

skip-help: true
title: Clase 1 Proyecto 2016
Author: Einar Lanfranco, Claudia Banchoff, Matías Pagano, Diego Vilches
description: HTML
keywords:
css: presentacion.css

Proyecto de Software

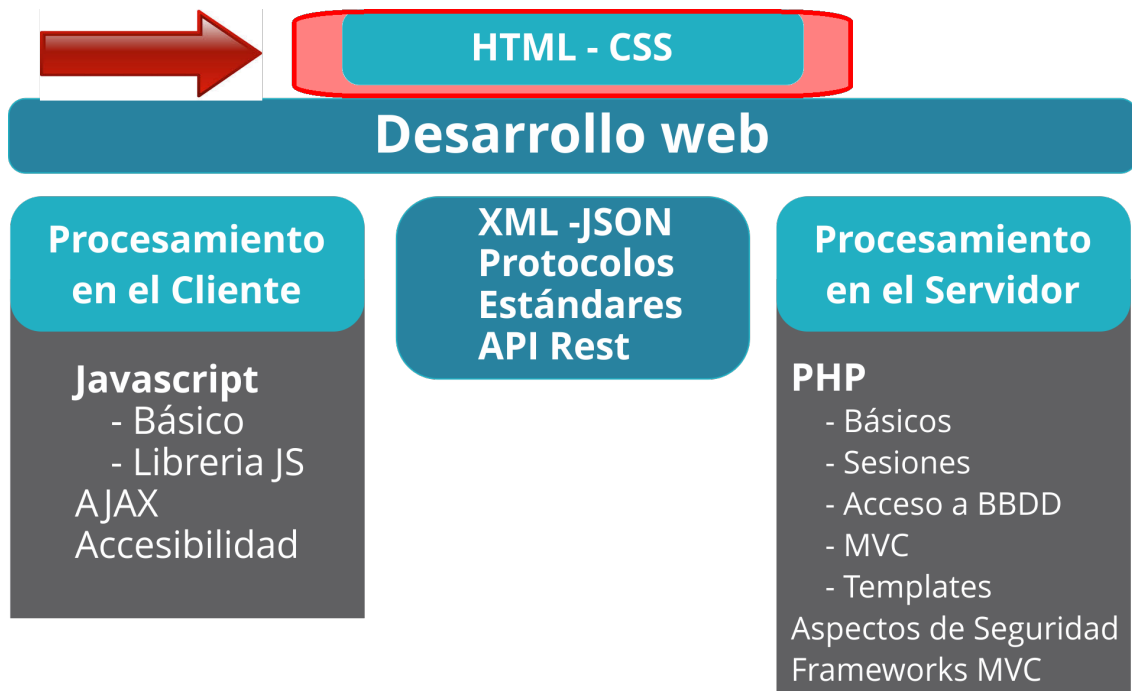
Cursada 2016

data-rotate: 270

Temario de la materia



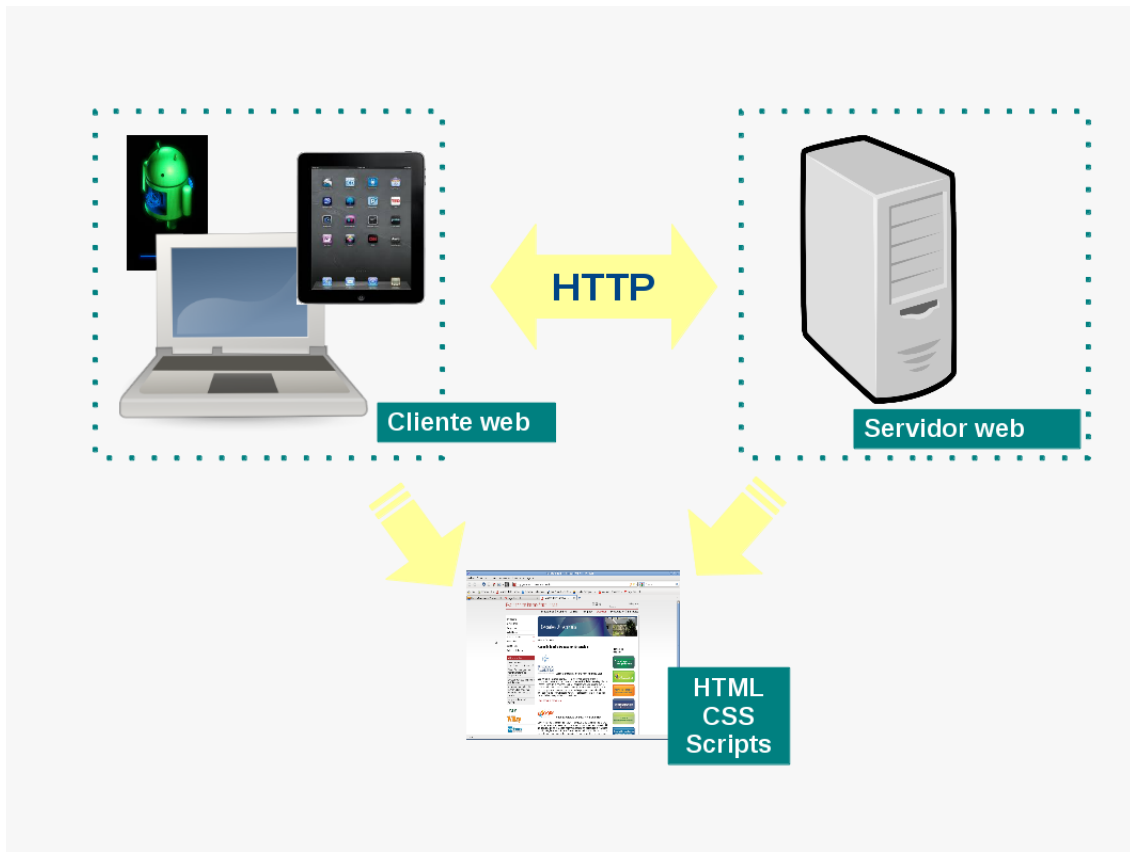
Hoy empezamos ...



Temario de hoy

- **La web**
 - Arquitectura
 - Web 2.0, web semántica ...
 - **Aspectos básicos**
 - Definición de URL/URI
 - Protocolo HTTP
 - Lenguaje HTML
 - Estilos
-

Arquitectura web básica



Internet y la web

RFCs – Request for Comments

- <http://www.faqs.org>

Algunas estadísticas

- <http://www.internetworldstats.com/>
- <http://news.netcraft.com/archives/category/most-popular/>
- <http://gs.statcounter.com/>

class: destacado

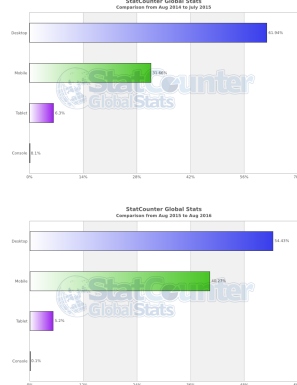
W3C – El consorcio de la web

- <http://www.w3c.org>
- Desarrollo de estándares y guías.

La misión del W3C es:

Guiar la web hacia su máximo potencial a través del desarrollo de protocolos y pautas que aseguren el crecimiento futuro de la web.

Tendencias ...



<http://gs.statcounter.com/#all-comparison-ww-monthly-201508-201608-bar>

La World Wide Web

- La web es una red de recursos de información.
 - **Se basa en tres pilares básicos:**
 - El concepto de URL/URI
 - El protocolo HTTP
 - El lenguaje HTML
-

URI/URL - RFC 3986

- Una **URI** -“**Uniform Resource Identifier**”- es un mecanismo por el cual se identifica todo recurso accesible en la web.
 - Una **URL** -“**Uniform Resource Locator**”- permite ubicar un recurso a través de su ubicación.
 - **Típicamente una URI se compone de:**
 - el esquema o **protocolo** utilizado para acceder al recurso;
 - el **nombre de dominio** de la máquina que almacena el recurso;
 - el **nombre del recurso** mismo dado como un camino dentro de la máquina (recurso)■.
-

URI/URL - RFC 3986 (cont.)

Ejemplos típicos

```
http://www.servidor.com.ar/especificacion#parte3  
https://www.taller.com.ar/info.php?id=12&qq=11  
mailto:proyecto@info.unlp.edu.ar  
file:///home/clau/git/proyecto2014/teorias2015/clase1/index.html
```

URL Encoding

- Las URLs se transmiten en ASCII.
- Algunos caracteres deben convertirse.
- Ejemplos:

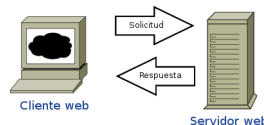
```
/Clase 1/EjemplosClase1/Ejemplo con enlaces.html
```

Luego:

```
../Clase%201/EjemplosClase1/Ejemplo%20con%20enlaces.html
```

class: destacado

Protocolo HTTP



- Una transacción HTTP consta de 4 pasos:
inicio conexión - solicitud - respuesta - cierre conexión

- **Protocolo sin estado**
- Clientes web: Firefox, IE, Chrome, Opera,
- Servidores web: Apache, IIS, Nginx, etc,

Protocolo HTTP

Versiones:

- HTTP 1.0 - RFC 1945 (1996)
- HTTP 1.1 - RFC 2068/2616 (1999)
- HTTP 2.0 - RFC 7540 (2015)

La mayoría de los navegadores y servidores web soportan 1.0 y 1.1, y 2.0 parcialmente.

<http://www.rfc-base.org/txt/rfc-1945.txt>

<http://www.w3.org/Protocols/rfc2616/rfc2616.txt>

<https://tools.ietf.org/html/rfc7540>

Mensajes HTTP

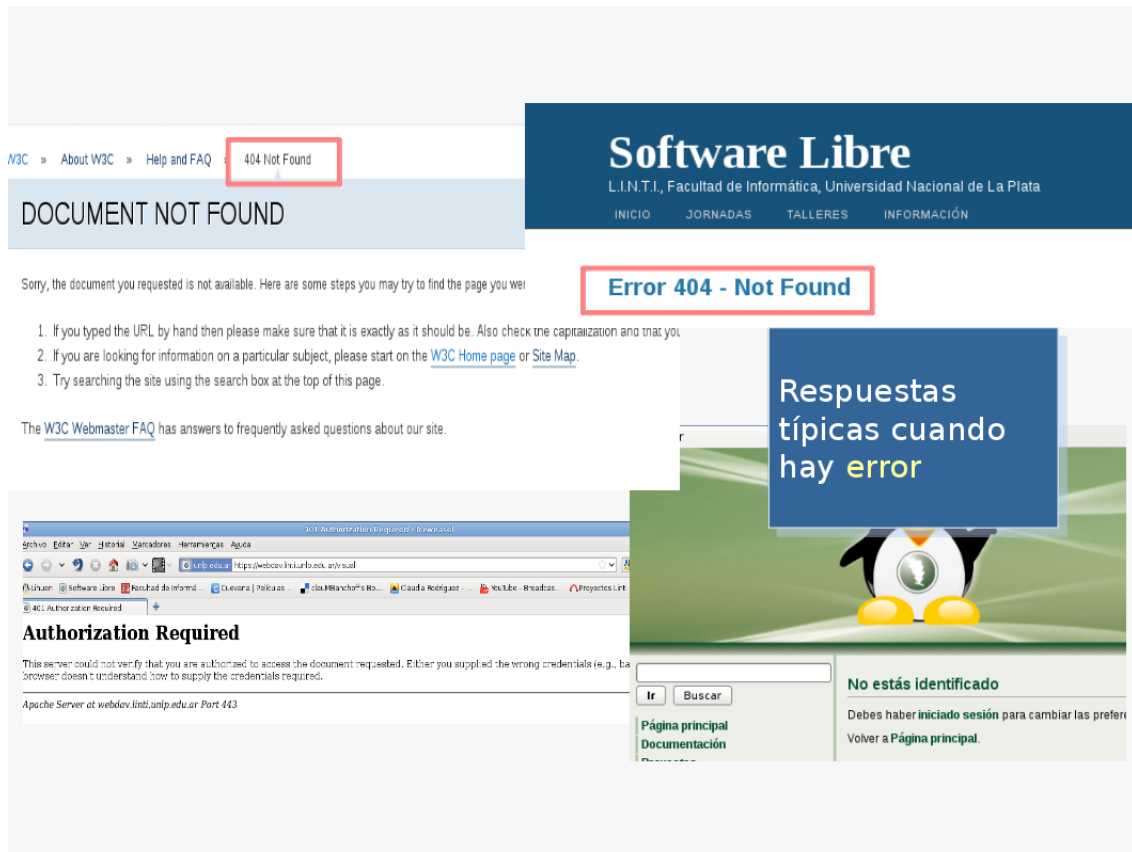
Solicitudes

- **GET**: retorna la información que está identificada por la URI-solicitada.
- **HEAD**: retorna la información del header del servidor.
- **POST**: en general se utiliza para la entrega o envío de formularios que son completados en forma interactiva por un usuario. Esta es la única solicitud que envía un cuerpo en el mensaje.

Respuestas

- El servidor retorna un código que indica el estado de la solicitud (por ejemplo: 200) y el recurso
-

Errores típicos: 404, 503, ...



Respuestas típicas

1. **1xx:** Respuestas informativas
2. **2xx:** Peticiones correctas
 - Ejemplo: 200 OK
3. **3xx:** Redirecciones
 - Ejemplo: 302 Movido temporalmente
4. **4xx:** Errores del cliente
 - Ejemplo: 404 No encontrado, 403 Prohibido o 401 No autorizado
5. **5xx:** Errores de servidor
 - Ejemplo: 500 Error interno o 503 Servicio no disponible

El Lenguaje HTML

Lenguaje HTML

- HTML - “**HyperText Markup Language**”- especifica el formato de las páginas web, separando el contenido de las páginas de su formato de presentación.
 - Fue creado en los laboratorios CERN por **Tim Berners-Lee**.
 - Define un conjunto de símbolos (etiquetas o tags) que **especifican la estructura** lógica de un documento y de todos sus componentes.
 - Es independiente de la plataforma.
 - **Su código es interpretado por los clientes web.**
-

class: tabla

HTML: Un poco de historia...

Versión	Año de publicación
HTML +	1993
HTML 2.0	1995
HTML 3.2	1997
HTML 4.01	1999
HTML 5	Rec desde Oct 2014

Estructura básica

```
<!DOCTYPE html>
<html>

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

<body>
</body>

</html>
```

HTML – Sintaxis General

La cláusula DOCTYPE

- Primera línea del documento.
- Indica la forma en que se validará el documento.

Ejemplos:

```
<!DOCTYPE html>
```

También puede ser (para versiones anteriores):

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

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

El encabezado

Delimitado por `<head>` y `</head>`

Algunas componentes

`<title>`: Corresponde al título de la página.
`<link>`: Indica una relación entre el documento y algún otro objeto de la web.
`<meta>`: Meta-información sobre el documento

Campos meta

- Son usados, entre otras cosas, por buscadores para mejorar la calidad de los resultados en las búsquedas.

Ejemplo:

```
<meta name="description" content="Proyecto de software" />
<meta name="author" content="Claudia Banchoff-Einar Lanfranco" />
<meta name="keywords" content="meteorología, clima">
```

O mejor:

```
<meta name="keywords" lang="es"
      content="deportes, tenis, futbol">
<meta name="keywords" lang="en"
      content="sports, tennis, soccer">
```

Campos meta (cont.)

- Sugerencias para los robots de búsquedas (Alternativa: archivo robots.txt)

```
<meta name="robots" content="noindex, nofollow, noimageindex">
```

O:

```
<meta name="GOOGLEBOT" content="noindex, nofollow">
```

- http-equiv permite predeterminar el diálogo entre cliente y servidor

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

El cuerpo del documento

- Delimitado por **<body>** y **</body>**
- Encabezados: **<h1>..</h1>**, **<h2>..</h2>**,, **<h6>..</h6>**
- Párrafos: **<p>..</p>**.
- Comentarios: **<!-- el comentario -->**
- Imágenes: ****
- Enlaces: **<a>..**
- Listas: **..**
- Tablas, formularios... y muchos elementos más.

Consideración importante

- El documento tiene una estructura y una forma de visualización
 - **Estructura:** usando las etiquetas HTML más apropiadas.
 - **Visualización:** usando hojas de estilo.
 - Ver sitio [csszengarden](https://csszengarden.com/)
-

A tener en cuenta ...

Referencias relativas y absolutas

```


```

A tener en cuenta ...

El set de caracteres

- Los navegadores soportan varios conjuntos de caracteres: UTF8 , Unicode, ISO-8859-1 ...
- Configurados adecuadamente solucionamos problemas por ejemplo con los acentos o las ñ
- Podemos usar el campo meta http-equiv:

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

- Veamos el sitio de [Lihuen](#)
-

Entidades HTML

- Se las utiliza para representar símbolos especiales que no son representados de la misma manera en todos los set de caracteres: **símbolos matemáticos**, **caracteres especiales**, **letras acentuadas**, etc.
- **Forma general:** &nombreEntidad;

Ejemplos:

```
&amp; (&) - &copy; (©) - &lt;
(<) - &gt; (>) -&quot; (") ...
&aacute; (á) - &eacute; (é) - ....
```

- Nos independizamos del set de caracteres del navegador
-

Especificando colores

- Se utiliza notación RGB.
- Forma general: #RRGGBB



Los formularios

- Se definen con **<form>** **</form>**
- Define y agrupa los **campos** que forman el formulario.

Algunos atributos:

- **action**: Especifica la URL donde será enviado el formulario.
- **method**: Especifica la forma en que se transfieren los datos: **get** y **post**

```
<form method="post" action="info.php">  
.....  
</form>
```

Los formularios en la arquitectura web



Formularios usando GET

- Si en el formulario se definen campos **nombre** y **seccion**, por ejemplo
- Mensaje HTTP con GET:

```
GET /index.php?nombre=pepe&seccion=1 HTTP/1.0  
Host: www.servidor.com  
User-Agent: Mozilla/4.5 [en]  
Accept: image/gif, image/jpeg, text/html  
Accept-language: en  
Accept-Charset: iso-8859-1
```

Formularios usando POST

- Si en el formulario se definen campos **nombre** y **seccion**, por ejemplo
- Mensaje HTTP con POST:

```
POST /index.php HTTP/1.0
Host: www.servidor.com
User-Agent: Mozilla/4.5 [en]
Accept: image/gif, image/jpeg, text/html
Accept-language: en
Accept-Charset: iso-8859-1

nombre=pepe&seccion=1
```

GET vs. POST

Mensaje HTTP con GET:

```
GET /index.php?nombre=pepe&seccion=1
Host: www.servidor.com
User-Agent: Mozilla/4.5 [en]
Accept: image/gif, image/jpeg, t
Accept-language: en
Accept-Charset: iso-8859-1
```

GET: Debería usarse cuando el formulario no tiene efectos secundarios, por ejemplo en una búsqueda.

Mensaje HTTP con POST:

```
POST /index.php HTTP/1.0
Host: www.servidor.com
User-Agent: Mozilla/4.5 [en]
Accept: image/gif, image/jpeg, text/html
Accept-language: en
Accept-Charset: iso-8859-1
```

POST: Debería usarse en formularios que modifican una base de datos o la suscripción a un servicio.

```
nombre=pepe&seccion=1
```

Los campos del formulario

- Formulario típico: [ejemplo-formulario](#)
- Algunos elementos introducidos por HTML5: [ejemplo1-formulario-HTML5](#)
- HTML5 define más tipos de elementos input: **tel**, **search**, **url**, **email**, **date**, **number**, **color**
- Y otros tipos de elementos: **<datalist>**, **<keygen>**, **<output>**

- Ver ejemplos en: [mas-ejemplos](#)
-

Contenido multimedial

- Elementos:
 - <audio>: Distintos tipos de sonidos, música, streams de audio.
 - <video>: Contenido de video.
 - <embed>: Contenido embebido, por ejemplo un plugin
 - Ejemplos: [audio](#) y [video](#)
-

Elemento canvas

- El elemento <canvas> se utiliza para graficar (usando javascript usualmente).
 - Tiene varios métodos que permiten graficar líneas, figuras, agregar imágenes, etc.
 - Ejemplos [sencillos](#)
 - Un lindo ejemplo: [ver-ejemplo](#)
-

Contenido semántico

- HTML5 introdujo elementos estructurales: <article>, <header>, <footer>, <nav>, <section>, etc.
-

data-scale: 0.1

web, web 2, web semántica ...¿?

data-scale: 1

La web 2



- En 2004, por primera vez mencionado por Tim O'Reilly

- Los **usuarios como productores de contenidos**
 - Herramientas típicas: blogs, wikis, redes sociales.
-

La web semántica

- Incorporar metadatos para agregar **significado** a la información del documento HTML.
 - Se debe seguir un **formalismo adecuado** para que se lo pueda procesar en forma adecuada.
 - En la materia, sólo veremos algunos aspectos sobre HTML semántico.
-

Hojas de Estilo

Volvamos a ver el sitio csszengarden

- El documento tiene una estructura y una forma de visualización
 - **Estructura:** usando las etiquetas HTML más apropiadas.
 - **Visualización:** usando hojas de estilo
 - Ver sitio [csszengarden](#)
-

Hojas de Estilo

- **Describen el formato de un documento HTML. Cómo se visualizarán en los distintos medios, por ejemplo en la pantalla o en la impresora.**
 - Permiten separar el contenido de un documento HTML de la forma en que se lo visualizará
 - El estándar usado: CSS (Cascading Style Sheets)■
 - Es posible incluir las definiciones dentro de la página o en un archivo separado (a partir del HTML 4.0).
-

Estado Actual

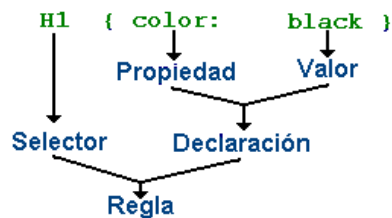
- Está definido en distintos módulos.
- <http://www.w3.org/Style/CSS/current-work>

Completed work	Status	Upcoming	iO
CSS Snapshot 2010	NOTE		iO
CSS Snapshot 2007	NOTE		iO
CSS Color Level 3	REC	REC	iO
CSS Namespaces	REC	REC	iO
Selectors Level 3	REC	REC	iO
CSS Level 2 Revision 1	REC	REC	iO
CSS Level 1	REC		iO
CSS Print Profile	NOTE		iO
Media Queries	REC	REC	iO
CSS Style Attributes	REC	REC	iO

Lo básico

- **Hoja de estilo:** Conjunto de reglas, que definen un estilo para cada elemento o grupos de elementos HTML.
- **Una regla de estilo tiene dos partes:**
 - Un **selector**, que identifica el elemento o grupo al que el estilo se aplicará.
 - Una **declaración de propiedades** a ser aplicadas al selector.

Lo básico (cont.)



Lo básico (cont.)

- Sintaxis:

```
selector { propiedad: valor;
           propiedad: valor...
}
```

- Ejemplo:

```
H1 { color: black}
```


¿Para qué se usan?

- Propiedades relativas a tipos de letra: *font-family*, *font-size*, *font-weight*, *font-style*
 - Propiedades del texto: *line-height*, *text-decoration*, *vertical-align*, *text-transform*, *text-align*
 - Propiedades de bloques: *margin-top*, *margin-right*, *margin-bottom*, *margin-left*, *padding-top*, *padding-right*, *padding-bottom*, *padding-left*, *border-color*, *border-style*
 - Otras propiedades: *color*, *background*, *display*, *list-style*
-

Se asocian a través de:

- Atributo style sobre tag HTML

```
<etiqueta style="prop1:valor1; prop2:valor2; ...">
</etiqueta>
```

- Etiqueta style

```
<style> selector {prop1:valor..}</style>
```

- Archivos externos

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

Escribiendo reglas

- Para reducir el tamaño de las hojas de estilo, es posible agrupar los selectores en una lista separada por comas:

```
h1, h2, h3 { font-family: Arial}
```

- Comentarios entre /* */

```
h2 { color: yellow } /* Los subtítulos van en amarillo */
```

- Los estilos se “heredan”. Las propiedades definidas para un elemento se trasladan a los elementos que éste “encierra”.

```
body { color: blue; }
h1 { color: red; } /* Todos azules menos los h1. */
```

CSS – Clases

¿Qué pasa si queremos aplicar una misma regla a varios elementos de distinto tipo? Ejemplo: algunos párrafos y algunos encabezados en rojo y el resto en el color por defecto.

```
<style>
.rojo { color: #FF0000 }
</style>
```

En la página:

```
<h1 class="rojo"> Este encabezado es rojo</h1>
<h1> Este encabezado tiene el color por defecto</h1>
<p class="rojo">Este párrafo es rojo</p>
<p>Este párrafo no</p>
```

CSS – Clases

- El atributo class está definido para casi todos los elementos HTML. Se pueden definir clases específicas para un tipo de elemento o genéricas
- Ejemplo:

```
<style>
  p.rojo { color: #FF0000 }
  .parrafoRojo { color: #FF0000 }
</style>
```

La última regla se puede aplicar a cualquier elemento, no sólo a elementos **<p>**

CSS – Pseudo Clases

- **Pseudo clases:** Son agregadas por el browser, y referidas como "clases" por las reglas de estilo. Permiten diferenciar diferentes usos para un mismo elemento.
- Ejemplo Típico:

```
a:link { color: red }
a:visited { color: blue }
```

CSS – Pseudo elementos

- **Pseudo elementos:** Permiten referirse a porciones de los elementos reales.
- Ejemplo típico:

```
p:first-line { font-variant: small-caps }
p:first-letter { font-size: 220%; float: left }
```

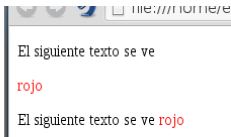
El atributo id

```
<style>
#destacado {color:red;
padding: 2px;
margin: 1em;
border-style: dashed;
line-height: 2.4em;}
</style>
...
<p> Este es un texto cualquiera </p>
<p id="destacado"> Este texto es especial </p>
<p> Este también es cualquier texto</p>
```

HTML – Etiquetas div y span

- Se utilizan para aplicar estilos a uno o más elementos.
- ¿Cuál es la diferencia?

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8"/>
    <title>Div vs SPAN</title>
    <style>
      .rojo {color:red;}
    </style>
  </head>
  <body>
    <p>El siguiente texto se ve <div class="rojo"> rojo</div></p>
    <p>El siguiente texto se ve <span class="rojo"> rojo</span></p>
  </body>
</html>
```

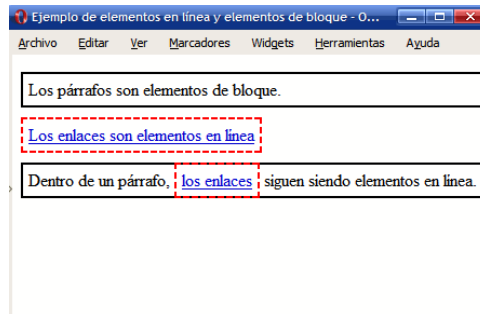


Elementos en línea o en bloque

- Algunos **elementos en línea** definidos por HTML: a, img, input, select, span, textarea.
 - Algunos **elementos de bloque** definidos por HTML: blockquote, div, dl, form, h1, h2, h3, h4, h5, h6, ol, p, table, ul.
 - Los siguientes elementos también se considera que son de bloque: dd, dt, li, tbody, td, tfoot, th, thead, tr.
 - Los siguientes elementos pueden ser en línea y de bloque según las circunstancias: button, iframe, map, object, script.
-

Modelo de cajas

- Todos los elementos de un doc HTML se representan mediante cajas rectangulares.
- Cajas creadas por los elementos de línea y los elementos de bloque

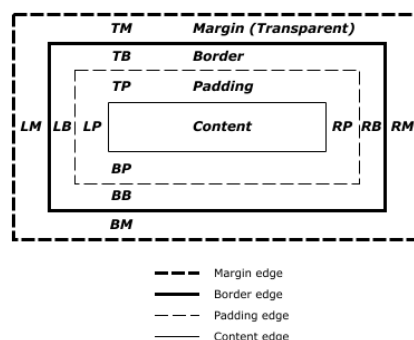


Modelo de cajas

- Propiedad **display**: determina el tipo de caja.
- Algunos valores:
 - **block** : se genera una caja de bloque principal
 - **inline-block**: genera una caja de bloque, la cual fluye como una caja en línea
 - **inline**: genera una o más cajas en línea

```
p {display: inline}
...
<p> El siguiente párrafo</p>
<p> continúa o salta de línea?</p>
```

Sobre márgenes, bordes, etc



CSS – Posicionamiento

- Permite organizar el documento en bloques o cajas.
 - Configurando algunas propiedades estos bloques pueden ocultarse o solaparse unos con otros.
 - Muchos efectos se logran con programación.
 - Usando Javascript por ejemplo
-

Esquemas de posicionamiento

- Propiedades `position` y `float`: determinan el algoritmo de posicionamiento usado para calcular la posición de la caja.

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN">
<html>
  <head>
    <title>Comparison of positioning schemes</title>
    <style>
      body { display: block; font-size:12px; line-height: 200%;
        width: 400px; height: 400px }
      p { display: block }
      span { display: inline }
    </style>
  </head>

  <body>
    <p>Principio del contenido de body.
    <span id="externo"> Comienzo contenido externo.
    <span id="interno"> Contenido interior </SPAN>
    Fin del contenido externo</SPAN>
    Fin del contenido de body
  </p>
</body>
</html>
```

Esquemas de posicionamiento

Principio del contenido de body. Comienzo contenido externo.
Contenido interior Fin del contenido externo Fin del contenido
de body

Flujo normal

```
#externo { color: red }
#interno { color: blue }
```

Principio del contenido de body. Comienzo contenido externo.
Contenido interior Fin del contenido externo Fin del contenido
de body

Flujo relativo

```
#externo { position: relative; top: -12px; color: red }
#interno { position: relative; top: 12px; color: blue }
```

Principio del contenido de body. Comienzo contenido externo.
Contenido interior Fin del contenido externo Fin del contenido de body

Flujo flotante

```
#externo { color: red }
#interno { float: right; width: 130px; color: blue }
```

Principio del contenido de body. Comienzo contenido externo.
Fin del contenido externo Fin del contenido de body Contenido interior

Flujo absoluto

```
#externo {position: absolute;top: 200px;
          left: 200px; width: 200px; color: red;}
#interno { color: blue }
```

Principio del contenido de body. Fin del contenido de body

Comienzo contenido externo.
Contenido interior Fin del contenido externo

Flujo absoluto (2)

```
#externo { position: relative; color: red}
#interno {position: absolute; top: 200px; left: -100px;
          height: 130px; width: 130px; color: blue;}
```

Principio del contenido de body. Comienzo contenido externo.
Fin del contenido externo Fin del contenido de body

Contenido interior

Estilos

- Define también otras propiedades de visualización que las verán en la práctica: fuentes, colores, cursor, etc.
- Permiten especificar cómo presentar el documento en diferentes medios: en la pantalla, en el papel, con un sintetizador de voz, con un dispositivo braille, etc.

```
@media print { body { font-size: 10pt } }  
@media screen { body { font-size: 13px } }  
@media screen, print { body { line-height: 1.2 } }
```

Estilos

Importante

- Utilizar los elementos HTML según el objetivo para lo que fueron creados.
- Ejemplo, no cambiemos los atributos de un elemento p para que sea un encabezado o un bloque inline.

Validadores

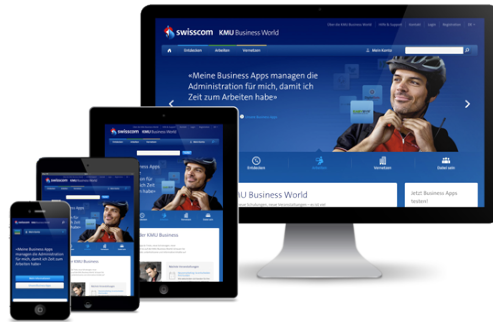
- Permiten verificar el cumplimiento de los estándares.
- La W3C provee algunos:
- Validador HTML: <http://validator.w3.org/>
- Validador de Hojas de Estilos: <http://jigsaw.w3.org/css-validator/>
- Unicorn <https://validator.w3.org/unicorn/>

Web responsive

- **Definición:** El diseño web adaptable o adaptativo, conocido por las siglas RWD (del inglés, Responsive Web Design) es una filosofía de diseño y desarrollo cuyo objetivo es adaptar la apariencia de las páginas web al dispositivo que se esté utilizando para visualizarla.

Fuente: [wikipedia](#)

Web responsive - Gráficamente



Se pueden hacer cosas como

- **Importar el css si es para imprimir en superficies mayores a 25cm**
`<link rel="stylesheet" media="print and (min-width: 25cm)" href="http://..." />`
 - **Definir en el css algo aplicable sólo a screen entre 400 y 700 pixeles**
`@media screen and (min-width: 400px) and (max-width: 700px) { ... }`
 - **Ver más:**
 - <http://www.w3.org/TR/css3-mediaqueries/#media1>
 - <http://www.w3.org/TR/mediaqueries-4/>
-

Algunos tips:

Escenciales:

- Chromium/Chrome: DevTools -> F12
- Firefox: Firefox Developer Tools -> F12

Adicionales:

- Para Todos: Web developer
 - Para Firefox: Tamper Data
-

Referencias (1)

- La web: la propuesta original: <http://www.w3.org/History/1989/proposal.html>
 - La evolución de la web: <http://evolutionofweb.appspot.com/>
 - Estándares web: http://www.w3.org/community/webed/wiki/Main_Page
 - Lenguaje HTML: <http://www.w3c.org/html>
 - HTML5 vs. anteriores: <http://www.w3.org/TR/html5-diff/>
 - Motores de búsqueda: <http://www.robotstxt.org>
 - Algunos Libros en la web: <http://www.librosweb.es>
 - Tutoriales básicos: <http://www.w3schools.com>
 - <http://diveintohtml5.info/>
-

Referencias (2)

- **Web 2.0**
 - <http://www.internality.com/web20/>
 - http://fundacionorange.es/areas/25_publicaciones/publi_253_11.asp
- **Web semántica**
 - <http://www.w3c.es/Divulgacion/GuiasBreves/WebSemantica>
 - <http://microformats.org/>