas

estructura

A la tabla se le puede agregar un título (caption):

```
<table border="2" cellspacing="5" cellpadding="5">  
  <caption>Programas de formación</caption>  
  <tr>  
    <th>Programa</th>  
    <th>Temas</th>  
    <th>Nº cursos</th>  
  </tr>  
  <tr>  
    <td>Core</td>  
    <td>Frameworks base</td>  
    <td>2</td>  
  </tr>  
  <tr>  
    <td>Business integration</td>  
    <td>Bases de datos y web services</td>  
    <td>2</td>  
  </tr>  
  ...  
</table>
```

Programas de formación

Programa	Temas	Nº cursos
Core	Frameworks base	2
Business integration	Bases de datos y web services	2
Applications	Aplicaciones web y de escritorio	4

tablas

estructura

Se pueden combinar celdas en una fila o una columna:

```
<table border="2" cellspacing="5" cellpadding="5">
  <caption>Cursos de formación</caption>
  <tr>
    <th rowspan="2">Programa</th>
    <th colspan="2">Curso</th>
  </tr>
  <tr>
    <th>Nombre</th>
    <th>Duración</th>
  </tr>
  <tr>
    <td rowspan="2">Business integration</td>
    <td>Persistencia</td>
    <td>24 horas</td>
  </tr>
  <tr>
    <td>Web services</td>
    <td>24 horas</td>
  </tr>
</table>
```

Cursos de formación

Programa	Curso	
	Nombre	Duración
Business integration	Persistencia	24 horas
	Web services	24 horas

tablas

estilos

A las tablas se les puede agregar estilos gráficos, utilizando CSS. Por ejemplo:

```
table {  
  border: medium solid #6B002A;  
  caption-side: bottom;  
}  
td, th {  
  font-family: Arial;  
  border: thin dotted blue;  
  padding-left: 15px;  
}  
caption {  
  padding-top: 20px;  
  font-style: italic;  
}
```

Programa	Curso	
	Nombre	Duración
	Persistencia	24 horas
Business integration	Web services	24 horas

Cursos de formación

tablas

Caso práctico 5-1: creación de tablas

- Resumen del ejercicio:
- Crear una tabla dada una definición gráfica, sin estilos.
- Agregar estilos a la tabla, de acuerdo a una definición.

6 JavaScript

- **introducción**
- sintaxis
- variables básicas
- árbol DOM
- HTML dinámico
- debug
- librerías útiles

7. formularios

8. evolución a HTML5

introducción

motivación: contenido interactivo

Hasta el momento, se ha construido contenido web **estático**:

- Las etiquetas HTML proveen un **contenido** que se muestra en el navegador.
- El contenido es estático, es decir, no es interactivo, pues no cambia frente a acciones del usuario.
- Agregando estilos gráficos con CSS, se enriquece el aspecto, y se pueden hacer efectos interactivos leves, como por ejemplo cambiar el aspecto de un hyperlink al pasar sobre él, pero siempre en forma limitada.

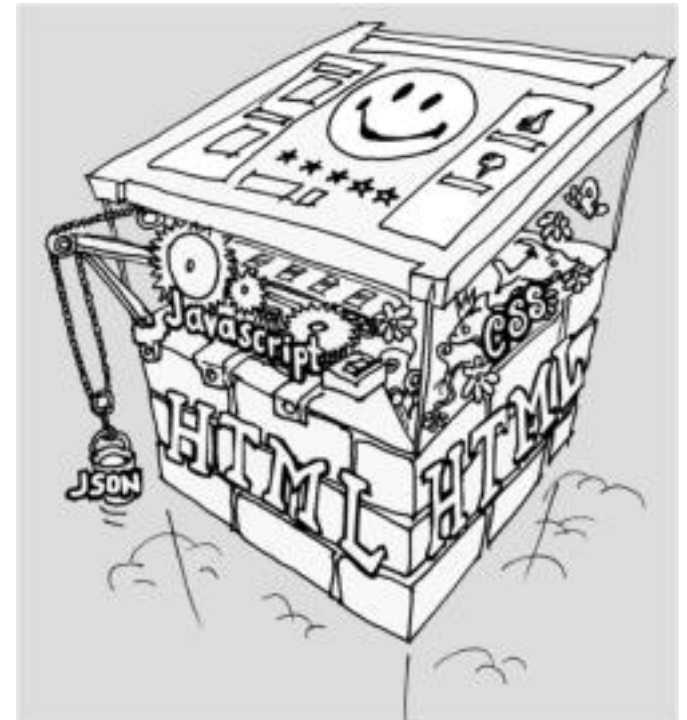
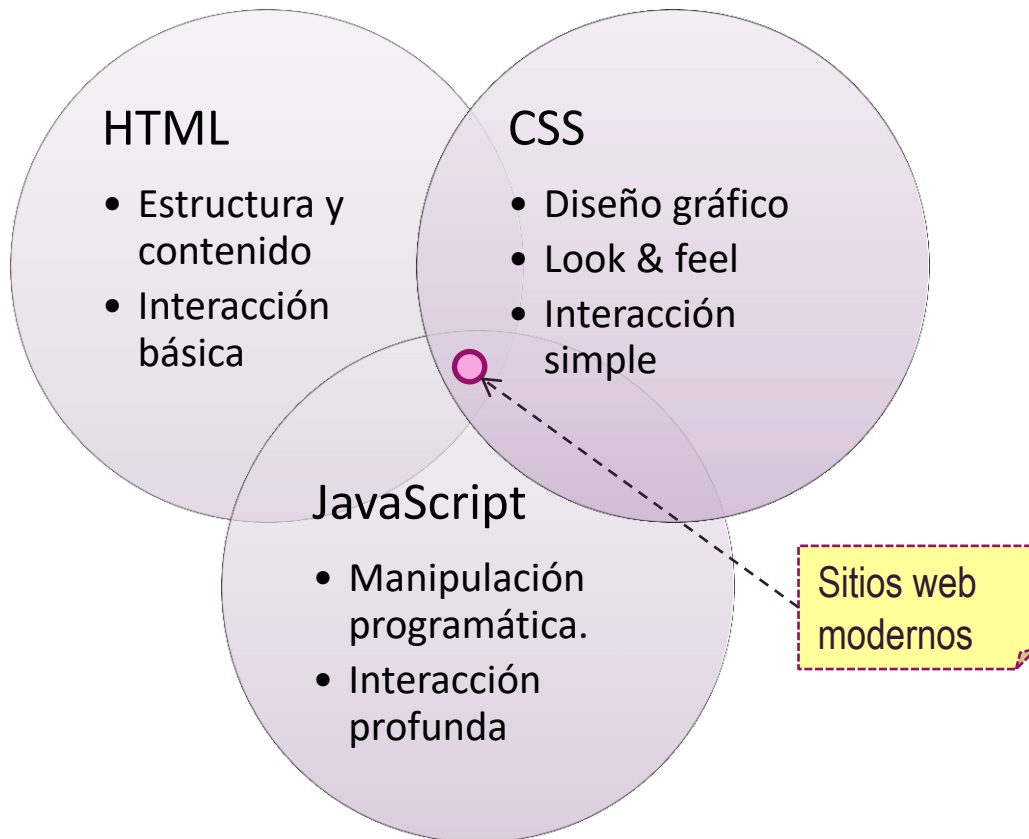
A través de **JavaScript**, se pueden definir funcionalidades que permiten tener un contenido interactivo y dinámico. Por ejemplo:

- Control de visibilidad, lo que permite por ejemplo distribuir el contenido en pestañas o cambiarlo dinámicamente.
- Manejo de eventos, lo que permite tener menús de opciones.
- Interacción con el servidor para despliegue de información (AJAX).

introducción

motivación: contenido interactivo

La característica dinámica se ilustra en la siguiente figura:



introducción

características



Algunos datos acerca del lenguaje **JavaScript**:

- Fue creado por **Netscape** Communications en 1995.
- Se llamaba originalmente LiveScript, pero Netscape cambió el nombre a **JavaScript** debido a que Netscape tenía soporte para Java (marketing).
- Es **interpretado** y **orientado a objetos**.
- Débilmente tipado (todas las variables son de tipo 'var').
- Utiliza una especificación de lenguaje llamada **ECMAScript**.
- **Sintaxis** similar a **C**, y **convenciones** de codificación similar a **Java**, aunque no están relacionados.
- Su mayor utilización es en el navegador (lado cliente), pero existe también por el lado servidor.
- Su uso se ha **incrementado** significativamente por el enfoque **interactivo** y **dinámico** de los sitios web actuales.

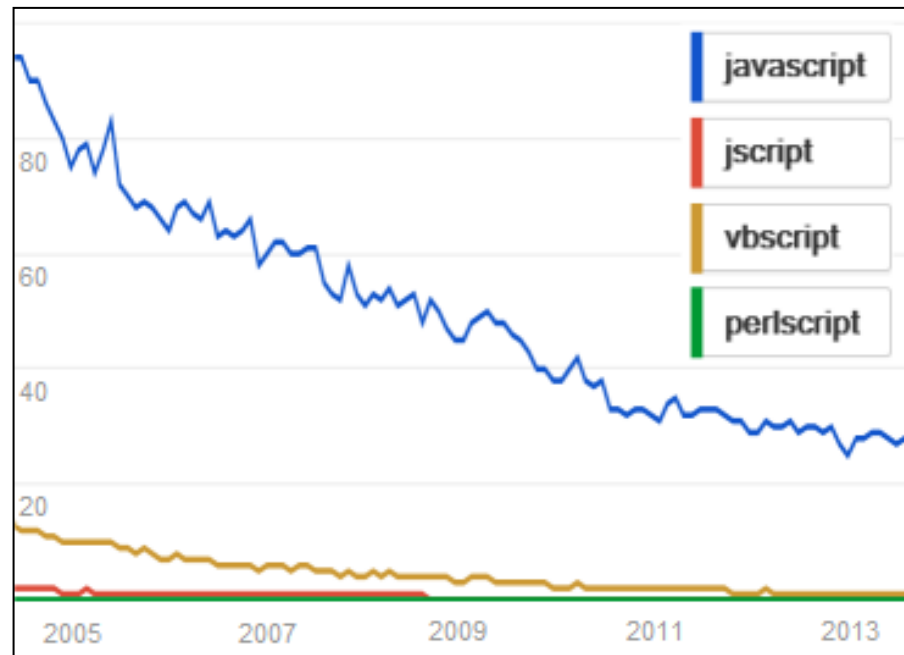


introducción

otros lenguajes de script

Originalmente se utilizaban varios lenguajes de script además de JavaScript, como VBScript o PerlScript, pero hoy en día se ha estandarizado y extendido el uso prácticamente sólo de JavaScript.

Según Google Trends:



introducción

utilización de JavaScript

El tag **<script>** permite añadir comportamiento dinámico a un sitio web por medio de lenguajes de script, y en particular JavaScript.

Para utilizar JavaScript, se puede hacer de dos formas:

- Dentro de la propia página, normalmente en el <head>:

```
<script type="text/javascript">  
    //código JavaScript  
</script>
```

type: Define el tipo MIME del script incluido. Debe ser "text/javascript".

introducción

utilización de JavaScript

- En un archivo aparte, de extensión 'js', referenciado desde la página:

```
<script type="text/javascript" src="lib.js"></script>
```

type: Define el tipo MIME del archivo de script referenciado. Debe ser "text/javascript".

src: URL que referencia al archivo que contiene el código JavaScript. Permite rutas relativas.

El contenido de lib.js es directamente el código JavaScript, sin etiquetas.

Esta opción permite reutilizar el código en otras páginas, por lo que es recomendable.

6 JavaScript

- **sintaxis**
- variables básicas
- árbol DOM
- HTML dinámico
- debug
- librerías útiles

7. formularios

8. evolución a HTML5

sintaxis

identificadores

Un **identificador** es un nombre mediante el cual se nombra un elemento del código (clase, variable, método, etc.)

- Es una **secuencia alfanumérica** cuyo primer carácter **no es un número**.
- Se pueden incluir vocales acentuadas, la ñ y letras con símbolos (ç, ...), aunque **no es recomendable** por problemas de portabilidad entre plataformas, ya que los juegos de caracteres pueden afectar el código.
- Se permiten incluir otros caracteres imprimibles (\$, _) exceptuando aquellos que tienen un significado especial dentro del lenguaje (por ejemplo % y ?). De todas formas, **no se recomienda**.

Ejemplos válidos (aunque no todos cumplen convenciones de código):

Arriba, caña, C3P0, àëîò, hola, _barco_

Ejemplos no válidos:

3com, 123, 4D5v, C#, a!b, r¨t, qwe`t

The diagram shows two groups of invalid identifiers. The first group, '3com, 123, 4D5v', has arrows pointing to the first character of each (3, 1, 4) with a label 'comienzan con número' (start with number). The second group, 'C#, a!b, r¨t, qwe`t', has arrows pointing to the characters #, !, ¨, and ` with a label 'caracteres no válidos' (invalid characters).

sintaxis

variable

Una **variable** es una referencia a un **valor** o a un **objeto** de un determinado tipo. Tiene un identificador para referirse a éste.

Para declarar una variable se utiliza la palabra **var** seguida de un identificador:

```
var <identificador>  
var x; var letra; var mensaje; var obj;
```

Asimismo se puede inicializar una variable asignándole un valor u objeto en la declaración, utilizando el símbolo "=":

```
var x = 45;  
var mensaje = "Hola";  
var enabled = true;  
var today = new Date();
```

Se observa que 'var' se utiliza para todos los tipos. Por eso se dice que JavaScript es "débilmente tipado".

sintaxis

tipos de variable

Una **variable** puede ser de varios tipos, aunque siempre se identifica como var. Los tipos utilizados son:

- **texto (string)**: se asignan con una cadena, con comillas dobles o simples:

```
var mensaje = "El campo 'nombre' es requerido";
```

```
var mensaje = 'Comillas "dobles"';
```

Las comillas simples y dobles son intercambiables, siempre que se mantenga el orden de cierre.

- **número (number)**: se asignan con un número de cualquier tipo. No se hace distinción entre enteros o números con decimales.

```
var max = 10;
```

```
var distancia = 12.34;
```

Separador decimal: punto.

sintaxis

tipos de variable

- Otros tipos utilizados:
- **lógico (boolean)**: se asignan con los valores true o false.

```
var enabled = true;  
var isValid = false;
```

- **objeto**: JavaScript maneja objetos internamente, y permite definir objetos personalizados, como se ve más adelante.

```
var values = new Array();  
var today = new Date();  
var obj = new Object();
```

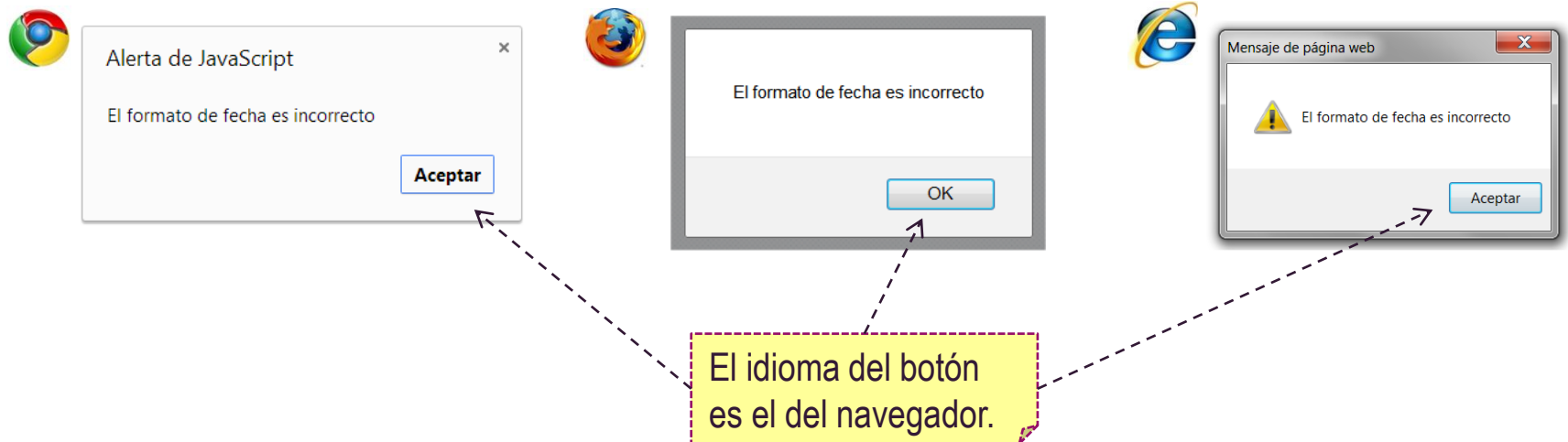
sintaxis

mensajes interactivos

JavaScript provee comandos para mostrar mensajes interactivos al usuario:

- **alert**: mensajes informativos o de advertencia.

```
alert("El formato de fecha es incorrecto");
```



sintaxis

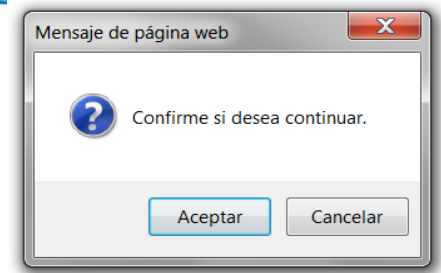
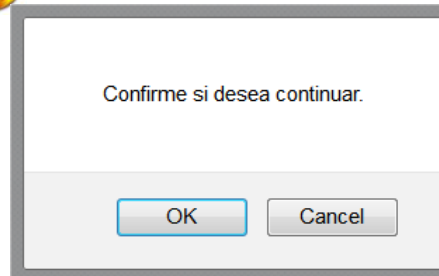
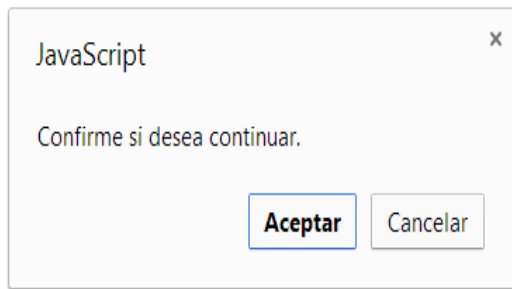
mensajes interactivos

- **confirm**: mensajes de decisión.

```
if (confirm("Confirme si desea continuar.")) {  
    //lógica si marca Aceptar/OK  
}
```

El retorno indica el botón presionado:

- Aceptar/OK: true
- Cancelar: false



- **prompt**: mensajes de solicitud de información. Poco utilizado.

sintaxis

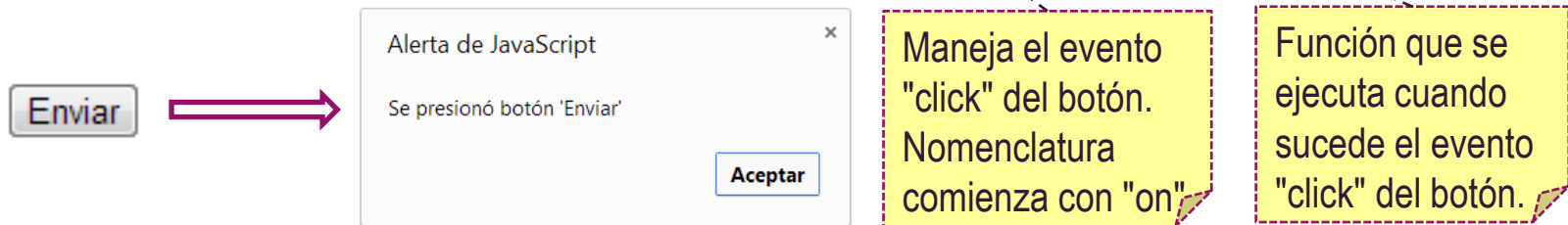
función

JavaScript define funciones, que pueden ser ejecutadas:

```
function mensaje() {  
    alert("Se presionó botón 'Enviar'");  
}
```

Para ejecutar una función, se puede asociar a un evento de la página:

```
<input type="button" value="Enviar" onclick="mensaje()" />
```



sintaxis básica

Caso práctico 6-1: funciones y eventos

- Resumen del ejercicio:
- Crear una nueva página HTML.
- Agregarle un botón llamado "Mensaje"
- Crear una función JavaScript que maneje el evento "click" del botón.
- Incluir mensajes de alert y confirm en la función.

sintaxis

ciclo de vida de una variable

Una **variable** se crea en el ámbito de un bloque que la contiene:

```
var count;
```

Las variables que están fuera de un bloque son globales dentro de la página.

```
function inc() {
```

Inicio bloque

```
  count++;
```

Variable local

```
  var message = "La cuenta es " + count;
```

```
  alert(message);
```

```
}
```

Fin bloque

Si una variable se declara sin 'var', se considera global. Esto es **no recomendable**.

- Todas las variables creadas son **destruidas al salir de la página**.
- Si se vuelve a entrar a la página, las variables se crean nuevamente, no estando relacionadas con valores anteriores de las mismas.

sintaxis

control de flujo

Sentencia condicional **if**

```
if (<condición>) {  
    <instrucciones>  
} else {  
    <instrucciones>  
}
```

Ejemplo:

```
if (a > b) {  
    alert("A gana");  
} else {  
    alert("B gana");  
}
```

```
if (<condición1>) {  
    <instrucciones>  
} else if (<condición2>) {  
    <instrucciones>  
}  
  
//varias condiciones else if..  
  
} else if (<condiciónN>) {  
    <instrucciones>  
} else {  
    <instrucciones>  
}
```


sintaxis

control de flujo

Sentencia condicional **switch-case**

```
switch (<expresión>){  
  case <literal1>:  
    <instrucciones>  
    break;  
  case <literal2>:  
    <instrucciones>  
    break;  
  //N valores...  
  case <literalN>:  
    <instrucciones>  
    break;  
  default:  
    <instrucciones>  
}
```

Cuando entra a un "case", continúa hasta el siguiente "break".

Permite cualquier tipo de variable. Normalmente se utiliza Number y String.

Ejemplo:

```
switch (estado){  
  case 0:  
    alert("Opción 0");  
    break;  
  case 1:  
    alert("Opción 1");  
  case 2:  
    alert("Opción 2");  
    break;  
  default:  
    alert("Ninguna");  
}
```

si estado es 0:
"Opción 0"
si estado es 1:
"Opción 1"
"Opción 2"

Ya que no hay break en case 1.

sintaxis

control de flujo

Sentencia iterativas **while** y **do-while**

```
while (<condición>) {  
    <instrucciones>  
}
```

Ejemplo:

```
var a = 3;  
  
while (a > 0) {  
    alert(a);  
    a--;  
}
```

```
do {  
    <instrucciones>  
while (<condición>);
```

Ejemplo:

```
var a = 0;  
  
do {  
    alert(a);  
    a--;  
} while (a > 0);
```

Pasa al menos
una vez.

sintaxis

control de flujo

Sentencia iterativa **for**

```
for (<declaración>; <condición>;  
<instrucción>) {  
    <instrucciones>  
}
```

Ejemplo:

```
for (var i = 0; i < 3; i++) {  
    alert(i);  
}
```

```
for (var <índice> in <iterable>) {  
    <instrucciones>  
}
```

Ejemplo:

```
var v = [4,7,9,1];  
  
for (var a in v){  
    alert(v[a]);  
}
```

Los array se ven
más adelante

sintaxis

break y continue

Existen dos elementos de control para los ciclos:

- **break**: detiene el ciclo, continuando con las instrucciones posteriores.
- **continue**: detiene la iteración, continuando con la siguiente.

```
for (var i = 0; i < 5; i++) {  
    if (i == 3) {  
        break;  
    }  
    alert(i);  
}
```

0
1
2

```
for (var i = 0; i < 5; i++) {  
    if (i == 3) {  
        continue;  
    }  
    alert(i);  
}
```

0
1
2
4

sintaxis básica

comentarios

Los comentarios pueden ser por línea o por bloque:

```
//Comentario  
//de  
//línea
```

```
/*  
Comentario de bloque  
*/
```

Ejemplo:

```
var a = 56; //assign. básica
```

Ejemplo:

```
/*  
Con el siguiente comando se obtienen  
las propiedades del navegador  
*/  
var nav = window.navigator;
```

6 JavaScript

- **variables básicas**
- árbol DOM
- HTML dinámico
- debug
- librerías útiles

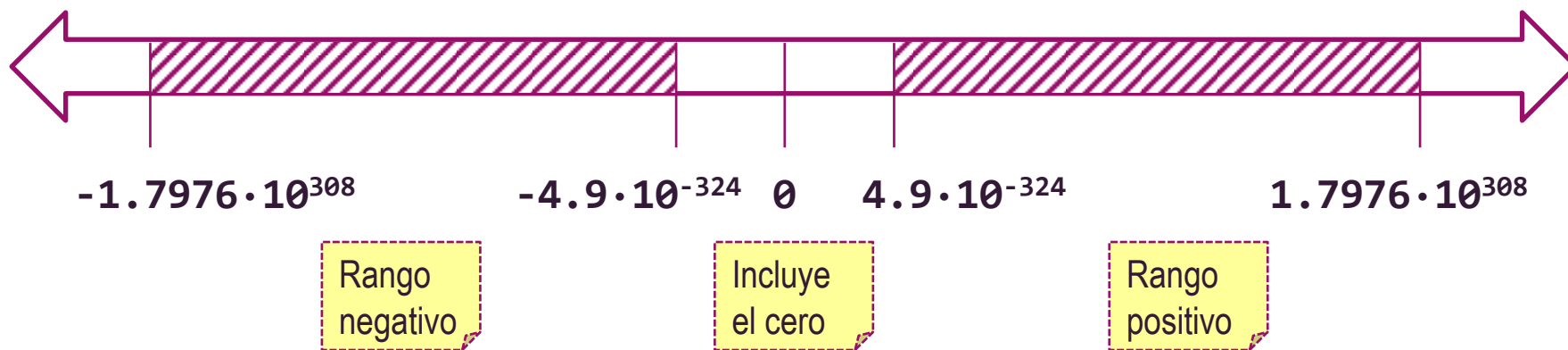
7. formularios

8. evolución a HTML5

variables básicas

número

JavaScript maneja variables de tipo **numérico (number)**. Internamente se representan como un número de punto flotante de **64 bits**, lo que permite **doble precisión** y el rango siguiente:



Utiliza un punto como separador decimal.

Posiblemente en futuras versiones se utilizarán tamaños mayores (128 bits).

variables básicas

operadores numéricos

Un **operador** es un **símbolo** formado por uno o más caracteres, que permite realizar una determinada operación entre uno o más datos y produce un resultado.

Operadores aritméticos:

- Unarios: negativo (-) y positivo (+)
- Binarios: suma (+), resta (-), producto (*), división (/) y módulo (%)

Operadores relacionales:

- igualdad (==). No confundir con =, que es para asignación.
- desigualdad (!=)
- mayor que (>)
- menor que (<)
- mayor o igual que (>=)
- menor o igual que (<=)

variables básicas

lógica

JavaScript maneja variables de tipo **lógico (boolean)**. Internamente se representan como un valor equivalente a un bit.

Ejemplos:

```
var enabled = true;
```

```
var isFinished = count > 10;
```

```
var hasValue = message != "";
```

Asignación

Comparación

Un boolean puede
contener el resultado de
un operador relacional

variables básicas

operadores lógicos

- Unarios: not (!)
- Binarios: and (&&) y or (||). Son perezosos (no evalúan el segundo operando cuando se deduce el resultado del primero). No perezosos: & y |, que son poco utilizados.

```
var a = false;  
var b = true;  
var c = true;
```

```
var j = a && (b || c);
```

Evalúa sólo a, y resulta false

```
var k = b || (b && c);
```

Evalúa sólo b, y resulta true

```
var m = !b & (b || c);
```

Evalúa !b y también el paréntesis, del cual sólo evalúa b. Resultado se convierte con Boolean(m).

```
var n = !a | (b && c);
```

Evalúa !a y también el paréntesis, del cual evalúa b y c. Resultado se convierte con Boolean(n).

variables básicas

operadores y asignación

El operador de **asignación** básico es (**=**) que, a parte de asignar, devuelve el valor asignado. La asignación es asociativa por la derecha.

Operadores **compuestos**: realizan una operación y asignan el resultado. Existen para la suma (**+=**), resta (**-=**), producto (***=**), división (**/=**), módulo (**%=**).

Operadores **incrementales**: realizan un incremento (**++**) o decremento (**--**) de una unidad en una variable. Si el operador se coloca antes del nombre de la variable devuelve el valor de la variable antes de incrementarla; si se hace después, se incrementa y devuelve el valor ya incrementado

variables básicas

operadores y asignación

Ejemplos de asignación (todos son **var** tipo **number**):

```
a = 56;      //asignación básica
```

```
b = 78 + a;  //asignación básica
```

```
c += 23;     //asignación compuesta. Equivale a c = c + 23
```

```
i++;        //Incremento. Equivale a i = i + 1
```

```
d = i++;    //Asigna e incrementa. Si i vale 3 al inicio, d vale 3.
```

```
d = ++i;    //Incrementa y asigna. Si i vale 3 al inicio, d vale 4.
```

variables básicas

operador condicional

Es el único de JavaScript con tres operandos, cuya sintaxis es:

`<condición> ? <expresión1> : <expresión2>`

- Se evalúa <condición>.
- Si es verdadera se devuelve el resultado de evaluar <expresión1>
- Si es falsa, el resultado de evaluar <expresión2>
- Ambas expresiones deben ser del tipo al que se asigna el resultado.

Ejemplo:

`max = (a > b) ? a : b;`

Si a es mayor que b, devuelve a.
En caso contrario, devuelve b.

variables básicas

texto, métodos

JavaScript maneja variables de tipo **texto (string)**. Se declaran como textos entre comillas simples o dobles. Internamente se almacenan como una cadena de caracteres, sin un largo máximo establecido. Ejemplo:

```
var mensaje = "Hola";
```

Las cadenas son objetos que tienen asociados métodos útiles para su manejo. A continuación se describen algunos de los más utilizados.

- Para conocer el largo de un string, es decir, el número de caracteres, se utiliza el método `length`:

```
str = "12345";
```

```
str.length
```

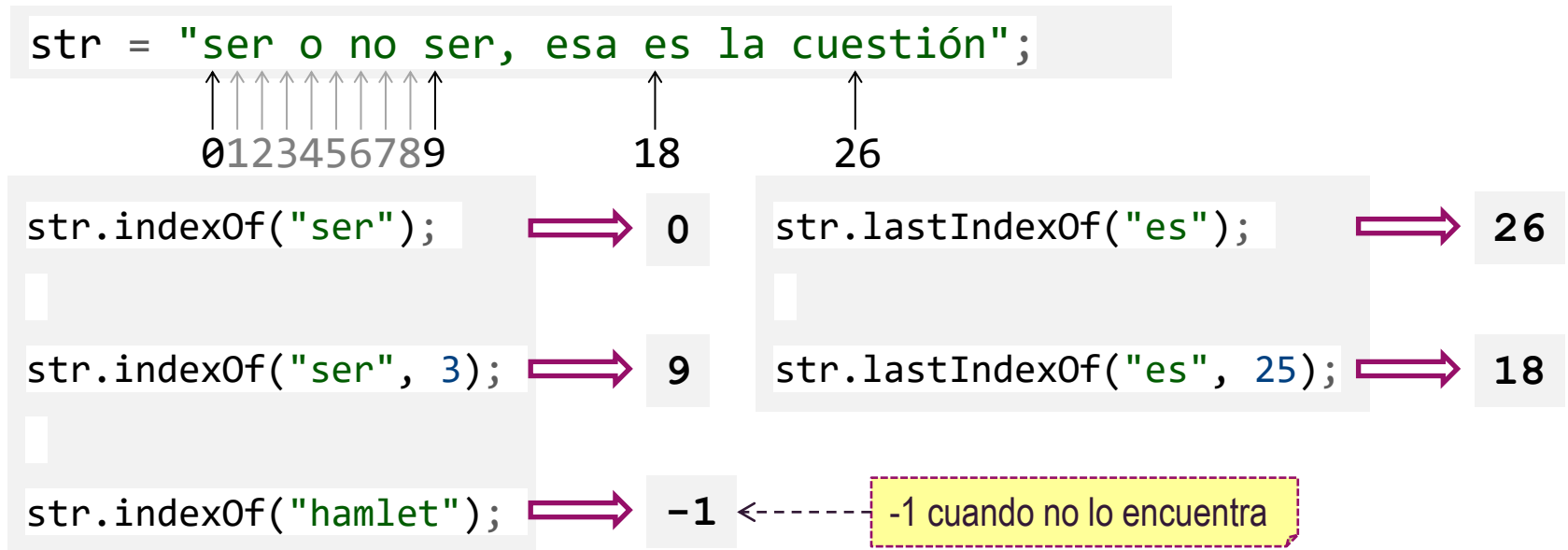


5

variables básicas

texto, métodos

- Para buscar la primera posición de un string dentro de otro, se utiliza **indexOf**, y la última, con **lastIndexOf**:



variables básicas

texto, métodos

Continuando:

- indexOf se utiliza también para saber si un string contiene una cadena:

```
str = "Cinema Paradiso";
```

```
str.indexOf("a P") != -1  $\Rightarrow$  true
```

- Para extraer un caracter en una posición, se utiliza charAt:

```
str = "Cinema Paradiso";
```

```
str.charAt(2)  $\Rightarrow$  'n'
```

0 a n-1

- Para extraer un trozo de una cadena, se utiliza substring:

Desde 2, largo 4

```
str = "casablanca";
```

0 1 2 3 4 5 6 7 8 9

Ubicar posición 2

```
str.substring(2, 6)
```

Restar: $6 - 2 = 4$

```
 $\Rightarrow$  "sabl"
```


variables básicas

texto, métodos

Continuando:

- Para pasar el contenido todo a minúsculas, se utiliza `toLowerCase()`, y a mayúsculas es con `toUpperCase()`:

```
str = "Artificial Intelligence";
```

```
str.toLowerCase() ➡ "artificial intelligence"
```

```
str.toUpperCase() ➡ "ARTIFICIAL INTELLIGENCE"
```

- Para comparar dos textos, se utiliza `"=="`:

```
var str1 = "euro";  
var str2 = "\u0065\u0075\u0072\u006f";
```

```
str1 == str2 ➡ true
```

"euro" en unicode

variables básicas

texto, métodos

Continuando:

- Para comparar dos textos, ignorando mayúsculas y minúsculas (*case-insensitive*), se deben pasar a mayúsculas o minúsculas y utilizar "==":

```
str1 = "lower or UPPER";  
str2 = "Lower or Upper";
```

```
str1.toLowerCase() == str2.toLowerCase() ➡ true
```

- Si se compara un texto y un número con el mismo valor, sucede lo siguiente:

```
str1 = "12";  
str2 = 12;
```

```
str1 == str2
```

```
➡ true
```

Compara valores

```
str1 === str2
```

```
➡ false
```

Compara valores y tipos

```
str1 === String(str2)
```

```
➡ true
```

```
Number(str1) === str2
```

```
➡ true
```

variables básicas

texto, métodos

Continuando:

- Para reemplazar en un String las ocurrencias de una cadena de caracteres por otra, se utiliza el método `replace`:

Por default, sólo reemplaza la primera ocurrencia.

```
name = "les aventures de tintin et milou";
```

```
name = name.replace("tin","tan");
```

⇒ "les aventures de tantin et milou"

- El método `replace` admite expresiones regulares (se ven más adelante), lo que flexibiliza su uso:

```
name = "Déjà vu";
```

Encuentra las minúsculas de la "a" a la "z"

```
name.replace(/[a-z]/, "_");
```

⇒ "Dé_à vu"

Reemplaza la primera por un "_"

```
name.replace(/[a-z]/g, "_");
```

⇒ "Dé_à _"

Reemplaza todas las minúsculas por un "_"

```
name.replace(/[a-z]/gi, "_");
```

⇒ "_é_à vu"

Reemplaza todas, ignorando mayúsculas y minúsculas

variables básicas

concatenación

Los textos se pueden **concatenar** utilizando el operador "+":

```
var str = "Esto es " + "un string";
```

Como el operador "+" también se utiliza para sumar números, hay ciertos matices a tener en cuenta, los que se ilustran a continuación:

```
s1 = 1 + 3 + "5" + "7";      457
```

Regla de izquierda a derecha.
Suma 1 + 3, y luego concatena "5" y "7".

```
s2 = 1 + (3 + "5") + 7;      1357
```

Calcula el paréntesis primero.
Concatena 1, el resultado, y 7.

```
s3 = "1" + (3 + 5) + 7;      187
```

Como el primer operando es un string,
comienza concatenando.

```
s4 = 1 + "3" + 5 + 7;        1357
```

Como la primera operación incluye a un
string, comienza concatenando.

```
s5 = "" + 1 + 3 + 5 + 7;     1357
```

Si se quieren concatenar sólo números,
se puede agregar un string vacío.

```
s6 = "" + (1 + 3 + 5 + 7);    16  
s6 = String(1 + 3 + 5 + 7);
```

Si se quiere convertir una suma a string,
similar a la anterior y también String(...).

variables básicas

caracteres especiales

En JavaScript existe una representación específica para algunos caracteres especiales, que permite definirlos y utilizarlos:

Caracter	Significado	Utilización
<code>\n</code>	newline (Nueva línea)	Para saltos de línea en manejo de cadenas
<code>\r</code>	carriage return (Salto de carro)	Idem anterior, en Windows.
<code>\t</code>	tab (tabulación)	Detectar o insertar tabulaciones
<code>\"</code>	comillas (también existe <code>\'</code>)	Colocar comillas en un String. Ej: <code>str = "Esto va \"entre comillas\".";</code>
<code>\\</code>	escapa un <code>\</code>	Para escribir un <code>"\"</code> en una cadena. Ej: <code>str = "Para newline se utiliza \\n";</code>
<code>\b</code>	backspace	Poco usado
<code>\f</code>	form feed	Poco usado

variables básicas

conversiones

Si se tiene un valor tipo texto, por ejemplo "1", y se quiere manejar como número, JavaScript provee funciones para **convertir** entre tipos de variables. Esto es útil porque al ser todas 'var', no es posible diferenciarlas por su tipo.

Ejemplos:

```
var strOne = "1";  
var numOne = Number(strOne);  
  
var strEnabled = "true";  
var enabled = Boolean(strEnabled);
```

También es posible obtener el tipo interno de una variable, utilizando typeof. Ejemplo:

```
if (typeof numOne == "number") {  
    alert("numOne es un number");  
}
```

variables básicas

arrays

JavaScript soporta arrays, que contienen un conjunto de variables:

- Para la declaración del array, se utilizan varias nomenclaturas:

```
var nums = new Array();  
nums[0] = 1;  
nums[1] = 2;  
nums[2] = 3;
```

```
var nums = [1, 2, 3];
```

- Para iterar y acceder los valores del array:

```
for (var i=0; i < nums.length; i++) {  
    //se accede con nums[i]  
};
```

sintaxis básica

Caso práctico 6-2: variables básicas

- Resumen del ejercicio:
- Utilizar la página HTML.
- Agregarle un botón llamado "Operación"
- Crear una función JavaScript que maneje el evento "click" del botón.
- En la función, realizar las siguientes operaciones:
 - Crear variable tipo texto con un string.
 - Realizar las operaciones sobre textos descritas en el capítulo.

6 JavaScript

- **árbol DOM**
- HTML dinámico
- debug
- librerías útiles

7. formularios

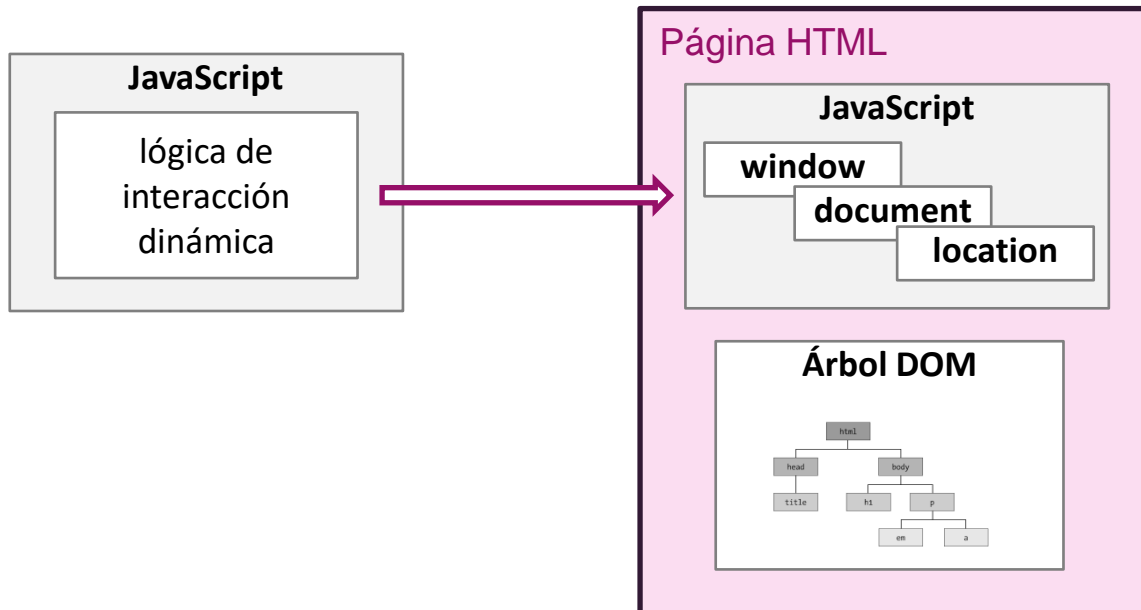
8. evolución a HTML5

árbol DOM

descripción

Hasta el momento, se ha descrito el lenguaje JavaScript, con su sintaxis básica y sus variables, con un enfoque de lenguaje de programación.

En este punto se describen los **elementos** JavaScript que asociados a la página HTML, que son los utilizados para lograr las características **dinámicas interactivas**.



árbol DOM

objetos predefinidos

Una página HTML tiene objetos JavaScript predefinidos, que permiten interactuar entre el contenido y el lenguaje:

- **window**: referencia a la ventana propiamente tal. Todos los elementos pertenecen implícitamente al objeto window. Si se abre una nueva ventana o pestaña, corresponde a otro window.
- **document**: es la raíz del documento HTML, y el "dueño" de todos los elementos. Provee atributos y métodos para acceder a los elementos de la página, lo que permite modificarlos dinámicamente, y de este modo realizar los efectos interactivos.

Por ejemplo, el método `alert(...)` pertenece al objeto window.

Por ejemplo, `document.write()` imprime texto

árbol DOM

objetos predefinidos

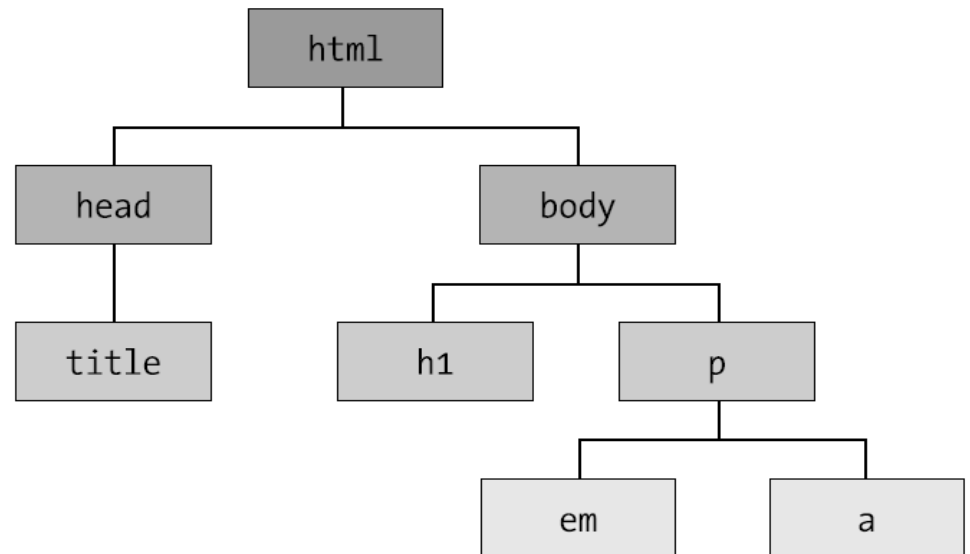
- **navigator**: contiene información sobre el navegador. Permite particularizar la programación, lo que es útil cuando existen diferencias entre los navegadores. También permite restringir una aplicación a un tipo o versión de navegador.
- **location**: contiene información sobre la URL de la página. Si se modifica, se abre una nueva URL, lo que permite utilizarlo para navegar programáticamente.
- **screen**: contiene información de la pantalla, incluyendo alto y ancho. Permite ajustar una página en función de la resolución de pantalla.
- **history**: contiene el historial de URLs visitadas, lo que permite volver atrás programáticamente.

estructura

el árbol DOM del documento

Es una ayuda el ver la estructura de un documento XHTML en forma de un árbol. El árbol comienza con el elemento raíz en la parte superior y todos los demás elementos parten hacia abajo desde esta raíz.

```
<html>
  <head>
    <title>DOM</title>
  </head>
  <body>
    <h1>DOM</h1>
    <p>
      Ejemplo <em>árbol DOM</em>
      <a href="#">Volver</a>
    </p>
  </body>
</html>
```



árbol DOM

Caso práctico 6-3: propiedades del navegador y pantalla

Resumen del ejercicio:

- Crear una nueva página HTML.
- Crear una función JavaScript llamada onload, que se ejecuta cuando finaliza la carga de la página.
- Obtener el nombre del navegador utilizado, y desplegarlo en un alert.
- Obtener la resolución de pantalla del equipo, y desplegarlo en un alert.

6 JavaScript

- **HTML dinámico**
- debug
- librerías útiles

7. formularios

8. evolución a HTML5

HTML dinámico

definición

HTML dinámico o **DHTML** es un conjunto de tecnologías para crear sitios web interactivos. Utiliza JavaScript para modificar el contenido **HTML** y los estilos **CSS**, a través de la utilización del **árbol DOM**.

Con DHTML se puede, por ejemplo:

- Controlar la **visibilidad** de una sección de la página, lo que permite hacer efectos como pestañas, menús o acordeón.
- Cambio dinámico de **estilo gráfico**, lo que permite por ejemplo mostrar selecciones, o efectos de habilitación y deshabilitación.
- Crear **controles personalizados** que no existen nativamente en HTML, como barras de progreso.
- **Actualizar** secciones de pantalla, como por ejemplo una de noticias, sin tener que recargar la página.

En este punto se ilustra el DHTML con algunos ejemplos.

HTML dinámico

cambio de clase de estilo

Por ejemplo, se puede cambiar dinámicamente la clase de estilo de una sección de pantalla. Por ejemplo, un párrafo tiene un estilo normal y uno destacado, y se quiere que al pasar el puntero sobre él, cambie.

- Resultado y estilos CSS:

Párrafo que cambia
al pasar el mouse.

**Párrafo que cambia
al pasar el mouse**

```
.sec {  
    font-family: Courier;  
    border: 1px solid blue;  
    padding: 3px;  
    cursor: default;  
    width: 200px;  
}  
  
.pnormal {  
    font-weight: normal;  
}  
  
.phigh {  
    font-weight: bold;  
}
```

HTML dinámico

cambio de clase de estilo

- Código HTML y JavaScript:

```
<div class="sec"
  onmouseover="change(true)"
  onmouseout="change(false)">
  <p id="par" class="pnormal">
    Párrafo que cambia al pasar el mouse.
  </p>
</div>
```

Cuando se pasa el mouse sobre el área del div, se invoca la función change(true).

Cuando se saca el mouse del área del div, se invoca la función change(false).

```
<script type="text/javascript">
  function change(opt) {
    var p = document.getElementById("par");
    p.className = (opt) ? "phigh" : "pnormal";
  }
</script>
```

Obtiene el objeto asociado al párrafo, con el id="par".

Cambia dinámicamente la clase de estilo del párrafo, según el parámetro.

HTML dinámico

cambio de contenido

Para cambiar el contenido HTML de una parte de una pantalla en forma dinámica, se utiliza la propiedad **innerHTML**. Aunque no es parte del estándar, todos los navegadores conocidos manejan la propiedad. Por ejemplo, se puede modificar el contenido de un elemento.

- Resultado:

Dynamic

Valor aleatorio: 4



Cuando se presiona el botón,
cambia dinámicamente el valor

Cambiar

HTML dinámico

cambio de contenido

- Código HTML y JavaScript:

```
<h1>Dynamic</h1>
<p id="par">
  Valor aleatorio: 0
</p>

<input type="button" value="Cambiar"
onclick="change()" />
```

Al presionar el botón, se invoca la función change().

```
<script type="text/javascript">
  function change() {
    var p = document.getElementById("par");
    var num = 1 + Math.floor((9*Math.random()));
    p.innerHTML = "Valor aleatorio: " + num;
  }
</script>
```

Obtiene el objeto asociado al párrafo, con el id="par".

Genera un número aleatorio entre 1 y 9

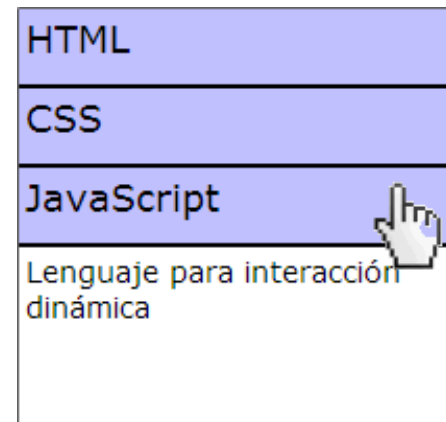
Cambia el contenido del objeto p, que corresponde al párrafo.

HTML dinámico

control de visibilidad

Para controlar la **visibilidad** de una parte de una página, se controla el atributo de estilo llamado "**display**" asociado a la rama del árbol DOM correspondiente. Por ejemplo, una página tipo "acordeón":

- Resultado:



HTML dinámico

control de visibilidad

- Estilos:

```
.header {  
  border: 1px solid black;  
  cursor: pointer;  
  width: 200px;  
  height: 30px;  
  padding: 3px;  
  background-color: #C0C0FF;  
  font-size: 1.1em;  
}
```

```
.content {  
  border: 1px solid black;  
  font-size: 0.9em;  
  width: 200px;  
  height: 80px;  
  padding: 3px;  
}
```



HTML dinámico

control de visibilidad

- Elementos HTML:

Al hacer click sobre el div, se ejecuta el método JavaScript "visib".

```
<div class="header" onclick="visib('ht')">
  HTML
</div>
<div class="content" id="ht">
  Lenguaje de contenido web
</div>
<div class="header" onclick="visib('cs')">
  CSS
</div>
<div class="content" id="cs" style="display: none">
  Estilos gráficos
</div>
<div class="header" onclick="visib('js')">
  JavaScript
</div>
<div class="content" id="js" style="display: none">
  Lenguaje para interacción dinámica
</div>
```

HTML
Lenguaje de contenido web
CSS
JavaScript

Inicialmente oculta el contenido del div y el espacio que utiliza

HTML dinámico

control de visibilidad

■ Función JavaScript:

```
function visib(divId) {  
    var ar = ["ht", "cs", "js"];  
    for (var i=0; i < ar.length; i++) {  
        var divElem = document.getElementById(ar[i]);  
        divElem.style.display =  
            (divId == ar[i]) ? "block":"none";  
    };  
}
```

array con ids de los contenidos

Accede al objeto del árbol DOM con el contenido, dado el id.

Cambia el atributo de estilos display, en función del parámetro:

- block: mostrar contenido
- none: ocultar contenido

divId = "ht"



HTML
Lenguaje de contenido web
CSS
JavaScript

divId = "cs"



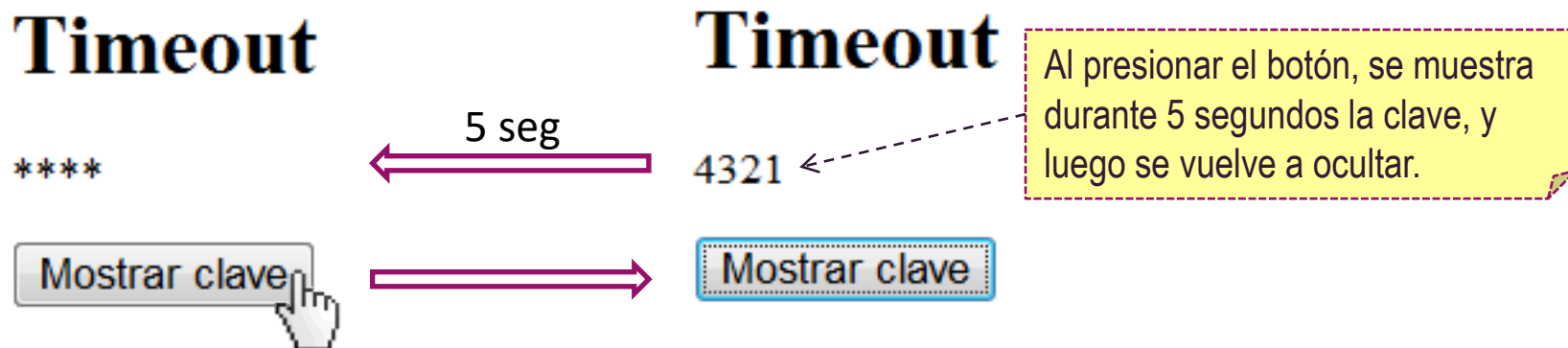
HTML
CSS
Estilos gráficos
JavaScript

HTML dinámico

timeout

A través del método `setTimeout` del objeto implícito `window`, se pueden ejecutar funciones un tiempo después. Esto permite controlar efectos en el tiempo. Por ejemplo, una página que muestra una clave durante 5 segundos, y luego la oculta.

- Resultado:



HTML dinámico

cambio de contenido

- Código HTML y JavaScript:

```
<h1>Timeout</h1>
<p id="pwd">****</p>
<input type="button" value="Mostrar clave" onclick="showPwd()" />
```

Al presionar el botón, se invoca la función showPwd().

```
<script type="text/javascript">
  function showPwd() {
    var pwd = "4321";
    document.getElementById("pwd").innerHTML = pwd;
    setTimeout(hidePwd, 5000);
  }
```

Cambia el contenido del objeto con id="pwd", que corresponde al párrafo que muestra la clave.

5000 milisegundos después, invoca la función hidePwd. Se coloca la referencia, sin paréntesis.

```
  function hidePwd() {
    document.getElementById("pwd").innerHTML = "****";
  }
</script>
```

Vuelve a colocar los **** en el contenido del párrafo con la clave.

HTML dinámico

Caso práctico 6-4: agregar filas a tabla dinámicamente

- Resumen del ejercicio:
- Crear una página HTML.
- Agregar una tabla de 3 filas y 2 columnas.
- Crear una función JavaScript que agregue una nueva fila, utilizando la propiedad innerHTML.
- Agregar un botón que invoque la función.

6 JavaScript

- **debug**
- librerías útiles

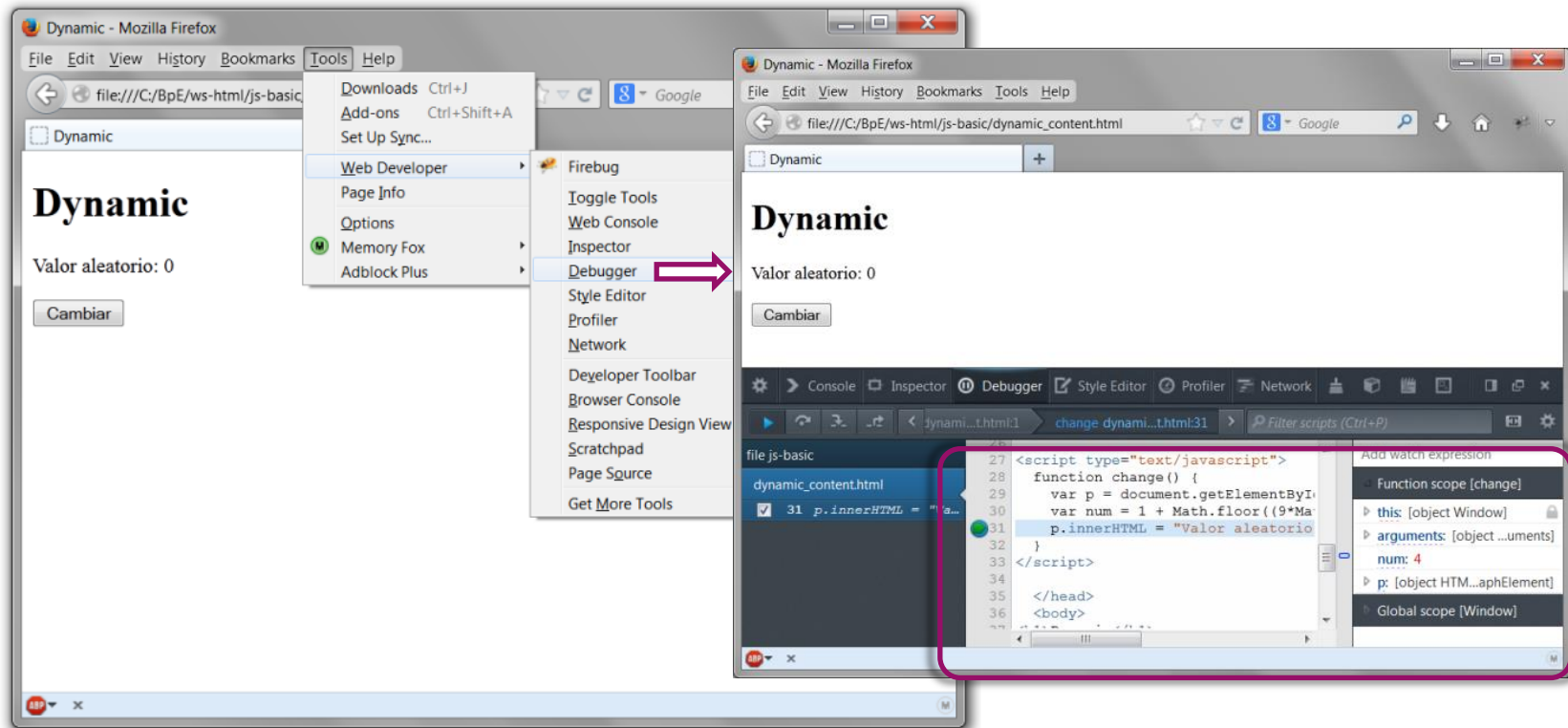
7. formularios

8. evolución a HTML5

debug

debug con Firefox

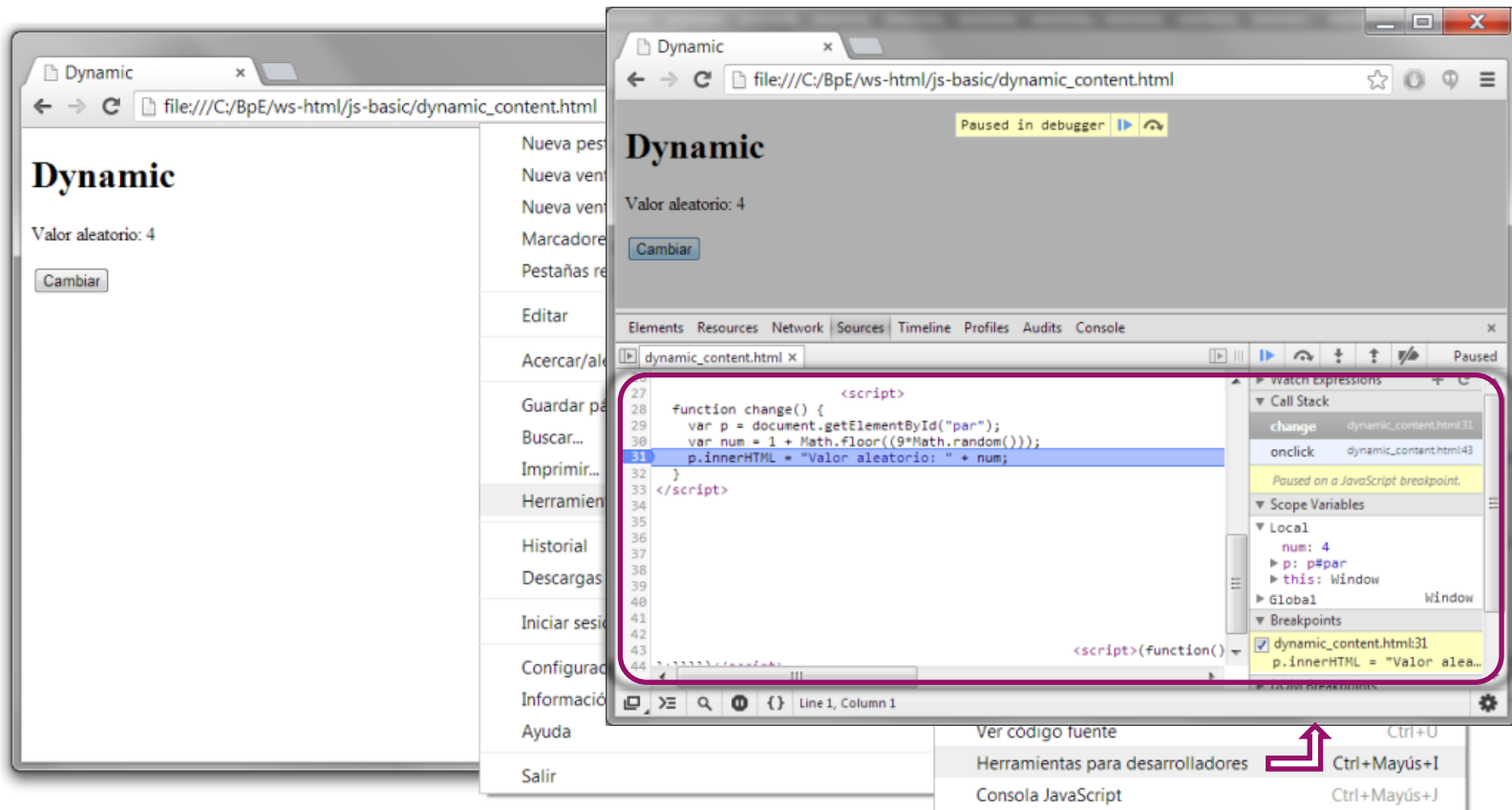
En **Firefox**, se puede hacer *debug* de JavaScript en forma nativa, utilizando el menú Tools → Web Developer → Debugger:



inspección de elementos

inspección con Chrome

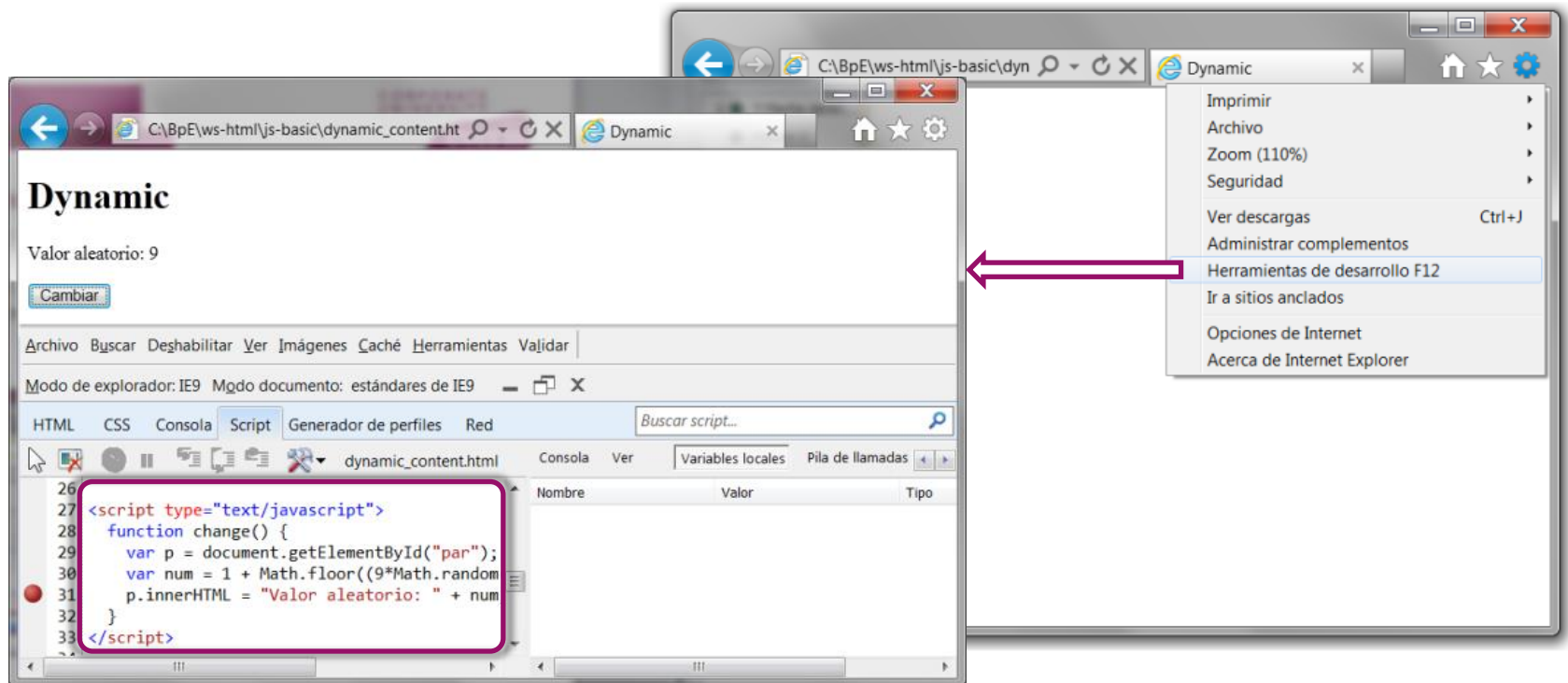
En **Chrome**, se pueden hacer debug de JavaScript:



inspección de elementos

inspección con Internet Explorer 9

Internet Explorer 9 también tiene herramientas para hacer debug, aunque no siempre funciona correctamente:



6 JavaScript

- librerías útiles

7. formularios

8. evolución a HTML5

librerías conocidas

descripción

Aunque con HTML dinámico se pueden hacer muchos efectos interesantes, el esfuerzo asociado puede ser grande en algunos casos. En la práctica, es conveniente utilizar librerías y frameworks que ya los tienen implementados.

- Librerías:
 - jQuery
- Frameworks de componentes:
 - dhtmlxSuite
 - Dojo toolkit
 - jQuery UI
- Librerías para *single-page application*:
 - backbone.js

A continuación se utilizan brevemente algunos, a través de casos prácticos.

librerías conocidas

Caso práctico: uso de jQuery

Descripción del caso práctico:

- Importar proyecto con jQuery. El proyecto incluye una página HTML con una hoja de estilos.
- Se implementa en forma guiada la distribución de la información de la página en pestañas, aprovechando las funcionalidades ofrecidas por jQuery.

librerías conocidas

Caso práctico: uso de dhtmlxSuite

Descripción del caso práctico:

- Importar proyecto con dhtmlxSuite. El proyecto incluye una página HTML incompleta, una hoja de estilos y una librería JS con datos fijos.
- Se agrega en forma guiada un objeto de tipo grid, construido a partir de los datos de la librería.
- Se implementa la selección de una fila y extracción de su información.

7 formularios

- **estructura**
- validaciones
- gestión de errores

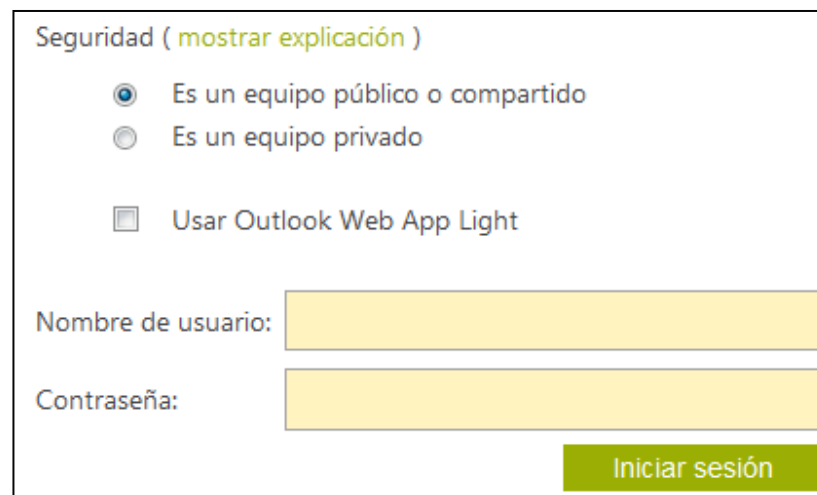
8. evolución a HTML5

formularios

descripción

Hasta el momento, se ha mencionado el concepto de sitio web interactivo, pero con una orientación hacia los efectos gráficos y lectura de información. Eso es una orientación hacia "páginas".

En un sitio interactivo real, el usuario debe poder intercambiar información con el servidor, para lo cual resultan muy útiles los **formularios (form)**.



Seguridad ([mostrar explicación](#))

☒ Es un equipo público o compartido

☐ Es un equipo privado

☐ Usar Outlook Web App Light

Nombre de usuario:

Contraseña:

← Ejemplo de web mail corporativo

formularios

descripción

Formulario:

- Un **conjunto de campos HTML** se colocan dentro de una etiqueta **<form>**.
- Cada campo tiene asociado un **nombre** (name), con el que es enviado y recuperado en el servidor, y un valor. No confundir con el "id".
- El formulario es enviado al servidor a través de la ejecución de un **"submit"**, pulsando un campo de ese tipo, o bien por JavaScript.
- El atributo **action** define la URL del objeto que procesa el formulario, por el lado servidor.

El que recibe el formulario es el lado servidor, que está implementado en alguna tecnología, como Java (JSP, Servlet), .NET, y otras como PHP. Los detalles del lado servidor están fuera del alcance de este curso, pues se conocen en los cursos respectivos posteriores (Programa web applications).

formularios

campos

Un formulario permite definir campos, que son interpretados y desplegados por el navegador. Ejemplo:

```
<form action="..." >
  <input type="hidden" name="issueId">
  <p>Summary: <input type="text" size="30"
    name="summary" value="this is a summary">
  </p>
  <p>Secret: <input type="password"
    name="secret" size="15" value="mypassword">
  </p>
  <p>Type: <select name="typeId">
    <option value="1" selected="selected">
      Technology</option>
    <option value="2">Production</option>
  </select>
  </p>
  <p><input type="checkbox" name="hasAttachments" />
  Has attachments</p>
  <p><input type="file" name="attachment"/>
  </p>
  ...
</form>
```

The diagram illustrates the mapping between HTML form code and its visual representation in a browser. It includes the following elements:

- Issue ID:** A hidden input field, represented by a box containing the text `issueId`. A yellow callout box labeled "Campo oculto" points to it.
- Summary:** A text input field with the value "this is a summary".
- Secret:** A password input field, represented by a box with 15 dots.
- Type:** A dropdown menu with "Technology" selected.
- Has attachments:** A checkbox followed by the text "Has attachments".
- Attachment:** A file upload control consisting of a text box and a "Browse..." button. A yellow callout box labeled "Campo para upload" points to it.

formularios

campos

Continuación del formulario:

El grupo de los radio button lo define el name común a todos.

```
...  
<p>State:  
<input type="radio" name="state"  
  value="P">Pending  
<input type="radio" name="state"  
  value="S" checked="checked">Solved  
<input type="radio" name="state"  
  value="C">Closed
```

State: ☐ Pending ☒ Solved ☐ Closed

Description:

This is a description

```
</p>  
<p>Description:<br/>  
<textarea rows="3" cols="40" name="desc">  
This is a description  
</textarea></p>
```

```
<p><input type="button" value="Clean" />  
</p>  
<p><input type="submit" value="Send" />  
</p>  
</form>
```

Clean

Send

Envía el formulario al destino definido en "action" utilizando el "method".

7 formularios

- **validaciones**
- gestión de errores

8. evolución a HTML5

formularios

validaciones nativas

HTML provee ciertas validaciones o comportamientos de los formularios en forma nativa:

- **Habilitación:** se puede deshabilitar un campo de cualquier tipo con el atributo disabled. Tiene normalmente texto gris.

```
<input type="text" value="45"  
      name="identifier" disabled="disabled">
```

- **Sólo lectura:** se puede dejar un campo de texto como sólo lectura con el atributo readonly. Tiene el mismo estilo, pero no permite escribir.

```
<input type="text" value="resumen"  
      name="summary" readonly="readonly">
```

- **Largo máximo:** los campos de texto tienen un atributo maxlength, que siempre es recomendable colocar por seguridad. Los textarea no lo tienen, por lo que debe resolverse por JavaScript.

formularios

manejo con JavaScript

Cada campo de un formulario se puede manejar como un objeto JavaScript, permitiendo aplicar HTML dinámico. Por ejemplo, el siguiente formulario:

```
<form class="frm" >
  <div>
    <label for="summary">Summary:</label><input type="text" size="30"
      id="idSumm" name="summary" value="this is a summary">
  </div>
  <div>
    <label for="secret">Secret:</label><input type="password"
      name="secret" size="15" value="mypassword">
  </div>
  <div>
    <label for="type">Type:</label><select id="idType" name="type">
      <option value="1" selected="selected">
        Technology</option>
      <option value="2">Production</option>
    </select>
  </div>
</form>
```

formularios

manejo con JavaScript

- Asignación de un valor a un campo de texto:

```
document.getElementById("idSumm").value = "other value";
```

- Selección de la primera opción de un campo tipo select:

```
document.getElementById("idType").selectedIndex = 0;
```

- Texto de la segunda opción de un campo tipo select:

```
document.getElementById("idType").options[  
    document.getElementById("idType").selectedIndex].text;
```

formularios

manejo con JavaScript

- Agregar una nueva opción a un campo de tipo select:

```
var opt = new Option("Methodology", "3");  
document.getElementById("idType").options[2] = opt;
```

options es un array con las opciones. Al agregarle un valor, se agrega la opción al select.

Se coloca el largo del array options.

- Eliminar una opción de un campo de tipo select:

```
document.getElementById("idType").options[2] = null;
```

Al colocar null, se elimina la opción del array.

formularios

manejo con JavaScript

- Dejar un campo como sólo lectura:

```
document.getElementById("idSumm").readOnly = true;
```

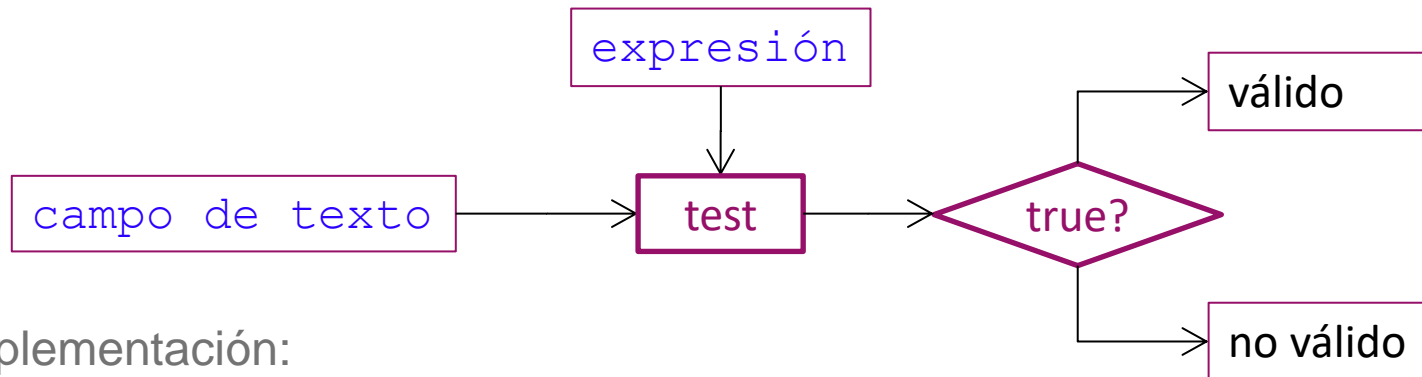
- Deshabilitar un campo:

```
document.getElementById("idType").disabled = true;
```

formularios

expresiones regulares

JavaScript permite utilizar expresiones regulares para la validación de textos, lo que resulta útil en el uso de formularios.



Implementación:

```
if (expr.test(field)) {  
    //válido  
} else {  
    //no válido  
}
```

formularios

expresiones regulares

Una expresión, aunque es una cadena, tiene una forma de codificarse que no necesita comillas.

- Comienzo: `/^`
- Final: `$/`

Para especificar tipos de caracteres:

- `\d`: dígito
- `\s`: espacio en blanco, tabulación o salto de línea
- `\w`: letra, número o `"_"`

Para especificar una cantidad de caracteres, se utiliza un número entre `{ y }`.

Por ejemplo, para validar un código postal de exactamente cinco cifras, se usa:

```
var exp = /^\\d{5}$/;
```


formularios

expresiones regulares

Si se quieren especificar varios grupos de caracteres, se colocan entre [y]:

- **[abc]**: busca las ocurrencias de "a", "b" o "c".
- **[a-e]**: busca las ocurrencias desde "a" a "e", es decir, "a", "b", "c", "d" o "e".
- **[a-eA-E]**: busca desde "a" a "e", o desde "A" a "E". **No** busca "eA".

Por ejemplo, para validar un nombre que tiene sólo letras, sin espacios, con largo mínimo 2 y máximo 20:

```
var exp = /^[a-zA-Z]{2,20}$/;
```

Si se quiere incluir la 'ñ' (que no está en el rango a-z), es:

```
var exp = /^[a-zA-ZñÑ]{2,20}$/;
```

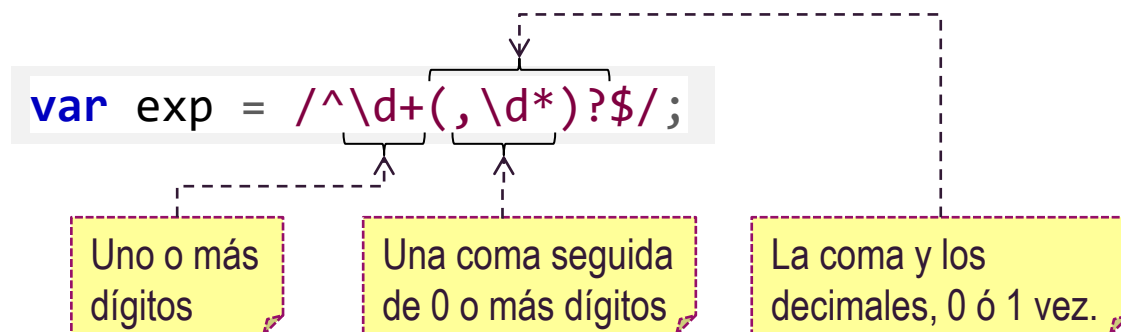
formularios

expresiones regulares

Si se quieren encontrar varias ocurrencias de una expresión, se pueden utilizar los cuantificadores:

- **+**: uno o más.
- *****: cero o más.
- **?**: cero o una.

Por ejemplo, para validar un campo con un número decimal positivo separado por coma, sin separador de miles, que admite '16,' y no admite ',15', es:



formularios

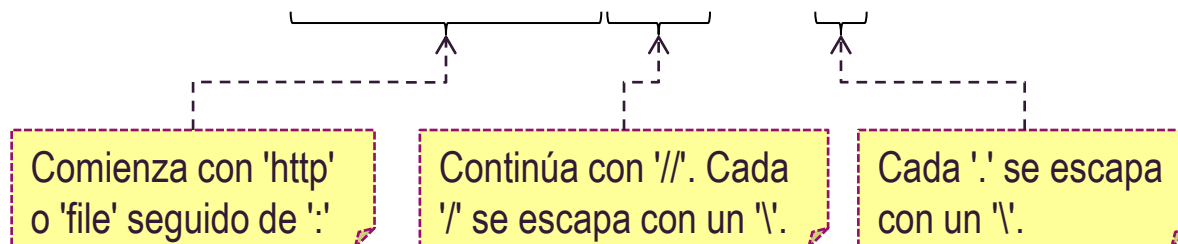
expresiones regulares

Algunas otros comportamientos:

- La expresión "." (punto) equivale a cualquier caracter.
- Si se quiere buscar caracteres utilizados en las expresiones, como '\ ' o '\.', hay que escaparlos con un '\\ '.
- Distintos valores de texto se pueden separar por '|'.

Por ejemplo, para validar una URL de un recurso, que puede tener como protocolo "http" o "file", seguida de una dirección separada por '_' y '.', es:

```
var exp = /^(http|file):\\\/\\\/\w+(\.\\w+)*$/;
```



formularios

Caso práctico: validación de formularios

Dado un formulario con campos de distinto tipo:

- Completar los largos máximos
- Realizar validaciones cruzadas
- En un campo de tipo texto, validar valor con expresión regular

7 formularios

- gestión de errores

8. evolución a HTML5

gestión de errores

descripción

En JavaScript los errores se gestionan a través de bloques try-catch. El control de excepciones se estructura en dos bloques:

```
try {  
    //Bloque a probar  
} catch (e) {  
    //Manejo del error  
}
```

Incluye las instrucciones normales de código. Si se detecta un error se interrumpe el flujo de ejecución y se pasa a ejecutar el bloque catch.

Incluye las instrucciones que tratan el error generado. Dispone de una referencia al error generado, que contiene información sobre la causa y la trazabilidad.

gestión de errores

descripción

Existen cinco tipos de errores primarios:

- **EvalError**. Se causa cuando se usa incorrectamente la función `eval()`.
- **RangeError**. Se lanza cuando una variable numérica se desborda.
- **ReferenceError**. Sucede cuando se accede a una referencia inválida.
- **SyntaxError**. Ocurre al producirse un error sintáctico en la lectura del código.
- **TypeError**. Se origina cuando el tipo de la variable no es el esperado.

Además, se pueden crear nuevos errores, utilizando:

```
throw new Error("mensaje de error");
```

8 evolución a HTML5

- **historia y evolución de HTML**
- adaptación a HTML5
- nuevas etiquetas de contenido
- nuevas etiquetas de formulario
- layout
- etiquetas eliminadas
- validación de código

historia y evolución de HTML

versiones de HTML

- **HTML 1.0 y 2.0.** Surgió con la masificación de Internet. Un contenido estático, bastante básico pero con hipertextos. Tenía una orientación a mostrar contenidos simples.
- **HTML 3.0 y 3.2.** Propuesto por el recién formado W3C. Incluyó facilidades para crear tablas, flujo del texto alrededor de figuras y muestra de fórmulas complejas. Se abandonó por la falta de apoyo de los fabricantes de navegadores web.

La versión 3.2 abandonó muchas características de la 3.0 y adoptó otras de los navegadores Netscape y Mosaic. Se incluye MathML para representar las fórmulas matemáticas. Durante este tiempo se vivió la "guerra de los navegadores", entre Netscape e Internet Explorer, lo que produjo muchas diferencias entre ambos.

1989

1991

1995

historia y evolución de HTML

versiones de HTML

- **HTML 4.0.** Fin de la "guerra de los navegadores", con el plan de la W3C de crear "un solo HTML". Planteó la separación entre contenido y estilos gráficos, a través de las hojas CSS. Incluyó muchos elementos específicos de navegadores concretos a la vez que comenzó a limpiar el lenguaje, declarando algunas etiquetas como “desaconsejadas”. Estandarizó los marcos (frame), que luego quedaron en desuso.
- **HTML 4.01.** Incorporó algunos cambios de detalle a la versión 4.0. Fue una versión suficientemente aceptada como para no ser modificada significativamente en mucho tiempo.

1998

1999

historia y evolución de HTML

versiones de HTML

- **XHTML 1.0.** Unió el conceptos de XML, que estaba en pleno proceso de asimilación y HTML, con el objetivo de hacerlo más riguroso. La recepción de XHTML fue variada, pues la rigurosidad fue percibida también como falta de flexibilidad.
- **HTML 5.** Extiende a HTML 4.01, agregando nuevas características con una orientación interactiva (capacidades gráficas, video, transformando la visión HTML de "páginas" por la de "aplicación". Descarta aquellos que están en desuso. Está disponible paulatinamente desde 2012, pues los navegadores la van incorporando poco a poco.

2001

2012

conceptos

notas sobre XHTML

XHTML (eXtensible Hyper Text Markup Language) es, en términos simples, HTML cumpliendo especificaciones de XML, más estrictas. Por ejemplo:

- Se deben **cerrar** todas las etiquetas:

`
` ✓

`` ✓

`
` ✗

`` ✗

- Los atributos de las etiquetas deben tener **comillas**, simples o dobles:

`<input type="text" />` ✓

`<input type=text>` ✗

- Se debe utilizar **minúsculas** en etiquetas y atributos:

`<ol type="A">` ✓

`<OL TYPE="A">` ✗

- Todos los atributos deben tener un **valor**, no usando forma minimizada:

`<input disabled="disabled" />` ✓

`<input disabled>` ✗

conceptos

notas sobre XHTML

Además de ser más estricto por la aplicación de XML, XHTML tiene otras características:

- Su objetivo es lograr páginas web donde la información y la forma de presentarla estén claramente **separadas**.
- XHTML es por tanto, un lenguaje semántico en el que no se define el aspecto de las cosas sino lo que significan.
- Se utiliza únicamente para transmitir los **contenidos** de un documento, dejando el aspecto y diseño para las hojas de estilo CSS y la interactividad y funcionalidad a JavaScript.

8 evolución a HTML5

- **adaptación a HTML5**
 - nuevas etiquetas de contenido
 - nuevas etiquetas de formulario
 - layout
 - etiquetas eliminadas
 - validación de código

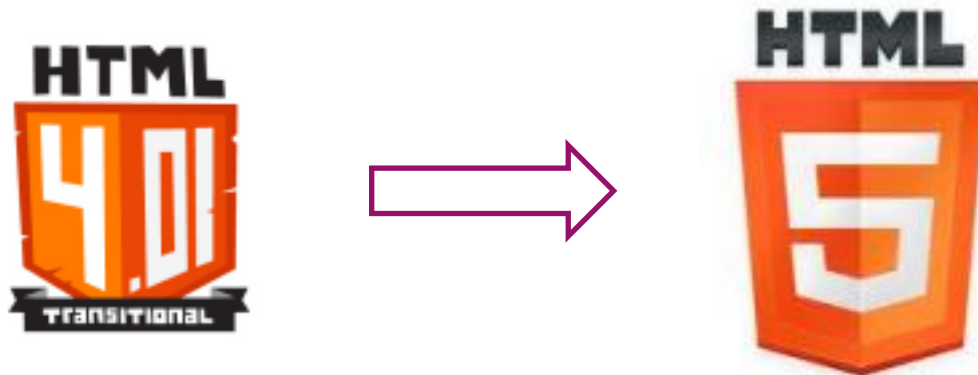
adaptación a HTML 5

diferencias entre HTML 4.01 y 5

En HTML 4.01, se utiliza una nomenclatura en la página, que es modificada en HTML 5:

- Declaración del doctype
- Declaración del juego de caracteres
- Hoja de estilos
- JavaScript

A continuación, se presentan las diferencias con ejemplos.



adaptación a HTML 5

doctype

En HTML 4.01, el doctype que es especifica en el encabezado es uno de los siguientes, dependiendo si es "strict" o "transitional":

```
<!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 Transitional//EN"  
"http://www.w3.org/TR/html4/loose.dtd">
```

En HTML5 se simplifica:

```
<!DOCTYPE html>
```

No especifica un "5". Se simplifica bastante.

Según la W3C, todas las futuras versiones de HTML (6, 7, ...) utilizarán este doctype, no siendo necesario actualizarlo.

adaptación a HTML 5

juego de caracteres

El juego de caracteres de la página, en HTML 4.01 "strict" o "transitional", se especifica así, por ejemplo con utf-8 (recomendable):

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

En HTML5 se simplifica, utilizando un atributo específico más compacto:

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

Esto facilita la declaración.

adaptación a HTML 5

declaración de hoja de estilos

En HTML 4.01, la hoja de estilos se declara así:

```
<link type="text/css" rel="stylesheet" href="styles.css">
```

En HTML5 no es necesario utilizar el **type**, ya que CSS se considera el estándar y default:

```
<link rel="stylesheet" href="styles.css">
```



adaptación a HTML 5

declaración de JavaScript

En HTML 4.01, la utilización de JavaScript se declara así:

```
<script type="text/javascript" src="lib.js"></script>
```

```
<script type="text/javascript">  
    var name = "basic";  
</script>
```

En HTML5 no es necesario utilizar el **type**, ya que JavaScript se considera el estándar y default:

```
<script src="lib.js"></script>
```

```
<script>  
    var name = "basic";  
</script>
```

adaptación a HTML 5

soporte de navegador

Los navegadores no han implementado todas las especificaciones de HTML5 en una release, sino que las van liberando poco a poco. Algunos tienen más grado de cumplimiento que otros. Esto se puede observar en <http://html5test.com>:

browsers

Select up to three browsers and compare their test results in detail

Chrome 28	Firefox 22	Internet Explorer 10
463	410	320
13 bonus points	14 bonus points	6 bonus points

Sólo para Windows 8.
Internet Explorer 9 tiene
138 de 500.

adaptación a HTML 5

grado de cumplimiento

Ejemplos de algunos ítems (se presentan 4 de los 23):



29.0

Security	15/20		
Web applications	18/20	✓	✗
User interaction	20	✓	✓
Forms	110/115	✗	✓
Field types			
▶ input type=text	Yes	✓	✓
▶ input type=search	Yes	✓	✓
▶ input type=tel	Yes	✓	✓
▶ input type=url	Yes	✓	✓
▶ input type=email	Yes	✓	✓
▶ input type=datetime	No	✗	
▶ input type=date	Yes	✓	



23.0

Security	10/20		
Web applications	20	✓	✗
User interaction	20	✓	✗
Forms	65/115	✓	✓
Field types			
▶ input type=text	Yes	✓	✓
▶ input type=search	Yes	✓	✓
▶ input type=tel	Yes	✓	✓
▶ input type=url	Yes	✓	✓
▶ input type=email	Yes	✓	✓
▶ input type=datetime	No	✗	
▶ input type=date	No	✗	



10.0

Security	10/20		
Web applications	16/20	✓	✗
User interaction	2/20	✓	✗
Forms	57/115	✗	✓
Field types			
▶ input type=text	Partial	○	✗
▶ input type=search	Yes	✓	
▶ input type=tel	Yes	✓	✗
▶ input type=url	Yes	✓	✗
▶ input type=email	Yes	✓	✗
▶ input type=datetime	No	✗	
▶ input type=date	No	✗	

soporte de navegador

soporte para IE7 e IE8

Las versiones de Internet Explorer 7 y 8 no soportan HTML5. Sin embargo, existen unas librerías javascript que permiten adaptarlos para que puedan ejecutarlos. Para ello, se coloca en la cabecera de la página lo siguiente:

```
<!--[if IE 7 ]> <html xmlns="http://www.w3.org/1999/xhtml" xml:lang="es" lang="es"
dir="ltr" class="ie7"> <![endif]-->
<!--[if IE 8 ]> <html xmlns="http://www.w3.org/1999/xhtml" xml:lang="es" lang="es"
dir="ltr" class="ie8"> <![endif]-->
<!--[if (gt IE 9)|!(IE)]><!--> <html xmlns="http://www.w3.org/1999/xhtml" xml:lang="es"
lang="es" dir="ltr"> <!--<![endif]-->
```

Y más abajo, dentro del head:

```
<!--[if IE 7 ]> <!--
<link rel="stylesheet" type="text/css" href="css/ie7.css" />
<script type = "text/javascript" src = "js/html5shiv.js"></script>
<![endif]-->

<!--[if IE 8 ]>
<script type = "text/javascript" src = "js/html5shiv.js"></script>
<link rel="stylesheet" type="text/css" href="css/ie8.css" />
<![endif]-->
```

Para HTML son comentarios, pero IE los interpreta.

CSS y JavaScript para emular HTML 5 en IE 7 y en IE 8

8 evolución a HTML5

- **nuevas etiquetas**
- layout
- etiquetas eliminadas
- validación de código

nuevas etiquetas

resumen

HTML5 agregan nuevas etiquetas que se agrupan en los siguientes tipos:

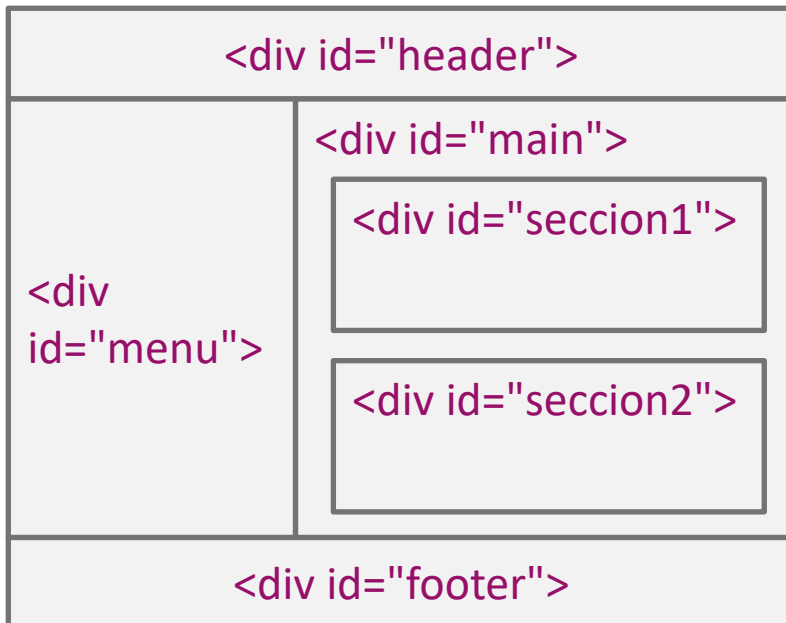
- **canvas**: para dibujar gráficos con JavaScript.
- **media**: audio, video, contenido interactivo.
- **formulario**: nuevos tipos de campos de entrada, autocompletado.
- **estructurales**: definición de elementos como header, section, footer, article.

nuevas etiquetas

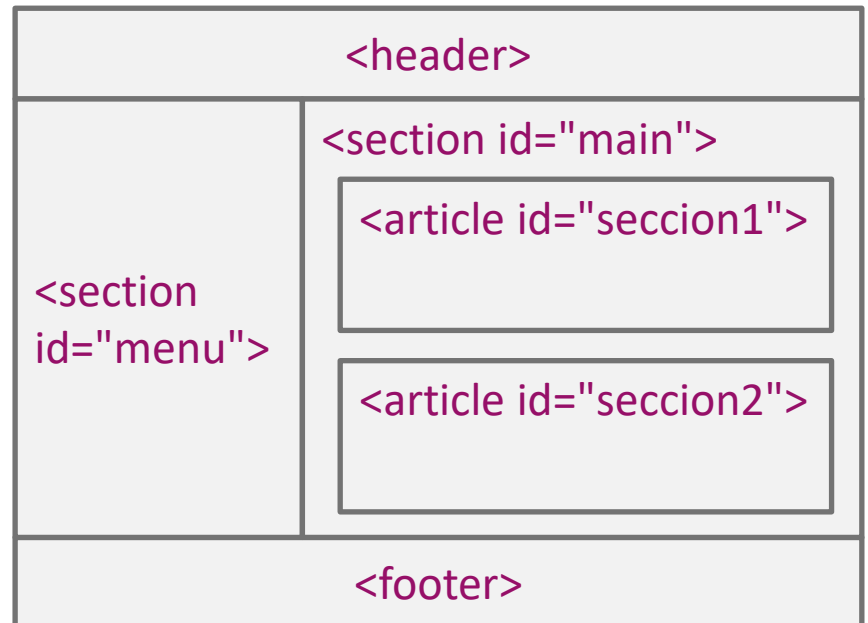
estructurales

En HTML 4, se utilizan etiquetas `<div>` para organizar las secciones de la página. En HTML5, se agregan nuevas etiquetas para la organización estructural. Ejemplo:

HTML 4



HTML 5



nuevas etiquetas

campos nuevos en formularios

HTML 5 incorpora las siguientes mejoras en los campos de formularios:

- **type**: agrega tipos específicos a los input, como "number", "date", "email", "color", y otros. Esto facilita la validación.
- **pattern**: permite agregar una expresión regular que valida el campo.

```
Cantidad: <input type="number"
             name="amount" size="10" min="0"
             max="10" /><br/>
Fecha: <input type="date"
             name="date" size="10" /><br/>
Correo: <input type="email"
             name="email" size="25" /><br/>
Color: <input type="color"
             name="color" size="10" /><br/>
Móvil (6xxxxxxxx): <input type="text"
                    name="movil" size="9"
                    pattern="[6]{1}[0-9]{8}" />
```

Cantidad:

Fecha:

Correo:

Color:

Móvil (6xxxxxxxx):

nuevas etiquetas

campos nuevos en formularios

Continuación:

- **required**: marca el campo como requerido, con lo que el propio browser valida que sea completado. Formatos:

```
<input ... required .../>  
<input ... required="required" .../>
```

- **datalist**: complementa a un textbox con autocompletado basado en una lista.

```
Nombre: <input type="text"  
        name="nombre" size="10" list="nombres" />  
<datalist id="nombres">  
  <option label="Juan" value="a1"/>  
  <option label="Luis" value="a2"/>  
  <option label="Jorge" value="b3"/>  
</datalist>
```

Nombre:

a	
a1	Juan
a2	Luis

Nota: A junio 2013, algunas de estas características están disponibles en Chrome (23+), Opera (12+) e IExplorer 10, y la mayoría no está disponibles en FireFox (22) e IExplorer 8 y 9.

8 evolución a HTML5

- **layout**
- etiquetas eliminadas

layout

diseño

El layout se refiere a la forma como se organiza el contenido de un conjunto de páginas. Depende del diseño y del tipo de aplicación.

Para ilustrarlo, se presenta el prototipo del caso de negocio.

8 evolución a HTML5

- **etiquetas eliminadas**
- validación de código

etiquetas eliminadas

descripción

Dado que las hojas de estilo están asimiladas y son un estándar, se eliminan las etiquetas de formato de las versiones de HTML antiguas: `<basefont>`, `<center>`, ``, `<strike>` y `<tt>` .

También se eliminan las etiquetas de uso de `frames` y `applets`.

Ejemplo de reemplazo:

```
<center>  
  <h3>Título</h3>  
  <p>Contenido</p>  
</center>
```



```
<div style="text-align: center">  
  <h3>Título</h3>  
  <p>Contenido</p>  
</div>
```

etiquetas eliminadas

descripción

Dado que las hojas de estilo están asimiladas y son un estándar, se eliminan las etiquetas de formato de las versiones de HTML antiguas: `<basefont>`, `<center>`, ``, `<strike>` y `<tt>` .

También se eliminan las etiquetas de uso de `frames` y `applets`.

Ejemplo de reemplazo:

```
<center>  
  <h3>Título</h3>  
  <p>Contenido</p>  
</center>
```



```
<div style="text-align: center">  
  <h3>Título</h3>  
  <p>Contenido</p>  
</div>
```