Servicios y Aplicaciones Telemáticas (2024-25)

Grado en Ingeniería Tecnologías de Telecomunicación (URJC)

Jesús M. González Barahona, Gregorio Robles, David Moreno,
Alberto Rodríguez, Alberto García, Sergio Montes

http://cursosweb.github.io
GSyC, Universidad Rey Juan Carlos

13 de marzo de 2025

Presentación de la asignatura

Datos, datos, datos

- Profesores:
  - Jesús M. González Barahona (jgb @ gsyc.urjc.es)
  - Gregorio Robles (grex @ gsyc.urjc.es)
  - David Moreno Lumbreras (david.morenolu @ urjc.es)
  - Alberto Rodríguez Iglesias (alberto.rodriguezi @ urjc.es)
  - Alberto García (@ urjc.es)
  - Sergio Montes (sergio.montes @ urjc.es)
- Grupo de Sistemas y Comunicaciones (GSyC)
- Horario: M (11:00-13:00) y J (11:00-13:00)
- Tutoría: Horario por definir (via videoconferencia)

Clases: Laboratorios III, L3209

Campus virtual: `http://aulavirtual.urjc.es`

Cursos web: `http://cursosweb.github.io`

GitLab de la EIF: `http://gitlab.eif.urjc.es`

¿De qué va todo esto?

Entendiendo cómo funciona la web

En concreto...

- Cómo se construyen los sistemas reales que se usan en Internet
- Qué tecnologías se están usando
- Qué esquemas de seguridad hay
- Cómo encajan las piezas
- En la medida de lo posible, “manos en la masa”

Ejemplos

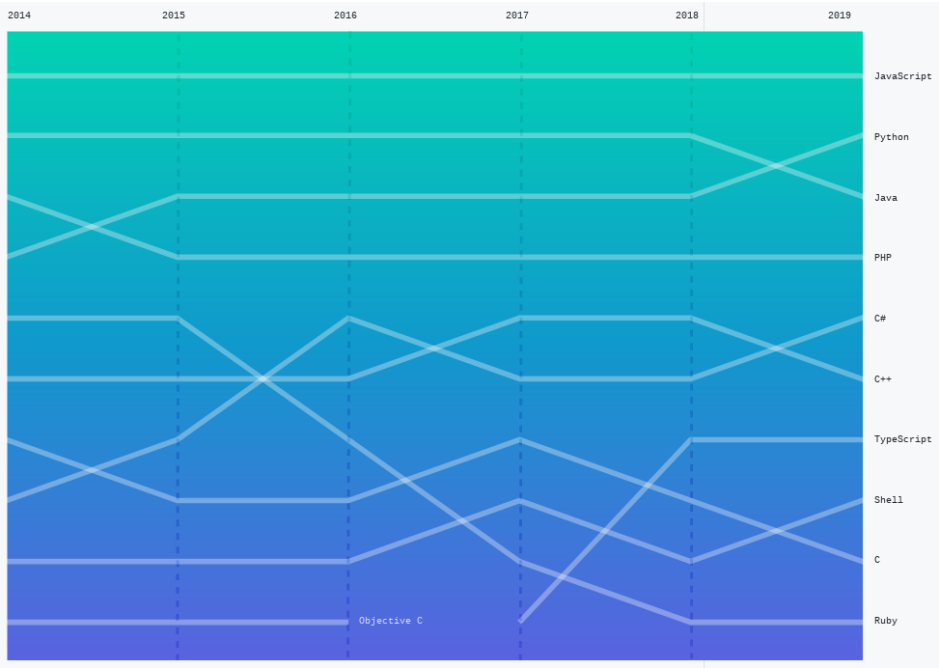
- ¿Qué es una aplicación web?
- ¿Qué es una sesión?
- ¿Cómo construir un servicio REST?
- Acaba con la magia de los servicios web
- ¿Cómo se hace un servicio basado en contenidos?

Fundamentos de la asignatura

Lenguaje de Programación: Python

Jan 2020	Jan 2019	Change	Programming Language	Ratings	Change
1	1		Java	16.896%	-0.01%
2	2		C	15.773%	+2.44%
3	3		Python	9.704%	+1.41%
4	4		C++	5.574%	-2.58%
5	7	↑	C#	5.349%	+2.07%
6	5	↓	Visual Basic .NET	5.287%	-1.17%
7	6	↓	JavaScript	2.451%	-0.85%
8	8		PHP	2.405%	-0.28%
9	15	⇈	Swift	1.795%	+0.61%
10	9	↓	SQL	1.504%	-0.77%

13	10	↓	Objective-C	0.929%	-0.85%
14	16	↑	Go	0.900%	-0.22%
15	14	↓	Assembly language	0.877%	-0.32%
16	20	⇈	Visual Basic	0.831%	-0.20%
17	25	⇈	D	0.825%	+0.25%
18	12	⇊	R	0.808%	-0.52%
19	13	⇊	Perl	0.746%	-0.48%
20	11	⇊	MATLAB	0.737%	-0.76%



Primer Mandamiento:
Amarás Python por encima de (casi) todo.

TIOBE (enero 2020) GitHub (noviembre 2019)

`https://www.youtube.com/watch?v=UNSoPa-XQNO`

Plataforma: Django

Primera Generación	Segunda Generación	Tercera Generación
HTML	PHP	RAILS
CGI	JSP	**DJANGO**
	PERL	SYMFONY

Segundo Mandamiento:
No tomarás el nombre de Django en vano.

Plataforma: Django (2)

Algunas referencias a principales marcos de desarrollo web (2025):

- "10 Software Development Frameworks That Will Dominate 2025"
  `https://www.index.dev/blog/10-programming-frameworks`
- "Web Application Development — Top 10 Frameworks in 2025"
  `https://www.sencha.com/blog/web-application-development-top-frameworks/`
- "The Top 10 Frameworks for Web Development in 2025"
  `https://medium.com/@technosoftwares21/the-top-10-frameworks-for-web-development-in-2025-4a7fc54f47dc`

Suele ser el preferido en Python, y uno de los 10 principales.

Metodología

Objetivo principal:
conceptos básicos de construcción de sitios web modernos

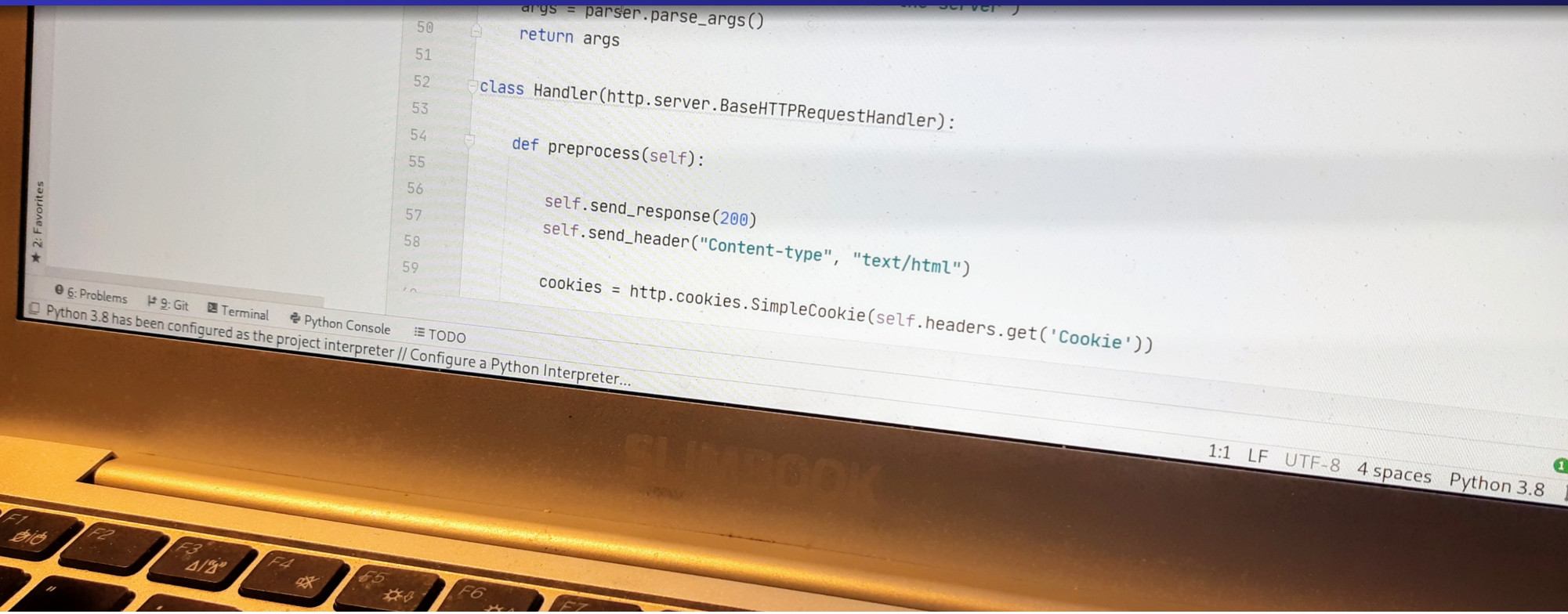
- Clases de teoría y de prácticas, pero...
- Teoría en prácticas, prácticas en teoría
- Uso de resolución de problemas para aprender
- Construcción de un proyecto a lo largo del curso

Fundamentalmente, entender lo fundamental

Fundamentos de la asignatura

Aprender no puede ser aburrido

Fundamentos de la asignatura(2)



La letra con código entra

Las Clases

- Empezamos en punto
- A veces: 5–10 min. de Frikiminutos
  - Gadgets tecnológicos
  - Aplicaciones
  - Cuestiones interesantes
  - ...
- A veces, explicación de los conceptos más importantes y luego realización de ejercicios
- A veces, al revés
- Ejercicios para hacer fuera de clase (y entregar)

Fundamentos de la asignatura

Laboratorio

- Laboratorios Linux de la EIF
- Este curso, a distancia: vía VNC, vía ssh...
- `https://labs.eif.urjc.es`

Software (fundamental) que utilizaremos:

- Ubuntu
- Python, Django
- PyCharm

Evaluación

- Microprácticas diarias (entrega foro/GitLab): 0 a 1
- Miniprácticas preparatorias: 0 a 1
- Proyecto final (obligatorio): 0 a 2.
- Opciones y mejoras práctica final: 0 a 3
- Teoría (obligatorio): 0 a 4.

  Ojo: teoría fuertemente relacionada con prácticas
- Nota final: Suma de notas, moderada por la interpretación del profesor
- Mínimo para aprobar:
  - Aprobado en teoría (2) y proyecto final (1), y
  - 5 puntos de nota final en total

Evaluación (2)

- Evaluación teoría: prueba escrita (quizás en ordenador)
- Microprácticas diarias y miniprácticas incrementales:
  - es muy recomendable hacerlas
- Evaluación proyecto final
  - posibilidad de examen presencial para proyecto final
  - ¡tiene que funcionar en el laboratorio!
  - enunciado mínimo obligatorio supone 1, se llega a 2 sólo con calidad y cuidado en los detalles
- Opciones y mejoras proyecto final:
  - permiten subir mucho la nota
- Evaluación extraordinaria:
  - prueba escrita (si no se aprobó la ordinaria)
  - nuevo proyecto final (si no se aprobó el ordinario)

Proyecto final

Ejemplos del pasado:

- Servicio de búsqueda de hoteles
- Servicio de apoyo a la docencia
- Sitio de intercambio de fotos
- Aplicación web de autoevaluación docente
- Agregador de blogs (canales RSS)
- Agregador de microblogs (Identi.ca, Twitter)

Proyecto final

Este curso...

. . . .

Ejemplos de prácticas finales de otros años

- Niki Montero (2024):
  `https://www.youtube.com/watch?v=coAf8o0gQ00`
- Ainhoa Hernández (2021:
  `https://www.youtube.com/watch?v=7Bd33IS3XBE`
- Noelia López (2019):
  `https://youtube.com/watch?v=VRg37vwUll0`
- Fernando Yustas (2014):
  `https://youtube.com/watch?v=TUUMVEaBzeg`

(puedes buscar en YouTube muchos más ejemplos)

¡Ánimo!

Aquí se enseñan cómo son las cosas
que se usan en el mundo real

Las buenas noticias son...
que no son tan difíciles

Cookies HTTP

Cookies

- Forma de mantener estado en un protocolo sin estado (HTTP)
- Forma de almacenar datos en el lado del cliente
- Dos versiones principales:
  - Versión 0 (Netscape)
  - Version 1 (RFC 2109 / RFC 2965)

Cabeceras HTTP para cookies (version 0)

- Set-Cookie: De servidor a navegador

```
Set-Cookie: statusmessages="deleted"; Path=/;
  Expires=Wed, 31-Dec-97 23:59:59 GMT
```

  - Nombre de la cookie: statusmessages
  - Valor: "deleted"
  - Path: / (todo el sitio del que se recibió)
  - Expira: 31-Dec-97 23:59:59 GMT

- Cookie: De navegador a servidor

```
Cookie: statusmessages="deleted"
```

  - Nombre de la cookie: statusmessages
  - Valor: "deleted"

RFC 2965 (version 1) tiene: "Cookie", "Cookie2" y "Set-Cookie2"

Estructura de Set-Cookie

- Nombre y valor (obligatorios)
  - Uso normal: "nombre=valor"
  - También valor nulo: "nombre="
- Fecha de expiración
  - "Expires=fecha"
  - Si no tiene, cookie no persistente (sólo en memoria)
- Path: camino para el que es válida
  - "Path=/camino"
  - Prefijo de caminos válidos
- Domain: dominio para el que es válida
  - El servidor ha de estar en el dominio indicado
  - Si no se indica, servidor que sirve la cookie
- Seguridad: se necesita SSL para enviar la cookie
  - Campo "Secure"
- Campos separados por ";"

Estructura de Cookie

- Lista de pares nombre - valor
- Cada par corresponde a un "Set-Cookie"
- Se envían las cookies válidas para el dominio y el path de la petición HTTP
- Si no se especificó dominio en "Set-Cookie", el del servidor
- Si no se especificó camino en "Set-Cookie", todo

```
Cookie: user=jgb; last=5; edited=False
```

Límites para las cookies

- Originalmente: 20 cookies del mismo dominio
- La mayoría de los navegadores: 30 o 50 cookies del mismo dominio
- Cada cookie: como mucho 4 Kbytes

Gestión de sesión en HTTP

- Mediante cookies: normalmente, identificador de sesión en la cookie

  ```
  Set-Cookie: session=ab34cd-34fd3a-ef2365
  ```

- Reescritura de urls: se añade identificador a la url

  ```
  http://sitio.com/path;session=ab34cd-34fd3a-ef2365
  ```

- Campos escondidos en formularios HTML

  ```
  <form method="post" action="http://sitio.com/path">
    <input type="hidden" name="session" value="ab34cd-34fd3a
    ...
    <input type="submit">
  </form>
  ```

Referencias

- Persistent Client State HTTP Cookies
  (especificación original de Netscape)
  `http://curl.haxx.se/rfc/cookie_spec.html`
- RFC 2109: HTTP State Management Mechanism
  `http://tools.ietf.org/html/rfc2109`
- RFC 2965: HTTP State Management Mechanism
  `http://tools.ietf.org/html/rfc2965`

REST: Representational State Transfer

REST: El estilo arquitectural de la web

- REST: REpresentational State Transfer
  "Conjunto de recursos web (máquina de estados),
  en el que se realizan acciones seleccionando recursos
  a los que se les aplican operaciones (transiciones de estado),
  obteniendo representaciones de ese estado
  (que recibe el navegador)."
- Estilo arquitectural para sistemas distribuidos
  - Ampliamente extendido en la web estática (ficheros y directorios)
  - Parcialmente extendido en la web programática (aplicaciones y servicios web)

Principios REST

- Cliente (interfaz de usuario) - servidor (lógica de negocio, datos)
- Sin estado: cada petición lleva todo su contexto
- Cacheable: cada respuesta es marcada como cacheable, o no
- Interfaz uniforme (recurso, urls, métodos, representaciones...)
- Sistema por capas: cada componente sólo "ve" la capa que invoca
- Código en el cliente (sobre todo, para interfaz de usuario)

Interfaz: Cada recurso debe tener una URL

Un recurso puede ser cualquier tipo de información que quiere hacerse visible en la web: documentos, imágenes, servicios, gente

```
http://example.com/profiles
http://example.com/profiles/juan
http://example.com/posts/2007/10/11/ventajas_rest
http://example.com/posts?f=2007/10/11&t=2007/11/17
http://example.com/shop/4554/
http://example.com/shop/4554/products
http://example.com/shop/4554/products/15
http://example.com/orders/juan/15
```

Interfaz: Cada recurso debe tener una URL

Dos tipos de recursos:

- Colecciones, como http://example.com/resources/
- Elementos, como http://example.com/resources/142

Los métodos tienen un significado ligeramente diferente se trate de una colección o un elemento.

Interfaz: Los recursos tienen hiperenlaces a otros recursos

Los formatos más usados (XHTML, Atom, ...) ya lo permiten. También aplica a los formatos inventados por nosotros mismos

```
<profiles self="http://example.com/profiles">
 <profile type="moderator"
       self="http://example.com/profiles/123">
  <name>Pepito Pérez</name>
  <company ref="http://example.com/companies/321">
    Compañía XYZ</company>
  </profile>
...
</profiles>
```

Interfaz: Conjunto limitado y estándar de métodos

- GET: Obtener información (quizá de caché)
  - No cambia estado, idempotente
- PUT: Actualizar o crear un recurso conocida su URL
  - Cambia estado, idempotente
- POST: Crear un recurso conociendo la URL de un constructor de recursos
  - Cambia estado, NO idempotente
- DELETE: Borrar un recurso conocida su URL
  - Cambia estado, idempotente

Interfaz: Múltiples representaciones por recurso

Las representaciones muestran el estado actual del recurso en un formato dado

```
GET /profile/1234
Host: example.com
Accept: text/html
```

```
GET /profile/1234
Host: example.com
Accept: text/x-vcard
```

```
GET /profile/1234
Host: example.com
Accept: application/mi-vocabulario-propio+xml
```

Interfaz: Comunicación sin estado

- En REST se mantiene estado, pero sólo en los recursos, no en la aplicación (o sesión).
- Una petición del cliente al servidor debe contener todo el contexto necesario (no puede basarse en información previa asociada a la comunicación).
- No hay datos que permitan identificar sesión en el servidor. El servidor sólo guarda y maneja el estado de los recursos que aloja.
- El cliente puede guardar identificación de sesión (y enviárselo al servidor), por ejemplo en forma de cookies.
- Esto permite escalabilidad (servidores más sencillos). Implica mayor ancho de banda, pero facilita las *cachés*.

REST: Ventajas de REST

- Maximiza la reutilización
  - Todos los recursos tienen identificadores = hacen más grande la Web
  - La visibilidad de los recursos que forman una aplicación/servicio permite usos no previstos
- Minimiza el acoplamiento y permite la evolución
  - La interfaz uniforme esconde detalles de implementación
  - El uso de hipertexto permite que sólo la primera URL usada para acceder acople un sistema a otro
- Elimina condiciones de fallo parcial
  - Fallo del servidor no afecta al cliente
  - El estado siempre puede ser recuperado como un recurso
- Escalado sin límites
  - Los servicios pueden ser replicados en cluster, cacheados y ayudados por sistemas intermediarios
- Unifica los mundos de las aplicaciones web y servicios web. El mismo código puede servir para los dos fines.

Funcionalidades de HTTP no tan conocidas

- Códigos de respuesta estandarizados: los entiende un navegador, los proxies, o nuestras propias aplicaciones
  - Information: 1xx, Success 2xx, Redirection 3xx, Client Error 4xx, Server Error 5xx
- Negociación de contenido: la misma URL puede mandar la representación más adecuada al cliente (HTML, Atom, texto)
- Redirecciones
- Cacheado (incluyendo mecanismos para especificar el periodo de validez y expiraciones)
  - Dos tipos: browser, proxy
  - Cabecera HTTP: Cache-Control

Funcionalidades de HTTP no tan conocidas (2)

- Compresión
- División de la respuesta en partes (Chunking)
- GET condicional (sólo se envía la respuesta si ha habido cambios)
  - Permite ahorrar ancho de banda, procesado en el cliente y (posiblemente) procesado en el servidor
  - Dos mecanismos:
    - ETag y If-None-Match: identificador asociado al estado de un recurso. Ejemplo: hash de la representación en un formato dado
    - Last-Modifed y If-Modified-Since: fecha de última actualización

Ejemplo de interacción HTTP: Petición

```
GET / HTTP/1.1
Host: www.ejemplo.com
User-Agent: Mozilla/5.0 (Windows; U; Windows NT 5.1;
   en-US; rv:1.8.1.3)...
Accept: text/xml,application/xml,application/xhtml+xml,
   text/html;q=0.9...
Accept-Language: en-us,en;q=0.5
Accept-Encoding: gzip,deflate
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7
If-None-Match: "4c083-268-423f1dc5"
If-Modified-Since: "4c083-268-423f1dc5"
Keep-Alive: 300
Connection: keep-alive
Cookie: [...]
Cache-Control: max-age=0
```

Ejemplo de interacción HTTP: Respuesta

```
HTTP/1.x 200 OK
Date: Thu, 17 May 2007 14:06:30 GMT
Server: Microsoft-IIS/6.0
ETag: "4c083-268-423f1dc6"
Last-Modified: Mon, 21 Mar 2005 19:17:26 GMT
Cache-Control: private
Content-Type: text/html; charset=utf-8
Content-Length: 16850
```

Mecanismos de seguridad

- Autenticación:
  - HTTP Basic
  - HTTP Digest
  - OAuth
- Importante: los mecanismos de autenticación de HTTP son extensibles
- Cifrado y firma digital: SSL

Límites actuales de REST en la web

- Obliga a los desarrolladores a pensar diferente
- Carencia de soporte de PUT y DELETE en navegadores y cortado en cortafuegos
- Mecanismos de seguridad limitados (en comparación con WS-Security): está mejorando rápidamente
- La negociación de contenidos no funciona del todo bien en la práctica
- Algunos piden un método más: PATCH
- Poca documentación

Ejemplos RESTful

- En la web
  - Todos los sitios web estáticos
  - Servicios web de sólo lectura
  - Servicios web Google (GData)
  - Aplicaciones web hechas con Ruby Rails 1.2+
- Estándares basados en REST
  - Atom Publishing Protocol
  - WebDAV
  - Amazon S3
  - GData (basado en Atom Publisihg Protocol)
  - OpenSearch (basado en Atom Publisihg Protocol)

Conclusiones

- REST es un estilo arquitectural para aplicaciones distribuidas
- Introduce restricciones que producen propiedades deseables a cambio
  - Estas propiedades han permitido la expansión con éxito de la web
  - Las restricciones también pueden aplicarse a las aplicaciones y servicios web desarrollados para mantener las propiedades
- Los protocolos de la web, HTTP y URL, facilitan el desarrollo de arquitecturas RESTful y permiten explotar sus propiedades
- REST no es el único estilo arquitectural
  - Es posible eliminar restricciones si no nos importa perder propiedades
  - Hay otros estilos arquitecturales con otras propiedades. Ejemplo: RPC

Referencias

- "Chapter 5: Representational State Transfer (REST)", Fielding, Roy Thomas, en "Architectural Styles and the Design of Network-based Software Architectures" (PhD thesis) University of California, Irvine 2000.
- "RESTful Web Services", Leonard Richardson, Sam Ruby, O'Reilly Press
- "Architectural Styles and the Design of Network-based Software Architectures" (Capítulo 5)

  `http://www.ics.uci.edu/~fielding/pubs/dissertation/rest_arch_style.htm`
- "Best Practices for Designing a Pragmatic RESTful API", Vinay Sahni

  `http://www.vinaysahni.com/best-practices-for-a-pragmatic-restful-api`
- Atom Publishing Protocol: `http://tools.ietf.org/html/rfc5023`

Arquitectura modelo-vista-controlador

¿Qué es la arquitectura MVC?

- Patrón de arquitectura (implementación)
- Desacopla tres elementos:
  - datos (estado de la aplicación)
  - representación en la interfaz
  - lógica de aplicación
- Modelo: datos (estado) y su gestión
- Vista: Representación en la interfaz (filtrado, actualización de datos)
- Controlador: lógica de la aplicación y flujo de información

MVC en aplicaciones web (1)

The model is any of the logic or the database or any of the data itself. The view is simply how you lay the data out, how it is displayed. [...]

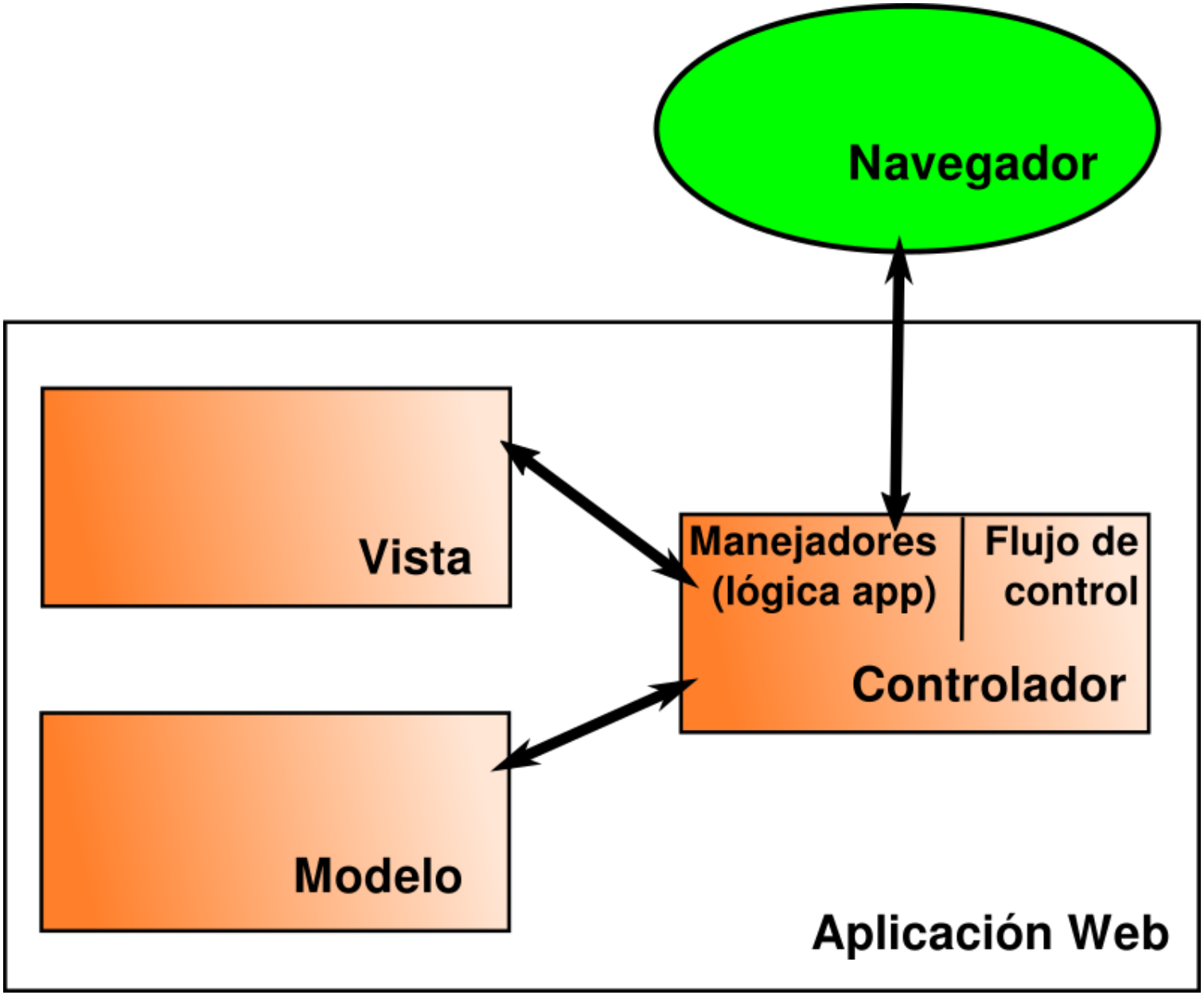
The controller in a web app is a bit more complicated, because it has two parts. The first part is the web server (such as a servlet container) that maps incoming HTTP URL requests to a particular handler for that request. The second part is those handlers themselves, which are in fact often called "controllers".[...]

"The Importance of Model-View Separation", Terence Parr

MVC en aplicaciones web

- Modelo:
  Descripción y gestion de base de datos
- Vista:
  Interfaz de usuario (HTML)
  que presenta el modelo al usuario
- Controlador:
  Recibe indicaciones del usuario,
  indica cambios al modelo,
  elige vista para mostrar resultados

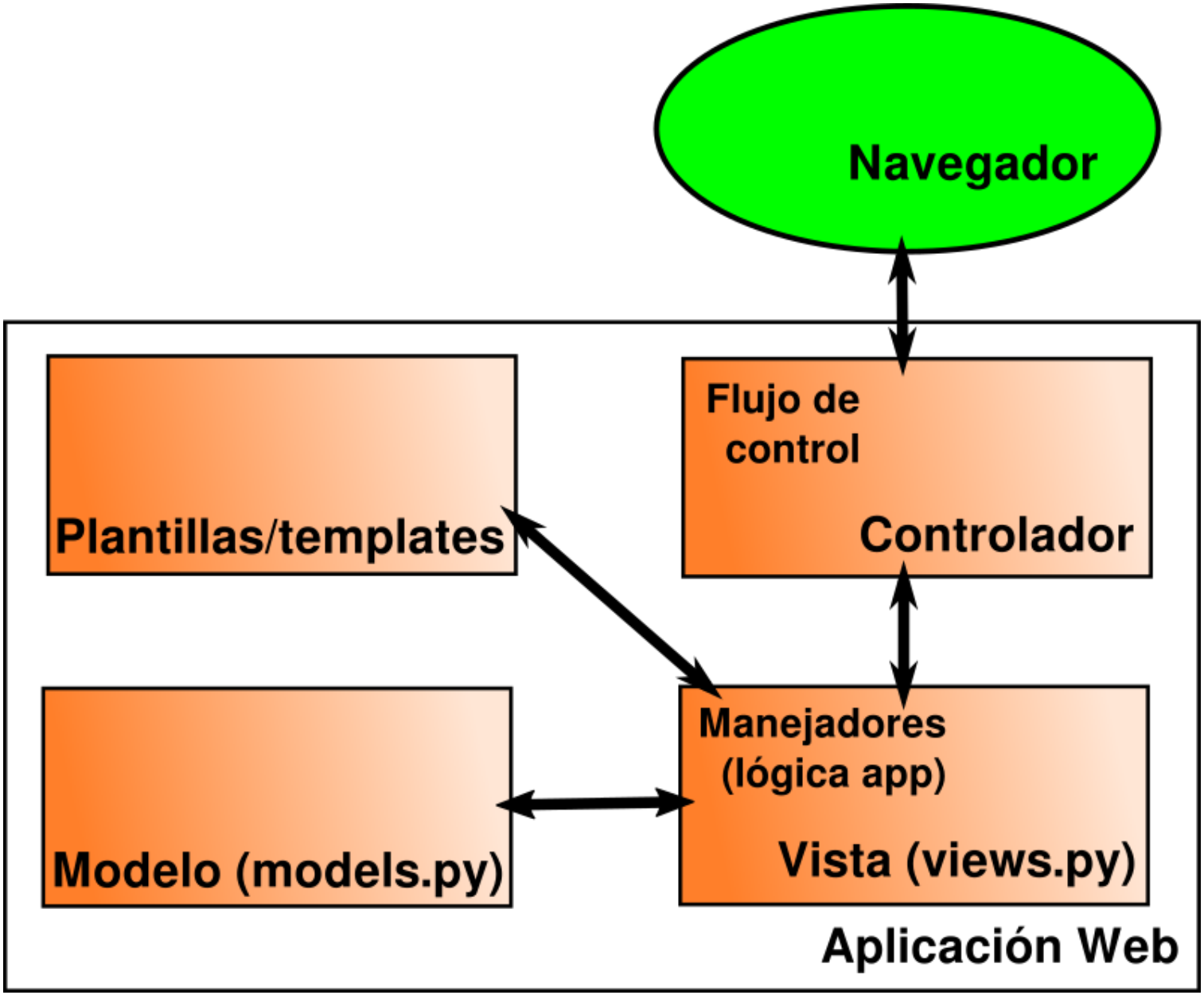
MVC en applicaciones web

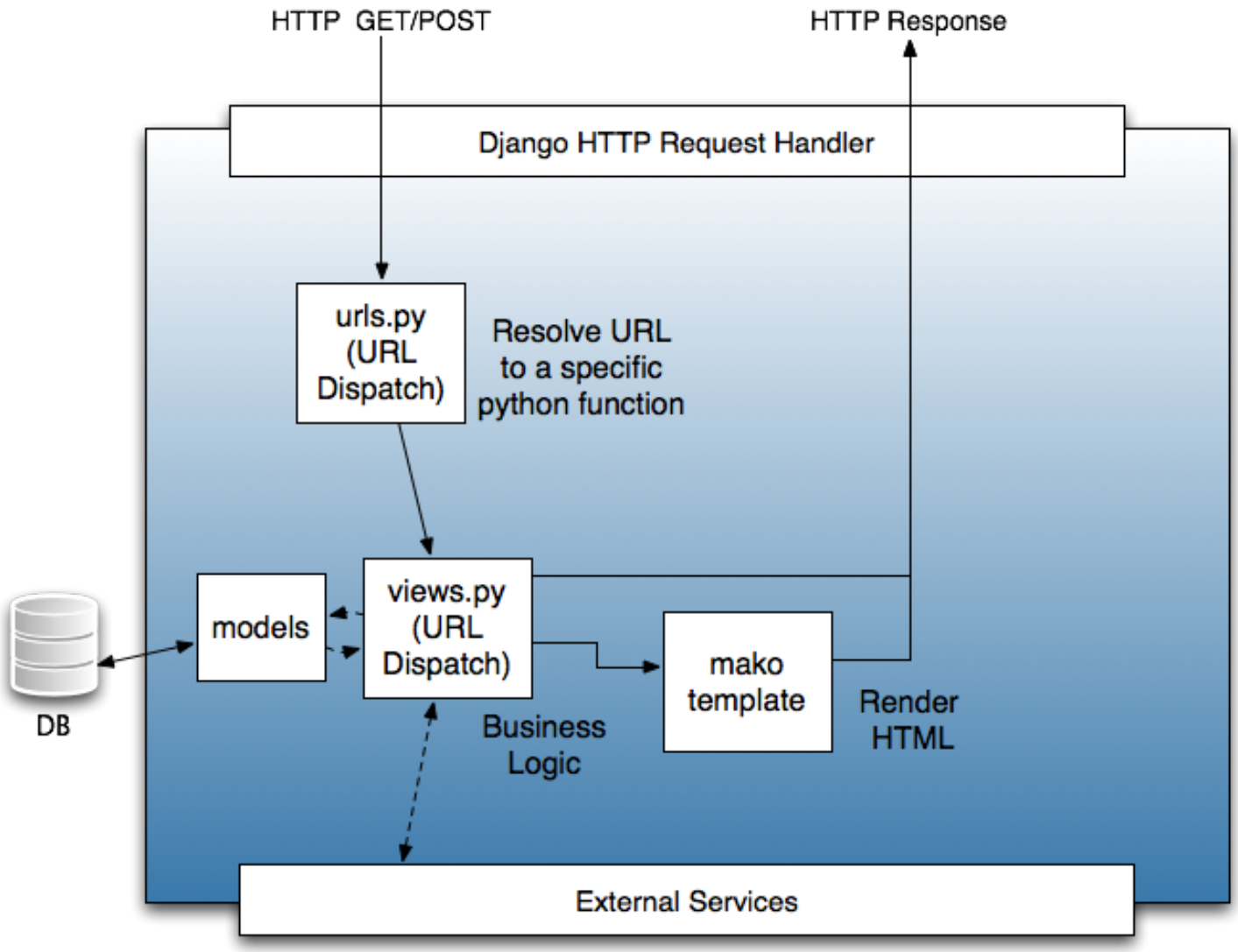


MVC en Django

- Vista: Manejadores (lógica de aplicación)
  funciones en views.py
- Vistas delegan la presentación en plantillas (templates)
- Modelo: Descripción de los datos
  clases en models.py
- Controlador: maquinaria de Django
  incluyendo urls.py
- Django como "plataforma MTV":
  modelo, plantilla (template), vista

Django: MTV


Navegador
Flujo de control
Controlador
Plantillas/templates
Manejadores (lógica app)
Vista (views.py)
Modelo (models.py)
Aplicación Web



Fuente: `http://archive.cloudera.com/cdh4/cdh/4/hue/sdk/sdk.html`

Referencias

- MVC, Xerox PARC 1978-79:
  `http://heim.ifi.uio.no/~trygver/themes/mvc/mvc-index.html`
- "The Importance of Model-View Separation. A Conversation with Terence Parr", by Bill Venners
  `http://www.artima.com/lejava/articles/stringtemplate.html`

Introducción a XML

Introducción

- XML: Extensible Markup Language
- Norma promovida por el W3C (1998)
- Es un dialecto simplificado de SGML (Standarized General Markup Language)
- Pretende ser razonablemente simple
- Se está usando en varios nuevos estándares: MathML, SMIL, (Synchronized Multimedia Integration Language), DocBook/XML, XHTML, RSS, etc.

Características

- Lenguaje de marcado: etiquetas de componentes según su semántica
- Puede especificarse que un documento ha de estar organizado de cierta forma
- Estructura jerárquica
- Sintaxis básica:
  - Marcas:

    ```
    <p> ... </p>
    <p/>
    ```

  - Atributos:

    ```
    <article lang="es">
    ```

Fichero XML sencillo

```
<?xml version="1.0" ?>

<document>
Este es el documento
</document>
```

Ejemplo más complejo

Estructura muy simplificada de DocBook:

```
<?xml version="1.0" ?>
<article>
  <artheader>
    <title>El lenguaje XML</title>
    <author>Tim Ray</author>
  </artheader>
  <sect1>
    <para>...</para>
  </sect1>
  <sect2>
    <para>...</para>
  </sect2>
</article>
```

Definición y validación

- XML ofrece la posibilidad de definir lenguajes (también llamados vocabularios) de etiquetas
- La definición de un lenguaje XML especifica:
  - Elementos del lenguaje
  - Anidamiento de los elementos
  - Atributos para cada elemento
  - Atributos por defecto, atributos obligatorios
- El objetivo es poder validar que un documento XML tiene la información que se espera y con la estructura correcta.

Definición y validación

- Sistemas de definir lenguajes XML
  - DTD (Document Type Definition): Estándar clásico
  - XML Schema: Nuevo estándar. Muy completo y complejo
  - RelaxNG: Alternativa al nuevo estándar. Más sencillo
- Cada documento XML debe tener una referencia a su definición al principio del fichero (si tiene una)

Ejemplos de lenguajes

DocBook XML:

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<!DOCTYPE article PUBLIC
   "-//Norman Walsh//DTD DocBk XML V3.1//EN"
   "/usr/lib/sgml/dtd/docbook-xml/docbookx.dtd">
```

LaEspiral Document:

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<!DOCTYPE article PUBLIC
   "-//laespiral.org//DTD LE-document 1.1//EN"
   "http://.../xml/styles/LE-document-1.1.dtd">
```

Especificación (ejemplo)

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<!DOCTYPE joke [
   <!ELEMENT joke (start, end+)>
   <!ATTLIST joke title CDATA #IMPLIED>
   <!ELEMENT start (#PCDATA)>
   <!ELEMENT end (#PCDATA)>
  ]>

<joke title="Van dos''>
  <start>Van dos por una calle</start>
  <end>y se cae el del medio</end>
  <end>y luego vienen</end>
</joke>
```

Bibliotecas para proceso XML

Tipos de bibliotecas disponibles:

- DOM
  - Document Object Model
  - Lee todo el documento, y permite recorrer el árbol de elementos
  - Permite modificar el documento XML
- SAX
  - Standard API for XML
  - Procesa el documento según está disponible
  - Es my rápido y requiere menos memoria, pero es sólo para lectura
- Otras: variantes de DOM y SAX más sencillas pero no estándar

Usos típicos de XML en aplicaciones web

Gestión de interfaz de usuario en navegador:

- Manipulación de árbol DOM de la página HTML
- Realizada normalmente mediante JavaScript
- Interfaces de usuario mucho más ricas
- Normalmente: combinación con código de aplicación web

Intercambio de datos entre aplicaciones web:

- Fuente para distintos formatos de documento
  (ej: documentos en Docbook servidos en HTML, ePub, etc.)
- Canales (feeds) de una web
- Interfaz para bases de datos (XPath, etc.)
- Web semántica (RDF, RDFa)

DOM de documento HTML en JavaScript (1)

- node.parentNode: nodo padre
- node.childNodes[1]: segundo nodo hijo
- node.firstChild, x.lastChild: primer, último nodo hijo
- document.getElementsByTagName(tag):
  vector con todos los elementos con etiqueta tag
  $< tag > XXX < /tag >$
- document.getElementById(id):
  elemento con identificador id
  $< tagid ='' id'' > XXX < /tag >$

DOM de documento HTML en JavaScript (2)

- x.nodeValue: valor del nodo
- x.innerHTML: HTML que "cuelga" del nodo
- node.setAttribute('attr', val):
  asignar a atributo 'attr' a valor val
- document.createElement(tag)
  crear nodo tag
- document.createTextNode
  crear nodo con texto
- node.appendChild(newnode)
  añade newnode como hijo de node
- node.removeChild(remnode)
  elimina hijo remnode de node

DOM de documento HTML en JavaScript: ejemplo (1)

```
<html>
  <head>
    <script type="text/javascript" charset="utf-8">
      function change()
      {
      document.getElementById('another').innerHTML =
          'Changed!';
      }
    </script>
  </head>
```

DOM de documento HTML en JavaScript: ejemplo (2)

```
<body>
  <h1>Simple DOM demo</h1>
  <p>First sentence</p>
  <p id="sentence">Second sentence, with id "sentence"</p>
  <p id="another">Another sentence, with id "another"</p>
  <p>Text of the sentence with id "sentence":
    <script type="text/javascript" charset="utf-8">
      document.write(
        document.getElementById('sentence').innerHTML
        );
    </script>
  </p>
```

DOM de documento HTML en JavaScript: ejemplo (3)

```
    <p>Text of the second sentence:
      <script type="text/javascript" charset="utf-8">
        document.write(
          document.getElementsByTagName('p')[1].innerHTML
          );
      </script>
    </p>
    <form>
      <input type="button" onclick="change()"
        value="Change" />
    </form>
  </body>
</html>
```

Preparación de documentos

- Los documentos se escriben en un lenguaje XML (por ejemplo Docbook)
- Usando hojas de estilo XSL (Extensible Style Language) pueden obtenerse diversos formatos
- Ventajas
  - Ayuda en la estructuración del texto
  - Simplifica el tratamiento automático
  - Simplifica la validación
  - Se separa el contenido de la presentación

Canales (feeds) de una web

- Introducido por Netscape en 1999, para definir “canales” (feeds)
- Permite a los sitios web publicar una lista de las novedades del sitio
- Esta lista es accedida por aplicaciones lectores de feeds
- Los usuarios pueden subscribirse a los feeds y recibir avisos de los cambios
- Lenguajes XML empleados
  - RSS 0.91 (Rich Site Summary)
  - RSS 0.9 y 1.0 (RDF Site Summary)
  - RSS 2.0 (Really Simple Syndication)
  - Atom 1.0

Ejemplo de canal RSS (1)

```
<?xml version="1.0" encoding="UTF-8"?>
<rss version="2.0">
<channel>
  <title>Menéame: publicadas</title>

  <link>http://meneame.net</link>
  <image>
    <title>Menéame: publicadas</title>
    <link>http://meneame.net</link>
    <url>http://meneame.net/img/mnm/eli-rss.png</url>
  </image>
  <description>Sitio colaborativo de publicación y comunicació
   blogs</description>
```

Ejemplo de canal RSS (2)

```
  <pubDate>Sat, 22 Nov 2008 13:45:02 +0000</pubDate>
  <item>
    <title>¿Qué pasa cuando un familiar te pide que le montes
      PC?</title>
    <link>http://feedproxy.google.com/...-montes-pc</link>
    <pubDate>Sat, 22 Nov 2008 12:55:03 +0000</pubDate>
    <description>&lt;p&gt;Un breve relato que viene a ...
    </description>
  </item>
  <item>...</item>
  <item>...</item>
</channel>
```

Atom

- Creado como evolución de RSS 2.0
- Mejoras que introduce:
  - Identificación del tipo de contenido de los elementos summary y content
  - Incorpora XML namespaces para permitir extender Atom con otros vocabularios XML
- Mucho más apropiado para su interpretación por parte de programas:
  - Esto está provocando su uso habitual en servicios web RESTful

Ejemplo de canal Atom (1)

```
<?xml version="1.0" encoding="utf-8"?>
<feed xmlns="http://www.w3.org/2005/Atom">
    <title>Ejemplo</title>
    <subtitle>Muy interesante</subtitle>
    <link href="http://example.org/"/>
    <updated>2003-12-13T18:30:02Z</updated>
    <author>
       <name>John Doe</name>
       <email>johndoe@example.com</email>
    </author>
    <id>urn:uuid:60a76c80-d399-11d9-b93C-0003939e0af6</id>
```

Ejemplo de canal Atom (2)

```
    <entry>
       <title>Atom-Powered Robots Run Amok</title>
       <link href="http://example.org/2003/12/13/atom03"/>
       <id>urn:uuid:1225c695-cfb8-4ebb-aaaa-80da344efa6a</id>
       <updated>2003-12-13T18:30:02Z</updated>
       <summary type="text">Some text.</summary>
    </entry>
</feed>
```

Tipos de contenido en Atom

```
<content type="text">
   Contenido del feed
</content>
<content type="html">
   Contenido del &lt;em&gt;feed&lt;/em&gt;
</content>
<content type="xhtml" xmlns:xhtml="http://www.w3.org/1999/xht
   <xhtml:div>
     Contenido del <xhtml:em>feed</xhtml:em>
   </xhtml:div>
</content>
```

Web semántica: RDF

- RDF: Resource Description Framework
  - Mecanismo uniforme para especificar la semántica de un recurso web, y su relación con otros.
  - Cada documento RDF es una serie de tripletes.
- Modelo de metadatos basado en declaraciones sobre recursos
- Sujeto (recurso), predicado (relación o aserción entre sujeto y objeto) y objeto (o valor)
- Una colección de declaraciones RDF constituyen un grafo orientado y etiquetado
- Es habitual que el recurso sea un URI, el objeto una cadena Unicode, y el predicado un URI

Otras tecnologías de la familia XML

- XLink: enlaces entre documentos XML
- XSLT: conversión automática de XML a otros formatos
- XPath: localización dentro de un documento ("query" sobre XML)

XML Namespaces

- Permiten agrupar varios vocabularios XML en un único documento
- Resuelven el problema de colisiones entre etiquetas con el mismo nombre

Ejemplo: Problema

Introducir en el canal de una web información geográfica

- Existen dos lenguajes XML distintos para cada información:
  - RSS o Atom: Permiten dar información del canal y sus items
  - geoRSS: Permite especificar longitud y latitud

Ejemplo: Solución

```
<rss xmlns:georss="http://www.georss.org/georss"
     version="2.0">
<channel>
  <item>
    <title>En Málaga colocar publicidad en los parabrisas de
     los coches se multará con hasta 750 euros</title>
     <georss:point>36.7196745 -4.4200359</georss:point>
     ...
```

Uso de Namespaces en la práctica

- El parser empleado debe soportar namespaces: la mayoría lo hacen ya
- Problema: el mecanismo de validación original de XML, los DTDs, no proporciona ninguna solución para validar documentos XML con namespaces
  - Solución: usar XML Schema o RelaxNG

Alternativa a XML: JSON

- JavaScript Object Notation
- Definido en RFC 4627
- Internet media type: application/json.
- Permite representar estructuras de datos
- Específicamente pensado para aplicaciones AJAX
- Es un subconjunto de JavaScript (y de Python)

Ejemplo de JSON

```
{
    "firstName": "John",
    "lastName": "Smith",
    "address": {
        "streetAddress": "21 2nd Street",
        "city": "New York",
        "state": "NY",
        "postalCode": 10021
    },
    "phoneNumbers": [
        "212 732-1234",
        "646 123-4567"
    ]
}
```

Uso en la práctica de JSON

Existen intérpretes para casi cualquier lenguaje:

- JavaScript: La función eval() es capaz de intérpretarlo directamente. Pero mejor usar el módulo JSON
- Python: módulo json
- Java: JSON Tools, xstream, Restlet, ...
- ...

Extensiones:

- JSONT: Equivalente a XSLT para JSON
- JSONP: convierte el contenido en activo al incluir la invocación a una función JavaScript.

Ventajas e Inconvenientes

Ventajas

- Requiere menos caractereres para la misma información (consume menos ancho de banda)
- Es fácil escribir un intérprete rápido (incluso en un navegador)
- Fácil de leer por personas

Inconvenientes

- No tiene mecanismos de definición de lenguajes y validación
- No hay equivalentes a Atom, XSLT, ...

Referencias

- Introducción al XML, por Jaime Villate:
  `http://gsyc.escet.urjc.es/actividades/linuxprog/xml/xml-notas.html`
- Especificación de XML:
  `http://www.w3.org/TR/xml-spec.html`
- Especificación anotada de XML:
  `http://www.xml.com/xml/pub/axml/axmlintro.html`
- Python and XML Processing:
  `http://pyxml.sourceforge.net/topics/`
- SIG for XML Processing in Python:
  `http://www.python.org/sigs/xml-sig/`

Referencias II

- Python/XML HOWTO:
  `http://py-howto.sourceforge.net/xml-howto/`
- XML Linking Language (XLink):
  `http://www.w3.org/TR/xlink/`
- XML Path Language (XPath):
  `http://www.w3.org/TR/xpath`
- XPath Tutorial, por Miloslav Nic y Jiri Jirat:
  `http://www.zvon.org/xxl/XPathTutorial/General/examples.html`
- XML Schema:
  `http://www.w3.org/XML/Schema`

Referencias III

- Using W3C XML Schema, por Eric van der Vlist: `http://www.xml.com/pub/a/2000/11/29/schemas/part1.html`
- XSL Transformations (XSLT): `http://www.w3.org/TR/xslt`
- Página DOM del W3C: `http://www.w3.org/DOM/`
- Página de SAX: `http://www.megginson.com/SAX/`
- Página de RDF: `http://www.w3.org/RDF/`
- RSS 1.0: `http://groups.yahoo.com/group/rss-dev/files/specification.html`

Referencias IV

- XML Namespaces:
  `http://www.w3.org/TR/REC-xml-names/`
- XML Schema:
  `http://www.w3.org/XML/Schema`
- Relax NG:
  `http://relaxng.org`
- Atom (RFC4287):
  `http://tools.ietf.org/html/rfc4287`
- W3C DOM -Introduction:
  `http://www.quirksmode.org/dom/intro.html`
- JSON:
  `http://json.org/`

Ajax y tecnologías relacionadas

Algunos palabros

- DHTML
- SPA
- AJAX
- Web 2.0
- HTML5
- Mashup

DHTML

- Dynamic HTML
- Típicamente: combinación de HTML, CSS (presentación) JavaScript y DOM (manipulación)
- La aplicación corren en el cliente, pero recibe datos, y los envía, del/al servidor
- Maneja datos normalmente usando DOM
- El estado típicamente se mantiene en el servidor
- Problemas debido a las diferencias de las APIs entre navegadores

SPA

- Single page application
- Es un tipo de aplicación DHTML (sin intervención del servidor)
- La aplicación funciona completamente en el navegador
- Maneja los datos de la propia página usando DOM
- Los datos modificados se almacenan localmente

Algunos ejemplos:
`https://onepagelove.com/`
`http://www.awwwards.com/websites/single-page/`

Ajax

- Asynchronous JavaScript and XML
- DHTML más XMLHttpRequest
- Principal cambio para el usuario: las páginas se actualizan, no se recargan completamente
- Normalmente, intercambio de datos con el servidor vía XML (no páginas completas HTML)
- Muchas acciones hechas localmente (cuando no hace falta nueva información)
- Ejemplo de aplicación: Google Mail

Artículo original:

`https://web.archive.org/web/20080702075113/http://www.adaptivepath.com/ideas/essays/archives/000385.php`

Web 2.0

- Evolución del web de colección de sitios a plataforma informática completa.
- Proporciona aplicaciones web a usuarios finales
- Supuestamente sustuirá a muchas aplicaciones de escritorio
- Explotación de "efectos red", por ejemplo con redes sociales (arquitectura de participación)
- Ejemplos: Google AdSense, Flickr, blogs, wikis
- Ejemplos de tecnologías: CSS, XHTML, Ajax, RSS/ATOM

`http://www.oreilly.com/pub/a/web2/archive/what-is-web-20.html`

HTML5

- Última versión de HTML (octubre de 2014)
- Originalmente, trabajo del WHATWG, ahora también del W3C
- Incluye:
  - Primera línea: $<!DOCTYPEhtml>$
  - Nuevo marcado HTML: $<$ *audio* $>$, $<$ *video* $>$,...
  - DOM Scripting (JavaScript)
  - APIs: Canvas 2D, drag-and-drop, off-line storage database, microdata, WebGL, SVG,...
- Los navegadores modernos ya lo soportan

`https://developer.mozilla.org/en-US/docs/Web/Guide/HTML/HTML5`
`http://en.wikipedia.org/wiki/HTML5`
`http://www.html5rocks.com/en/`

Mashup

- Combinación de contenidos (y funcionalidad) de varios sitios en una aplicación web
- Combinación usando APIs de terceros ejecutadas en servidor, feeds (Atom, RSS), JavaScript, etc.
- APIs populares: eBay, Amazon, Google, Windows Live, Yahoo
- Buena integración con otros conceptos de Web 2.0
- Puede haber mashups de mashups...
- Idea general: creación de aplicaciones mediante composición de aplicaciones web

Prácticas: Introducción a Python

Python

Hola Mundo

- Desde la shell (o la consola de Python de *PyCharm*), accede al intérprete de Python **3**:

```
$ python3
>>>
```

- Y ya podemos introducir instrucciones en Python:

```
>>> print("hola mundo")
hola mundo
```

- A partir de ahora obviaremos generalmente los >>> del intérprete.

Más ejemplos

- Podemos usar Python como calculadora
- Verás que Python es sensible mayúsculas
- Los comentarios se indican con #
- En Python hay diferentes tipos de datos

```
saludo = "hola mundo"
print(saludo)    # esto es un comentario
euros = 415
pesetas = euros * 166.386
print(str(euros) + " euros son "+ str(pesetas) + " pesetas")
print(type(saludo))
```

Más ejemplos

- Comprobación de tipos en Python
- El siguiente código es equivalente al de la transparencia anterior
- Simplemente le hemos añadido anotación de tipos
- Esta anotación es *transparente* para el intérprete de Python3

```
saludo: str = "hola mundo"
print(saludo)    # esto es un comentario
euros: int = 415
pesetas: float = euros * 166.386
print(str(euros) + " euros son "+ str(pesetas) + " pesetas")
print(type(saludo))
```

Sangrado y separadores de sentencias

- ¡En Python NO hay llaves ni `begin-end` para encerrar bloques de código!
- Un mayor nivel de sangrado indica que comienza un bloque, y un menor nivel indica que termina un bloque.
- Ejemplo:

```
# Ejemplo de dos funciones en Python
def a_centigrado(faren: float) -> float:
    """Convierte grados farenheit en grados centígrados"""
    return (faren - 32) * (5.0/9)


def a_farenheit(cels: float) -> float:
    """Convierte grados centígrados en grados farenheit"""
    return (cels * 1.8) + 32
```

Condicional

Sentencia `if`:

```
entero: int = 3
if entero:
    print('verdadero')
else:
    print('falso')
```

Nótese como el caracter `:` introduce cada bloque de sentencias. Si hay `:`, entonces la siguiente línea estará indentada.

Cadenas (tipo *str*)

- No existe tipo `char`
- Comilla simple o doble
  `print("hola")` o `print('hola')`
  `print('me dijo "hola"')`
  más legible que `print('me dijo \'hola\'')`
- Puede haber caracteres especiales
  `print("hola\nque tal")`
- El operador + concatena cadenas, y el * las repite un número entero de veces

Listas (tipo *list*)

- Tipo de datos predefinido en Python, va mucho más allá de los arrays
- Es un conjunto **indexado** de elementos, no necesariamente homogéneos
- Sintaxis: Identificador de lista, más valores separados por comas entre corchetes
- Cada elemento se separa del anterior por un caracter ,

```
colores: list = ['rojo','amarillo']
colores.append('verde')
print(colores)
print(colores[2])
print(len(colores))

cosas: list = ['uno', 2, 3.0]
```

Más sobre listas

- El primer elemento tiene índice 0.
- Un índice negativo accede a los elementos empezando por el final de la lista. El último elemento tiene índice -1.
- Pueden referirse **rodajas** (*slices*) de listas escribiendo dos índices entre el caracter :
- La rodaja va desde el **primero, incluido**, al **último, excluido**.
- Si no aparece el primero, se entiende que empieza en el primer elemento (0)
- Si no aparece el segundo, se entiende que termina en el último elemento (incluido).

Ejemplos de listas

```
lista: list = [0, 1, 2, 3, 4]
print(lista)      # [0, 1, 2, 3, 4]
print(lista[1])   # 1
print(lista[0:2]) # [0, 1]
print(lista[3:])  # [3, 4]
print(lista[-1])  # 4
print(lista[:-1]) # [0, 1, 2, 3]
print(lista[:-2)] # [0, 1, 2]
```

¡La misma sintaxis se aplica a las cadenas!

```
cadena: str = "estudiante"
print(cadena[-1])
```

Bucles

Sentencia `for`:

```
>>> amigos: list = ['ana', 'jacinto', 'guillermo']
>>> for invitado in amigos:
...     print(invitado, len(invitado))
...
ana 3
jacinto 7
guillermo 9
```

Diccionarios (tipo *dict*)

- Es un conjunto **desordenado** de elementos
- Cada elemento del diccionario es un par clave-valor.
- Se pueden obtener valores a partir de la clave, pero no al revés.
- Longitud variable
- Elementos heterogéneos
- Hace las veces de los *registros* en otros lenguajes
- Atención: Se declaran con `{}`, se refieren con `[]`

Más sobre diccionarios

```
paises: dict = {'de': 'Alemania', 'fr': 'Francia', 'es': 'España'}
print(paises); print(paises["fr"])

extensiones: dict = {}
extensiones['py'] = 'python'
extensiones['txt'] = 'texto plano'
extensiones['ps'] = 'PostScript'

for pais in paises: # iteramos por el diccionario
   print(pais, paises[pais])

del paises['fr']   # borra esta llave (y su valor)
print(len(paises)) # devuelve el número de elementos en el diccionario
paises.clear()  # vacía el diccionario
```

Sobre los diccionarios

- Asignar valor a una clave existente reemplaza el antiguo
- Una clave de tipo cadena es sensible a mayúsculas/minúsculas
- Pueden añadirse entradas nuevas al diccionario
- Los diccionarios se mantienen desordenados
- Los valores de un diccionario pueden ser de cualquier tipo
- Las claves pueden ser enteros, cadenas y algún otro tipo
- Pueden borrarse un elemento del diccionario con `del`
- Pueden borrarse todos los elementos del diccionario con `clear()`

Características

Parémonos a ver las características de Python. Python es:

- de alto nivel
- interpretado (no compilado)
- orientado a objetos (todo son objetos)
- dinámicamente tipado (frente a estáticamente tipado)
- fuertemente tipado (frente a débilmente tipado)
- sensible a mayúsculas/minúsculas

Utilizando un editor o un IDE (I)

- Usar el intérprete de Python para programar es tedioso
- Es mejor utilizar cualquier editor de texto (p.ej., gedit) o un IDE (como Eclipse)
- Lo que crearemos son ficheros de texto plano.
- Se puede añadir información en la parte superior del fichero para indicar a la shell que es un fichero en Python y con caracteres UTF-8 (y así añadir eñes y tildes, p.ej.).

```
#!/usr/bin/python3
# -*- coding: utf-8 -*-

print("¡Hola Mundo!")
```

Utilizando un editor o un IDE (y II)

- Si lo guardamos como hola.py, se ejecuta desde la línea de comandos como:

  ```
  $ python3 hola.py
  ```

- Podemos darle permisos de ejecución al fichero:

  ```
  $ chmod +x hola.py
  ```

- Y entonces, se ejecuta desde la línea de comandos como:

  ```
  $ ./hola.py
  ```

Ficheros

- `open(fichero: str, modo: str, encoding: str) -> TextIO`
  devuelve un objeto fichero
  modo:
  - `w`: Escritura. Destruye contenido anterior
  - `r`: Lectura. Modo por defecto
  - `r+`: Lectura y Escritura
  - `a`: Append

  codificación (opcional): Por ejemplo, encoding='utf-8'
- `write(cadena: str)` escribe la cadena en el fichero
- `read() -> str` devuelve el contenido del fichero
- `readlines() -> list` devuelve una lista con cada línea del fichero
- `readline() -> str` devuelve la siguiente línea del fichero
- `close()` cierra el fichero

Ejemplos de uso de ficheros

```
#!/usr/bin/python3
from typing import TextIO

fich: TextIO = open('/tmp/prueba','w')
fich.write("lunes\n")
fich.close()

fich: TextIO = open('/tmp/prueba','a')
fich.write("martes\n")
fich.close()

fich: TextIO = open('/etc/hosts','r')
maquinas: list = fich.readlines()
fich.close()

for maquina in maquinas:
    print(maquina)
```

Un programa en Python

```
def sum(sumando1: float, sumando2: float) -> float:
    """Sums two integer/floats

    Returns integer/float."""

    return sumando1 + sumando2

if __name__ == "__main__":
    primero: int = int(input("Please enter an integer/float: "))
    segundo: int = int(input("Please enter another integer/float: "))
    print(sum(primero, segundo))
```

Ejecución:

```
$ python3 suma.py
```

El atributo __name__ de un módulo

Los módulos son objetos, con ciertos atributos predefinidos.
El atributo `__name__`:

- si el módulo es importado (con `import`), contiene el nombre del fichero, sin trayecto ni extensión
- si el módulo es un programa que se ejecuta sólo, contiene el valor `__main__`

Puede escribirse ejecución condicionada a cómo se use el módulo:

```
if __name__ == "__main__":
    ...
```

Importar módulos

- `import nombre-módulo`
  permite acceder a los símbolos del módulo con la sintaxis `nombre-módulo.X`
- `from nombre-módulo import a, b, c`
  incorpora los símbolos a, b, c al espacio de nombres, siendo accesibles directamente (sin cualificarlos con el nombre del módulo)
- `from nombre-módulo import *`
  incorpora los símbolos del módulo al espacio de nombres, siendo accesibles directamente (sin cualificarlos con el nombre del módulo).

Referencias

Si necesitas practicar más:

- Code Academy for Python
  https://www.codecademy.com/learn/learn-python
- Playground and cheatsheet for learning Python
  https://github.com/trekhleb/learn-python
- Awesome Python
  https://github.com/vinta/awesome-python

Ámbito de las variables

- Las variable declaradas fuera de una función son globales

```
#!/usr/bin/python3
numero: int = 5
def f(parametro: int) -> int:
    return parametro + numero
print(f(3))    # 8
```

- Las variable declaradas dentro de una función son locales

```
#!/usr/bin/python3
def f(parametro: int) -> int:
    numero: int = 5
    return parametro + numero
print(f(3))
print(numero)    # ERROR: numero es de ambito local
```

Más sobre ámbito de variables

- Dentro de una función se puede ver una variable global pero no modificar

```
#!/usr/bin/python3
numero: int = 5
def f(parametro: int) -> int:
    numero: int = numero-1  # ERROR: no se puede modificar variable global
    return paramentro + numero

print(f(3))
```

- A menos que se use la sentencia global

```
#!/usr/bin/python3
numero: int = 5
def f(parametro: int) -> int:
    global numero  # permite modificar una variable global
    numero = numero-1 # Nota que no se puede anotar el tipo ahora
    return parametro + numero

print(f(3))    #7
print(numero)  #4
```

Más sobre ámbito de variables

- Un poco más complicado:

```
#!/usr/bin/python3
numero: int = 5
def f(parametro: int) -> int:
    numero: int = 4   # ahora numero es variable local
    return parametro + numero

print(f(3))   # 7
print(numero) # 5
```

Definición de variables

Python es

- fuertemente tipado (frente a débilmente tipado)
- sensible a mayúsculas/minúsculas

En Python la declaración de variables es implícita
(no hay declaración explícita)

- Las variables "nacen" cuando se les asigna un valor
- Las variables "desaparecen" cuando se sale de su ámbito

Sangrado y separadores de sentencias (II)

- Las sentencias se terminan al acabarse la línea (salvo casos especiales donde la sentencia queda "abierta": en mitad de expresiones entre paréntesis, corchetes o llaves).
- El caracter \ se utiliza para extender una sentencia más allá de una linea, en los casos en que no queda "abierta".
- El caracter : se utiliza como separador en sentencias compuestas. Ej.: para separar la definición de una función de su código.
- El caracter ; se utiliza como separador de sentencias escritas en la misma línea.

Tuplas

Tipo predefinido de Python para una lista inmutable.
Se define de la misma manera, pero con los elementos entre paréntesis.
Las tuplas no tienen métodos: no se pueden añadir elementos, ni cambiarlos, ni buscar con `index()`.
Sí puede comprobarse la existencia con el operador `in`.

```
>>> from typing import Tuple
>>> tupla: Tuple[str] = ("a", "b", "mpilgrim", "z", "example")
>>> tupla[0]
'a'
>>> 'a' in tupla
1
>>> tupla[0] =  "b"
Traceback (most recent call last):
  File "<stdin>", line 1, in ?
TypeError: object doesn't support item assignment
```

Utilidad de las tuplas:

- Son más rápidas que las listas
- Pueden ser una clave de un diccionario (no así las listas)
- Se usan en el formateo de cadenas

`tuple(lista)` devuelve una tupla con los elementos de la lista `lista`
`list(tupla)` devuelve una lista con los elementos de la tupla `tupla`

Funciones predefinidas

- `abs()` valor absoluto
- `float()` convierte a float
- `int()` convierte a int
- `str()` convierte a string
- `round()` redondea
- `raw_input()` acepta un valor desde teclado

Operadores

En orden de precedencia decreciente:

```
+x, -x, ~x    Unary operators
x ** y    Power
x * y, x // y, x % y    Multiplication, division, modulo
x + y, x - y    Addition, subtraction
x << y, x >> y    Bit shifting
x & y    Bitwise and
x | y    Bitwise or
x < y, x <= y, x > y, x >= y, x == y, x != y,
x <> y, x is y, x is not y, x in s, x not in s
                        Comparison, identity,
                        sequence membership tests
not x    Logical negation
x and y    Logical and
lambda args: expr            Anonymous function
```

- La declaración implícita de variables como en perl puede provocar resultados desastrosos

```
#!/usr/bin/perl
$sum_elementos= 3 + 4 + 17;
$media=suma_elementos / 3;    # deletreamos mal la variable
print $media;   # y provocamos resultado incorrecto
```

- Pero Python no permite referenciar variables a las que nunca se ha asignado un valor.

```
#!/usr/bin/python3
sum_elementos = 3 + 4 + 17
media = suma_elementos / 3    # deletreamos mal la variable
print(media)   # y el compilador nos avisa con un error
```

adi

Operaciones sobre cadenas

- `join()` devuelve una cadena que engloba a todos los elementos de la lista.
- `split()` devuelve una lista dividiendo una cadena
- `upper()` devuelve la cadena en mayúsculas
- `lower()` devuelve la cadena en minúsculas

Estas funciones de biblioteca, como todas, podemos encontrarlas en la *python library reference* (disponible en el web en muchos formatos)

Más sobre cadenas

```
#!/usr/bin/python3

cadena = "más vale pájaro en mano"
print(cadena.split())
print(cadena.upper())

otra_cadena = "que,cocodrilo,en,tobillo"
print(otra_cadena.split(','))

lista = ['rojo', 'amarillo', 'verde']
print(lista.join())
```

Operaciones sobre diccionarios

- `len(d)` devuelve el número de elementos de `d`
- `d.has_key(k)` devuelve 1 si existe la clave `k` en `d`, 0 en caso contrario
- `k in d` equivale a: `d.has_key(k)`
- `d.items()` devuelve la lista de elementos de `d`
- `d.keys()` devuelve la lista de claves de `d`

Recogiendo datos del usuario con raw_input

```
#!/usr/bin/python3
entero = int(raw_input("Please enter an integer: "))
if entero < 0:
     entero = 0
     print('Negative changed to zero')
elif entero == 0:
     print('Zero')
elif entero == 1:
     print('Single')
else:
     print('More')
```

No existe `switch/case`

Más sobre listas

- `append()` añade un elemento al final de la lista
- `insert()` inserta un elemento en la posición indicada

```
>>> lista
['a', 'b', 'mpilgrim', 'z', 'example']
>>> lista.append("new")
>>> lista
['a', 'b', 'mpilgrim', 'z', 'example', 'new']
>>> lista.insert(2, "new")
>>> lista
['a', 'b', 'new', 'mpilgrim', 'z', 'example', 'new']
```

Más sobre listas

- `index()` busca en la lista un elemento y devuelve el índice de la primera aparición del elemento en la lista. Si no aparece se eleva una excepción.
- El operador `in` devuelve 1 si un elemento aparece en la lista, y 0 en caso contrario.

```
>>> lista
['a', 'b', 'new', 'mpilgrim', 'z', 'example', 'new']
>>> lista.index("example")
5
>>> lista.index("new")
2
>>> lista.index("c")
Traceback (innermost last):
  File "<interactive input>", line 1, in ?
ValueError: list.index(x): x not in list
>>> "c" in lista
0
```

Más sobre listas

- `remove()` elimina la primera aparición de un elemento en la lista. Si no aparece, eleva una excepción.
- `pop()` devuelve el último elemento de la lista, y lo elimina. (Pila)
- `pop(0)` devuelve el primer elemento de la lista, y lo elimina. (Cola)

```
 >>> lista
 ['patatas', 'bravas', 'alioli', 'huevo', 'tortilla', 'chorizo']
>>> lista.remove("alioli")
>>> lista
['patatas', 'bravas', 'huevo', 'tortilla', 'chorizo']
>>> lista.remove("oreja")
Traceback (innermost last):
  File "<interactive input>", line 1, in ?
ValueError: list.remove(x): orija not in list
>>> lista.pop()
'chorizo'
>>> lista
 ['patatas', 'bravas', 'alioli', 'huevo', 'tortilla']
```

Más sobre listas

- El operador + concatena dos listas, devolviendo una nueva lista
- El operador * concatena repetitivamente una lista a sí misma

```
>>> lista = ['patatas', 'bravas', 'alioli']
>>> lista = lista + ['huevo', 'tortilla']
>>> lista
['patatas', 'bravas', 'alioli', 'huevo', 'tortilla']
>>> lista += ['chorizo']
>>> lista
['patatas', 'bravas', 'alioli', 'huevo', 'tortilla', 'chorizo']
>>> lista = [1, 2] * 3
>>> lista
[1, 2, 1, 2, 1, 2]
```

Más sobre listas

- `sort()` ordena una lista. Puede recibir opcionalmente un argumento especificando una función de comparación, lo que enlentece notable su funcionamiento
- `reverse()` invierte las posiciones de los elementos en una lista.

Ninguno de estos métodos devuelve nada, simplemente alteran la lista sobre la que se aplican.

```
>>> li = ['a', 'b', 'new', 'mpilgrim', 'z', 'example', 'new', 'two', 'eleme
>>> li.sort()
>>> li
['a', 'b', 'elements', 'example', 'mpilgrim', 'new', 'new', 'two', 'z']
>>> li.reverse()
>>> li
['z', 'two', 'new', 'new', 'mpilgrim', 'example', 'elements', 'b', 'a']
```

Asignaciones múltiples y rangos

- Pueden hacerse también tuplas de variables:

```
>>> tupla = ('a', 'b', 'e')
>>> (primero, segundo, tercero) = tupla
>>> primero
'a'
```

- La función `range()` permite generar listas al vuelo:

```
>>> range(7)
[0, 1, 2, 3, 4, 5, 6]
>>> (MONDAY, TUESDAY, WEDNESDAY, THURSDAY,
... FRIDAY, SATURDAY, SUNDAY) = range(7)
>>> MONDAY
0
>>> SUNDAY
6
```

Mapeo de listas

- Se puede mapear una lista en otra, aplicando una función a cada elemento de la lista:

```
>>> li = [1, 9, 8, 4]
>>> [elem*2 for elem in li]
[2, 18, 16, 8]
>>> li
[1, 9, 8, 4]
>>> li = [elem*2 for elem in li]
>>> li
[2, 18, 16, 8]
```

Filtrado de listas

- Sintaxis:

```
[expresión-mapeo for elemento in lista-orig if condición-filtrado]
```

- Ejemplos:

```
>>> li = ["a", "mpilgrim", "foo", "b", "c", "b", "d", "d"]
>>> [elem for elem in li if len(elem) > 1]
['mpilgrim', 'foo']
```

Control de flujo

Sentencia `while`:

```
>>> a, b = 0, 1
>>> while b < 1000:
...     print(b, end=' ')
...     a, b = b, a+b
...
1 1 2 3 5 8 13 21 34 55 89 144 233 377 610 987
```

Nótese el efecto de un caracter `,` al final de un `print`
Nótese otro modelo de asignación múltiple

- break sale de un bucle:

```
#!/usr/bin/python3
a=10
while a > 0:
    print(a, end='')
    a=a-1
```

equivale a

```
#!/usr/bin/python3
a=10
while 1:
    print(a, end='')
    if a==1:
        break
    a=a-1
```

Sentencia nula: `pass`

Valor nulo: `None`

Uso de bibliotecas

- Llamada al shell

```
#!/usr/bin/python3
import os
os.system('ls -l')
```

- Argumentos de linea de comandos

```
#!/usr/bin/python3
import sys
print(sys.argv[1:])
```

Las funciones de biblioteca podemos encontrarlas en la *python library reference* (disponible en el web en muchos formatos)

Excepciones

- Un programa sintácticamente correcto puede dar errores de ejecución

```
#!/usr/bin/python3
while 1:
    x=int(raw_input("Introduce un nº"))
    print(x)
```

- Definimos una acción para determinada excepción

```
#!/usr/bin/python3
while 1:
    try:
        x=int(raw_input("Introduce un nº:"))
        print(x)
    except ValueError:
        print("Número incorrecto")
```

- Se puede indicar una acción para cualquier excepción pero es *muy* desaconsejable (enmascara otros errores)
- El programador puede levantar excepciones

```
#!/usr/bin/python3
try:
    x=int(raw_input("Introduce un nº:"))
    print(x)
except :     # para cualquier excepción
    print("Número incorrecto")

raise SystemExit
print("nunca se ejecuta")
```

Objetos en Python

Todo son objetos, en sentido amplio:

- Cualquier objeto puede ser asignado a una variable o pasado como parámetro a una función
- Algunos objetos pueden no tener ni atributos ni métodos
- Algunos objetos pueden no permitir que se herede de ellos

Ejemplos de objetos Python: Strings, listas, funciones, módulos...

Todos los objetos tienen:

- **Identidad**:
  - Nunca cambia.
  - El operador `is` compara la identidad de dos objetos.
  - La función `id()` devuelve una representación de la identidad (actualmente, su dirección de memoria).
- **Tipo**:
  - Nunca cambia.
  - La función `type()` devuelve el tipo de un objeto (que es otro objeto)
- **Valor**:
  - Objetos inmutables: su valor no puede cambiar
  - Objetos mutables: su valor puede cambiar

Contenedores: objetos que contienen referencias a otros objetos (ej.: tuplas, listas, diccionarios).

Cadenas de documentación

- No son obligatorias pero sí muy recomendables (varias herramientas hacen uso de ellas).
- La cadena de documentación de un objeto es su atributo `__doc__`
- En una sola línea para objetos sencillos, en varias para el resto de los casos.
- Entre triples comillas-dobles (incluso si ocupan una línea).
- Si hay varias líneas:
  - La primera línea debe ser una resumen breve del propósito del objeto. Debe empezar con mayúscula y acabar con un punto
  - Una línea en blanco debe separar la primera línea del resto
  - Las siguientes líneas deberían empezar justo debajo de la primera comilla doble de la primera línea

De una sola línea:

```
def kos_root():
    """Return the pathname of the KOS root directory."""
    global _kos_root
    ...
```

De varias:

```
def complex(real=0.0, imag=0.0):
    """Form a complex number.

    Keyword arguments:
    real -- the real part (default 0.0)
    imag -- the imaginary part (default 0.0)

    """
    if imag == 0.0 and real == 0.0: return complex_zero
```

Documentando el código (tipo Javadoc)

- Permite documentar el código -generalmente las funciones- dentro del propio código
- Genera la documentación del código en formatos legibles y navegables (HTML, PDF...)
- Se basa en un lenguaje de marcado simple
- PERO... hay que mantener la documentación al día cuando se cambia el código

Ejemplo

```
def interseccion(m, b):
  """
  Devuelve la interseccion de la curva M{y=m*x+b} con el eje X.
  Se trata del punto en el que la curva cruza el eje X (M{y=0}).

  @type  m: número
  @param m: La pendiente de la curva
  @type  b: número
  @param b: La intersección con el eje Y

  @rtype:   número
  @return:  la interseccíoin con el eje X de la curva M{y=m*x+b}.
  """
  return -b/m
```

Prácticas: Aplicaciones Web

Servidores de aplicaciones y Aplicaciones web

- Los servidores de aplicaciones son programas informáticos que proporcionan un entorno de ejecución para aplicaciones web. Estos servidores están diseñados para manejar las solicitudes de los clientes, ejecutar el código de la aplicación y devolver las respuestas adecuadas.
- Las aplicaciones web suelen seguir una estructura general que incluye varios componentes clave que trabajan juntos para proporcionar funcionalidades y servicios a los usuarios a través de Internet.

Servidores de aplicaciones

- **Entorno de ejecución:** Los servidores de aplicaciones proporcionan un entorno de ejecución donde las aplicaciones web pueden ser desplegadas y ejecutadas. Esto incluye un conjunto de recursos como memoria, CPU, bibliotecas y otros componentes necesarios para ejecutar las aplicaciones.
- **Gestión de solicitudes y respuestas:** Los servidores de aplicaciones manejan las solicitudes HTTP entrantes de los clientes y generan las respuestas correspondientes. Esto implica interpretar la solicitud del cliente, ejecutar el código de la aplicación y generar una respuesta HTTP para enviar de vuelta al cliente.

Servidores de aplicaciones

- **Soporte para tecnologías específicas:** Los servidores de aplicaciones pueden proporcionar soporte integrado para tecnologías específicas, como Java EE, .NET, Node.js, Python, entre otras. Esto significa que pueden ejecutar código escrito en diferentes lenguajes de programación y utilizar diferentes frameworks y librerías.
- **Gestión de sesiones y estados:** Muchos servidores de aplicaciones ofrecen capacidades para gestionar sesiones de usuario y estados de aplicaciones. Esto permite mantener la coherencia y la persistencia de los datos a través de múltiples solicitudes del mismo usuario.

Servidores de aplicaciones

- **Escalabilidad y tolerancia a fallos:** Los servidores de aplicaciones suelen ser escalables y pueden distribuir la carga de trabajo entre varios servidores para manejar grandes volúmenes de tráfico. Además, pueden proporcionar mecanismos para la tolerancia a fallos, como la recuperación automática y la redundancia de los servidores.

Aplicaciones web Frontend (Cliente)

- **Interfaz de usuario (UI):** El frontend de una aplicación web incluye la interfaz de usuario con la que interactúan los usuarios. Esto puede ser una combinación de HTML, CSS y JavaScript que define la apariencia y el comportamiento de la aplicación en el navegador web del usuario.
- **Lógica del cliente:** También puede haber lógica del cliente escrita en JavaScript que se ejecuta en el navegador del usuario. Esta lógica se utiliza para manejar eventos de usuario, validar datos y realizar acciones interactivas sin necesidad de comunicarse con el servidor.

Aplicaciones web Backend (Servidor)

- **Servidor web:** El backend de una aplicación web consta de un servidor web que recibe y procesa las solicitudes HTTP de los clientes. Este servidor puede ser implementado utilizando diferentes tecnologías como Node.js, Python (con frameworks como Flask o Django), Ruby on Rails, Java (con frameworks como Spring Boot) o PHP, entre otros.
- **Lógica de la aplicación:** La lógica de la aplicación se encarga de manejar las solicitudes del cliente, procesar datos, realizar operaciones en la base de datos y generar respuestas adecuadas. Esta lógica puede estar dividida en diferentes componentes como controladores, servicios, modelos, entre otros, dependiendo de la arquitectura de la aplicación.

Aplicaciones web Backend II (Servidor)

- **Base de datos:** La mayoría de las aplicaciones web interactúan con una base de datos para almacenar y recuperar datos. Esto puede ser cualquier tipo de base de datos como SQL (por ejemplo, MySQL, PostgreSQL) o NoSQL (por ejemplo, MongoDB, Firebase).

Aplicaciones web Comunicación entre el frontend y el backend

- **API (Interfaz de programación de aplicaciones):** La comunicación entre el frontend y el backend se realiza a través de una API que define cómo interactuar con la aplicación y qué datos se pueden solicitar o enviar. Esto puede ser una API RESTful que utiliza solicitudes HTTP estándar (GET, POST, PUT, DELETE) para operaciones CRUD (Crear, Leer, Actualizar, Eliminar) en recursos de la aplicación.
- **Formato de intercambio de datos:** Los datos intercambiados entre el frontend y el backend suelen estar en formato JSON (JavaScript Object Notation) o XML (eXtensible Markup Language) para facilitar el procesamiento y la interoperabilidad entre diferentes sistemas.

Aplicaciones web Despliegue y hospedaje

- **Servidores y entornos de ejecución:** Una aplicación web se despliega en servidores que están configurados para ejecutar el software necesario para su funcionamiento. Estos servidores pueden estar ubicados en la nube (por ejemplo, AWS, Azure, Google Cloud) o en infraestructuras locales.
- **Nombres de dominio y DNS:** Las aplicaciones web suelen tener un nombre de dominio asociado que permite a los usuarios acceder a ellas a través de Internet. Este nombre de dominio se asocia con una dirección IP mediante registros DNS (Sistema de Nombres de Dominio).

Gestión del path en aplicaciones

- **Enrutamiento de solicitudes:** El path en una URL especifica la ubicación de un recurso dentro de la aplicación. Un servidor de aplicaciones debe ser capaz de interpretar el path de una solicitud entrante y enrutarla correctamente al recurso correspondiente. Por ejemplo, si un cliente solicita /productos.^{en} la URL, el servidor debe dirigir la solicitud al controlador o función encargada de manejar la lógica relacionada con los productos.
- **Organización de la aplicación:** Gestionar el path de manera adecuada permite organizar la aplicación de forma estructurada y coherente. Al utilizar una estructura de paths lógica y consistente, es más fácil para los desarrolladores y administradores de sistemas comprender y mantener la aplicación. Esto facilita la adición de nuevas funcionalidades, la depuración de errores y la realización de tareas de mantenimiento.

Gestión del path en aplicaciones

- **Seguridad:** Gestionar el path correctamente también es importante desde el punto de vista de la seguridad. Un servidor de aplicaciones puede implementar controles de acceso basados en el path para restringir el acceso a ciertos recursos o funcionalidades a usuarios autorizados. Por ejemplo, se pueden aplicar reglas de autorización basadas en el path para garantizar que solo los usuarios autenticados puedan acceder a ciertas partes de la aplicación.

Gestión del path en aplicaciones

- **SEO (Optimización para motores de búsqueda):** La estructura del path en una URL también puede afectar el SEO de una aplicación web. Los motores de búsqueda suelen dar preferencia a las URLs que son descriptivas y significativas. Por lo tanto, gestionar el path de manera que refleje la estructura y el contenido de la aplicación puede ayudar a mejorar su visibilidad en los resultados de búsqueda.

Prácticas: Introducción a Django

Enfoques comunes de desarrollo web

- Frameworks de desarrollo web
  - PHP, JavaEE, Python+HttpServer...
- Entornos de desarrollo web completos
  - Django (Python), `http://djangoproject.org`
  - Ruby on Rails (Ruby), `http://rubyonrails.org/`
  - CakePHP (PHP), `http://cakephp.org/`
  - Grails (Groovy, sovre JVM), `http://grails.org/`
  - RIFE (Java), `http://rifers.org/`
- Plataformas extensibles
  - CMS: Joomla, Drupal...
  - Portal: Plone/Zope, Liferay Portal...
  - Plataformas de propósito específico: Moodle, Wordpress...

¿Qué es Django?

- Entorno integrado de desarrollo de aplicaciones web
- Herramientas para gestionar la aplicación
- Framework (armazón) para presentación de la aplicación
- Acceso a base de datos (correspondencia objeto-relacional)
- Seguridad (XSS, SQL Injection, ...)
- Componentes listos para usar (gestión de usuarios, sesiones, interfaz administración,...)
- *Caché*, internacionalización, plantillas, etc.

`http://docs.djangoproject.com/`

Otras plataformas Python similares a Django

Flask: `https://flask.palletsprojects.com/`

- Aproximación minimalista, centrada en la simplicidad
- Con extensiones, funcionalidad similar a Django

FastAPI: `https://fastapi.tiangolo.com/`:

- Especialmente orientada a APIs HTTP
- Se apoya en el tipado de Python

Django: conceptos principales

- Objetivo principal: desarrollo muy rápido
  - Entorno integrado y completo
  - Cambios en caliente
  - Descripciones de error muy descriptivas
  - Convenciones preferible a configuración
  - Evitar duplicación a toda costa (DRY, don't repeat yourself)
- Desarrollo dirigido por el modelo
  - Se comienza por el diseño del modelo de datos

Preparativos

- Usaremos una versión de Django 5.1.7, instalándola vía pip3.
- Disponible para Linux, *BSD, Windows, MacOS, etc.
- Creación de entorno virtual:

  ```
  % python3 -m venv django-venv
  ```
- Activación del entorno virtual:

  ```
  % source django-venv/bin/activate
  ```
- Instalación de Django en el entorno virtual:

  ```
  % pip3 install django==5.1.7
  ```
- Comprobación (debe responder 5.1.7):

  ```
  % django-admin --version
  ```

Armazón para proyecto y aplicación

- Creación (primero proyecto, luego aplicación)

```
$ cd dir-practica
$ django-admin startproject myproject
$ cd myproject
$ python3 manage.py startapp myfirstapp
```

- Más opciones de manage.py

```
$ python3 manage.py --help
```

- Ejecución de la aplicación (`http://localhost:8000`)

```
$ python3 manage.py runserver
```

Ficheros creados para un proyecto

- Raíz:
  - `manage.py`: herramienta para gestionar el proyecto
- Proyecto : `myproject/`:
  - `__init__.py`: fichero vacío, directorio debe ser considerado un paquete Python
  - `settings.py`: configuración del proyecto
  - `urls.py`: URLs del proyecto
  - `wsgi.py`: fichero para servir Django mediante WSGI (p.ej. con Apache)
  - `asgi.py`: fichero para servir Django mediante ASGI (servidor asíncrono)

Ficheros creados para una aplicación

- Raíz:
  - Aplicación (`myfirstapp/`):
    - `__init__.py`: fichero vacío, directorio debe ser considerado un paquete Python
    - `views.py`: vistas (código invocado para cada recurso)
    - `models.py`: definición de las clases del modelo de datos
    - `admin.py`: registro de los modelos que se pueden gestionar vía web
    - `apps.py`: configuración de la app
    - `tests.py`: fichero para la implementación de tests
    - `migrations/`: versiones (y *migraciones*) de los modelos

Generalmente, también se crea un `urls.py`, con las URLs de la aplicación.

Fichero settings.py

- Fichero de configuración, en Python
- Configuración de la base de datos (usaremos SQLite3)

```
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.sqlite3',
        'NAME': os.path.join(BASE_DIR, 'db.sqlite3'),
    }
}
```

- Aplicaciones instaladas

```
INSTALLED_APPS = [
    ...,
]
```

- Otros: zona horaria, codificación, directorio de plantillas...

Declaración de URLs (proyecto)

- En el fichero urls.py (del proyecto): `myproject/urls.py`
- Ejemplo:

```
from django.contrib import admin
from django.urls import include, path

urlpatterns = [
    path('myfirstapp/', include('myfirstapp.urls')),
    path('admin/', admin.site.urls),
]
```

- `include()` permite añadir otros `urls.py` de aplicaciones

Declaración de URLs (aplicación)

En el fichero urls.py (de la aplicación): `myfirstapp/urls.py`

```
from django.urls import path
from . import views

urlpatterns = [
    path('', views.index, name='index'),
    path('hello', views.say_hello, name='hello'),
    path('bye/<name>', views.say_bye_to, name='bye'),
    path('number/<int:number>', views.say_number, name='numbe
]
```

Declaración de URLs: La función path

`path(route, view, kwargs=None, name=None)`

- `route`: patrón de la URL.
  - No tiene en cuenta si la petición es GET o POST, o el dominio
  - Se ejecuta la primera que se encuentra (i.e., el orden importa)
- `view`: función *vista* a la que se llama, pasándole:
  - Como primer parámetro un objeto HttpRequest con la petición
  - Otros posibles parámetros como argumentos
- `kwargs`: diccionario con parámetros adicionales a la vista.
- `name`: nombre de la URL (muy útil para plantillas).

Views

- Código invocado para una URL o conjunto de URLs
- Debe ser un método (o un objeto)
- Los métodos se definen en el fichero `myfirstapp/views.py`
- Ejemplo:

```
from django.http import HttpResponse

def index(request):
     return HttpResponse('<h1>Welcome to my App!</h1>')
def say_hello(request):
     return HttpResponse('Hello!')
def say_bye_to(request, name):
     return HttpResponse('Bye %s'%name)
def say_number(request, number=0):
     return HttpResponse('Number: %s'%number)
```

Gestión de datos persistentes

- Django hace corresponder un objeto Python con cada tabla
- Cada aplicación tiene su models.py
  - Una clase por cada entidad (tabla) del modelo
  - Un campo por cada dato (columna) de la entidad
  - __str__() da una representación del objeto (p.ej., un campo)
  - Ejemplo:

```
from django.db import models

class MyFirstAppData(models.Model):
    name = models.CharField(max_length=200)
    birthday = models.DateTimeField()

    def __str__(self):
        return self.name
```

- Creación de tablas

```
$ python manage.py makemigrations
$ python manage.py migrate
```

Definición del modelo

- Tipos de campos:
  - CharField(maxlength)
  - TextField()
  - IntegerField()
  - DateField()
  - BooleanField()
- Relaciones:
  - ForeignKey(othermodel)
  - ManyToManyField('self', symmetrical=False)

`http://docs.djangoproject.com/en/dev/ref/models/fields/`

Migraciones de datos

Cuando se modifica el modelo, puede existir una inconsistencia con los datos en la base de datos. ¡Peligro: se pueden perder datos!

- El proceso de migraciones se encarga de la propagación de modificaciones en el modelo (p.ej., añadir/borrar un campo)
- `makemigrations`: crea nuevas migraciones, basándose en los cambios a los modelos.
- `migrate`: aplica la migración, además de listar el estado.
- Los archivos para las migraciones se encuentran en el directorio `migrations` de las aplicaciones. Se crean automáticamente con `makemigrations`, aunque (luego) se pueden modificar a mano.
- `createsuperuser`: crea un usuario superusuario para poder entrar en el interfaz de admin admin

Admin site

Versión simple:

- INSTALLED_APPS (en settings.py) ha de incluir:
  - django.contrib.admin
  - django.contrib.sessions (dependencia del anterior)
  - ...y lo necesario para usuarios
- Hay que crear las tablas pertinentes (manage.py migrate)
- Enganche en urls.py

```
from django.contrib import admin
urlpatterns = [
    url('^admin/', include(admin.site.urls)),
]
```

Admin site (2)

Ahora, proporcionemos interfaz para nuestra tabla Pages:

- Crea en el directorio de la aplicación Django de gestión de contenidos el fichero admin.py
- Registra en él los modelos a manejar:

```
from django.contrib import admin
from .models import Pages

admin.site.register(Pages)
```

- Prueba que ahora puedes manejar esta tabla desde el sitio de administración

Consultas a la base de datos

- Métodos para realizar consultas a la base de datos
- Acceso a entradas de la base de datos mediante el objeto 'objects' (ej. MyFirstAppData.objects)
- Métodos:
  - MyFirstAppData.save()
  - MyFirstAppData.objects.all()
  - MyFirstAppData.objects.filter(campo=valor)
  - MyFirstAppData.objects.get(campo=valor)
    Excepción si no lo encuentra

Modelos: relación muchos a uno (ForeignKey)

```
class Manufacturer(models.Model):
    # ...
class Car(models.Model):
    manufacturer = models.ForeignKey(Manufacturer)
    # ...

# Creating
m = Manufacturer(name='Seat')
c = Car(name='Toledo')
m.save(); c.save()
# Relationship
c.manufacturer = m
# Obtaining
c.manufacturer
c.manufacturer.id
```

Acceso al modelo desde las vistas

- Las vistas pueden usarse para leer y modificar al modelo

```
from django.http import HttpResponse,HttpResponseNotFound
from content.models import Pages

def show_content(request, resource):
    try:
        record = Pages.objects.get(name=resource)
        return HttpResponse(record.page)
    except Pages.DoesNotExist:
        return HttpResponseNotFound(
            'Page not found: /%s.' % resource
            )
```

Seguridad

- Django viene con alguna protección de seguridad por defecto
- Por ejemplo, CSRF: *Cross-site request forgery* o falsificación de petición en sitios cruzados
- Si el POST/PUT no incluye un csrf_token creado con anterioridad por el servidor, da un error de seguridad
- Se evita con el @csrf_exempt para un método concreto en views.py

```
from django.views.decorators.csrf import csrf_exempt
@csrf_exempt
def vista(request):
    if request.method == "POST":
        return HttpResponse("Has enviado " + request.body)
```

Usuarios

- INSTALLED_APPS (en settings.py) ha de incluir:
  - django.contrib.auth
  - django.contrib.contenttypes
- Hay que crear las tablas pertinentes (manage.py migrate)

Acceso a información de usuario y logout

- Accedemos a información del objeto User, que tenemos en HTTPRequest
- En views.py:

```
def show_content(request, resource):
  if request.user.is_authenticated:
    logged = 'Logged in as ' + request.user.username
  else:
    logged = 'Not logged in.'
```

- Para logout, utilizamos función. En views.py:

```
from django.contrib.auth import logout

def logout_view(request):
    logout(request)
    return redirect("/")
```

Login

- Utilizamos view predefinida (entiende GET y POST)
- En urls.py:

```
from django.contrib.auth.views import login
urlpatterns = [
    url(r'^login', login),
]
```

- Necesita una plantilla registration/login.html:
  - En settings.py:

```
TEMPLATES = [
  {
    'DIRS': ['templates'], # para todas las apps del proyec
  },
]
```

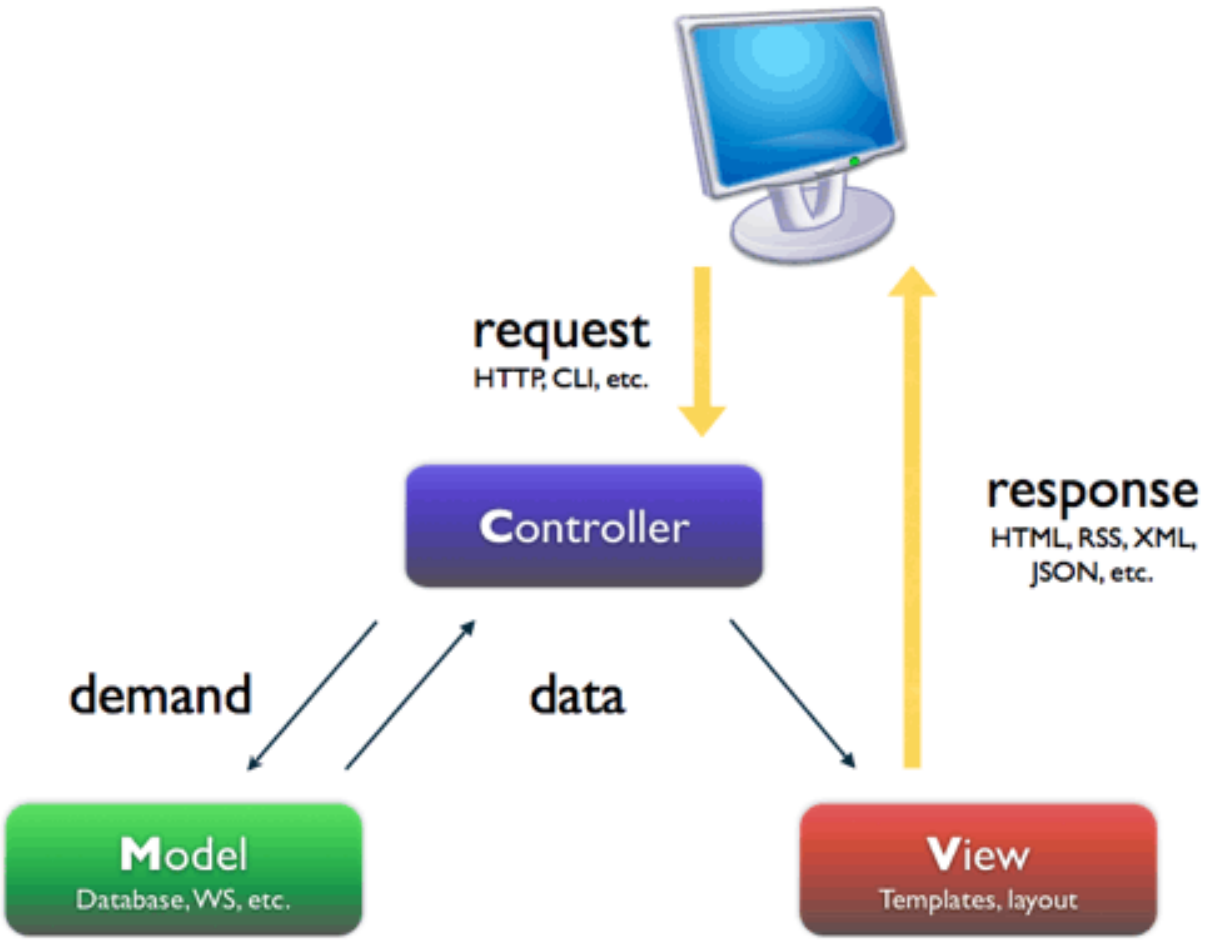
  - Creación de templates/registration/login.html

Info detallada: "User authentication in Django"

templates/registration/login.html

```
<html><body>
<form method="post" action="/login">
{% csrf_token %}
<table>
  <tr><td>Username</td>
      <td>{{ form.username }}</td></tr>
  <tr><td>Password</td>
      <td>{{ form.password }}</td></tr>
</table>
<input type="submit" value="login" />
</form>
</body></html>
```

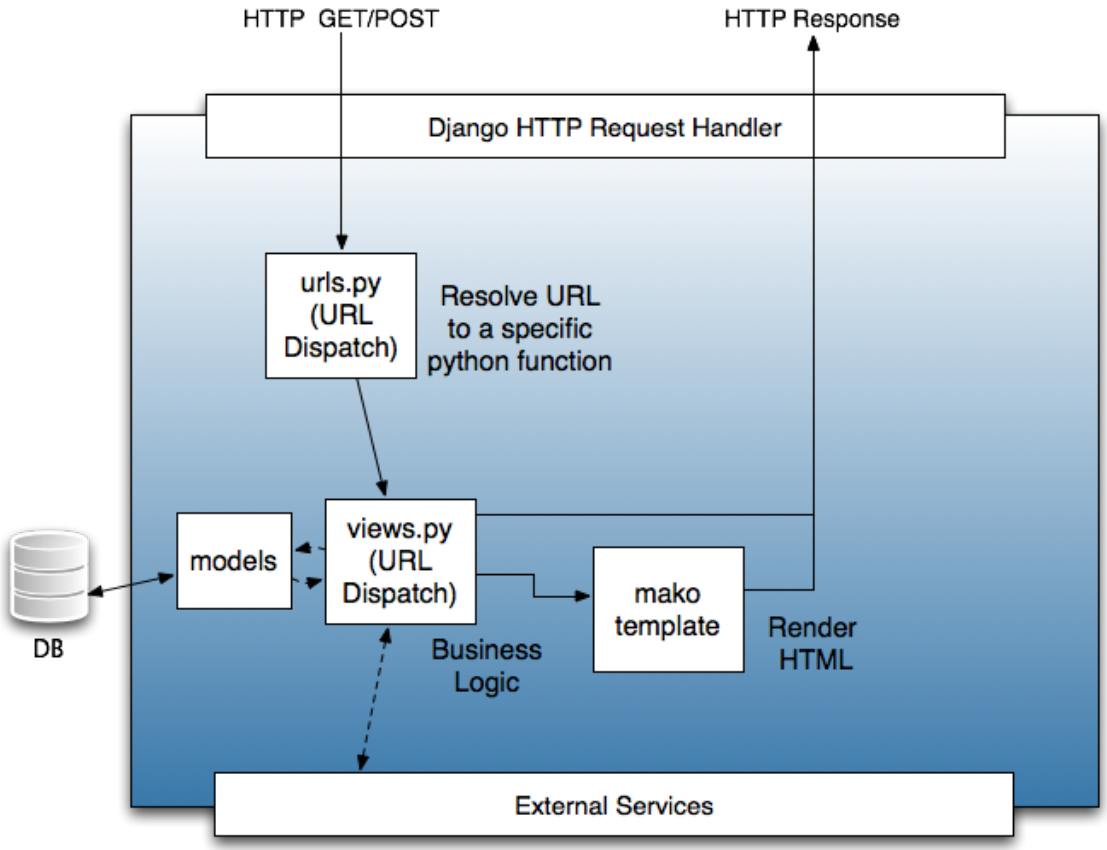
Model View Controller (el tradicional)



Fuente:

http://djangoexamples.blogspot.com.es/2013/05/about-django-mvc.html

Model View Template (el de Django)



Fuente: `http://archive.cloudera.com/cdh4/cdh/4/hue/sdk/sdk.html`

Plantillas (templates)

- Ficheros de texto que pueden generar cualquier formato basado en texto (HTML, XML, CSV, etc.)
- Contienen:
  - Texto (que queda igual)
  - Variables (reemplazadas por su valor cuando se evalúan)
  - Filtros (modifican variables cuando se evalúan)
  - Etiquetas (controlan la lógica de la evaluación de la plantilla)
  - Comentarios {# Comentario #}
- Pueden extender (heredar de) otras plantillas
- Se colocan en los directorios de plantillas (TEMPLATE_DIRS en settings.py)

Plantillas: variables y filtros

- Variables:

```
{{ variable }}
```

- Filtros:

```
{{ variable|filtro|otrofiltro }}
```

- Filtro con argumentos:

```
{{ variable|filtro:30 }}
```

- Ejemplos de filtros:

```
{{ value|default:"nothing" }}
{{ value|length }}
{{ text|striptags }}
{{ text|truncatewords:30 }}
{{ text|escape|linebreaks }}
{{ list|join:", " }}.
```

Ejemplo de plantilla

```
<html>
  <body>
    <div id="title">
      {{ title }}
    </div>

    <div id="content">
      {{ content }}
    </div>
  </body>
</html>
```

Plantillas: uso en vistas

Directorios con plantillas:

- directorio templates en directorio de la app
- lista DIRS en variable TEMPLATES en settings.py

```
from django.shortcuts import render

def view (request):
    ...
    return(render(request, 'template.html',
                  {'title': title,
                   'content': content}))
```

Plantillas: uso en vistas (2)

```
from django.template import loader

def view (request):
    ...
    template = loader.get_template('template.html')
    html = template.render({'title': title,
                            'content': content},
                           request)
    return(HttpResponse(html, status=status))
```

Plantillas (avanzado): etiquetas

- ifequal, ifnotequal

```
{% ifequal athlete.name coach.name %}
    ...
{% endifequal %}
{% ifnotequal athlete.name "Joe" %}
    ...
{% endifnotequal %}
```

- block: Indica bloques a redefinir en plantillas que hereden
- extends: Hereda de plantilla madre (sólo hace falta redefinir *blocks* que se quieren modificar)

Herencia en Plantillas: Ej. plantilla "madre" (base.html)

```
<body>
  <div id="sidebar">
  {% block sidebar %}
    <ul>
     <li><a href="/">Home</a></li>
     <li><a href="/blog/">Blog</a></li>
    </ul>
  {% endblock %}
  </div>

  <div id="content">
     {% block content %}{% endblock %}
  </div>
</body>
</html>
```

Ejemplo de plantilla que hereda (plantilla.html)

```
{% extends "base.html" %}
{% block title %}
  {{ title }}
{% endblock %}

{% block content %}
  <h1>{{ title }}</h1>
  {% for story in story_list %}
  <h2>
    <a href="{{ story.get_absolute_url }}">
      {{ story.headline|upper }}
    </a>
  </h2>
  <p>{{ story.tease|truncatewords:"100" }}</p>
  {% endfor %}
{% endblock %}
```

Modelos: relación muchos a muchos (ManyToManyField)

```
class Topping(models.Model):
    name = models.CharField(max_length=32)
class Pizza(models.Model):
    name = models.CharField(max_length=32)
    toppings = models.ManyToManyField(Topping)
```

Modelos: relación muchos a muchos (ManyToManyField) (2)

```
pb = Pizza(name='Barbecue')
pq = Pizza(name='4 Cheese')
b = Topping(name='Barbecue sauce')
m = Topping(name='Mozzarela')
pb.save(); pp.save(); b.save(); m.save()
pb.toppings.add(b, m)
pq.toppings.add(m)
pq.toppings.create(name='Rochefort')
m.pizza_set.all()
pb.toppings.all()
Pizza.objects.filter(toppings__name='Mozzarela')
```

Ficheros estáticos con Django

- Los ficheros estáticos no se deberían servir con Django...
- (lo hace mucho mejor un servidor web como Apache o Cherokee)
- ...pero se pueden servir
- django.views.static.serve()

```
from django.views.static import serve

urlpatterns = [
  url(r'^css/(.*)$', serve, {'document_root': 'sfiles'}),
]
```

(donde sfiles es un subdirectorio que cuelga del proyecto)

Plantillas: uso en urls.py

```
from django.conf.urls.defaults import *
from django.views.generic.simple import direct_to_template

urlpatterns = [
    url(r'^about$', direct_to_template, {
        'template': 'about.html'
    }),
]
```

Otras acciones de gestión del proyecto

- Ejecución en el contexto de Python con acceso al código de la aplicación

  ```
  % python manage.py shell
  ```

- Validacion de modelos de datos

  ```
  % python manage.py validate
  ```

- Exportación de datos de la base de datos

  ```
  % python manage.py dumpdata
  ```

- Importación de datos en la base de datos

  ```
  % python manage.py loaddata
  ```

Plantillas (avanzado): etiquetas

- for

```
{% for athlete in athlete_list %}
    <li>{{ athlete.name }}</li>
{% endfor %}
```

- if

```
{% if athlete_list %}
    Number of athletes: {{ athlete_list|length }}
{% else %}
    No athletes.
{% endif %}
```

La shell de Django

Acceso a la API de los objetos de nuestro proyecto

```
% python manage.py shell
>>> from myproject.myfirstapp.models import MyFirstAppData
>>> MyFirstAppData.objects.all()
[]
>>> p = MyFirstAppData(name="Jesus",
                       birthday="2009-05-05")
>>> p.save()
>>> p.id
1
>>> MyFirstAppData.objects.filter(name="Jesus")
...
>>> MyFirstAppData.objects.get(pk=1).name
u'Jesus'
```

Generador de canales

- Django viene con módulos para generar canales RSS y Atom
- View de alto nivel que genera el canal (feed):

```
(r'^feeds/(?P<url>.*)/$', 'django.contrib.syndication.view
    {'feed_dict': feeds}),
```

- Hay que proporcionar un diccionario con la correspondencia canal a objeto Feed:

```
feeds = {
     'latest': LatestEntries,
     'categories': LatestEntriesByCategory,
}
```

Generador de canales: objetos Feed

- Representan los datos de un canal:

```
from django.contrib.syndication.feeds import Feed
from content.models import Pages

class LatestEntries(Feed):
    title = "My CMS contents"
    link = "/feed/"
    description = "Contents of my CMS."

    def items(self):
        return Pages.objects.order_by('-pub_date')[:5]
```

- También hay que definir plantillas (templates) para <title> y <description> de cada item del canal RSS

Tests

- Pruebas de comportamiento del programa
- Pruebas "internas" (vía llamadas directas)
  Usando `django.test` (que extiende `unittest`)
- Pruebas "externas" (via API HTTP)
  Usando el "Django test client"
- Permite probar también plantillas, formularios, modelos...
- Normalmente, en fichero `tests.py`

```
% ./manage.py test
Creating test database for alias 'default'...
System check identified no issues (0 silenced).
........
----------------------------------------------------------------------
Ran 8 tests in 0.022s

OK
Destroying test database for alias 'default'...
```

https://docs.djangoproject.com/en/3.1/topics/testing/

Tests (pruebas internas)

```
from django.test import TestCase
from . import views

class TestViews(TestCase):

    def setUp(self):
        """Ejecutado antes de los tests"""
        ...

    def test_simple(self):
        """Test de la función aux"""
        result = views.aux(...)
        self.assertEqual(result, expected)
```

Tests (pruebas API HTTP)

```
from django.test import SimpleTestCase
from . import views

class TestHTTP(SimpleTestCase):
    """Tests of HTTP API"""

    def setUp(self):
        """Ejecutado antes de los tests"""
        ...

    def test_get_ok(self):
        response = self.client.get('/')
        self.assertEqual(response.status_code, 200)
        self.assertEqual(response.resolver_match.func, views.main)
        self.assertInHTML("<h1>Title</h1>",
                response.content.decode(encoding='UTF-8'))
```

Internacionalización

- Cadenas de traducción en código Python

```
from django.utils.translation import ugettext as _

def my_view(request):
    output = _("Welcome to my site.")
    return HttpResponse(output)

def my_view(request, m, d):
    output = _('Today is %(month)s, %(day)s.') %
       {'month': m, 'day': d}
    return HttpResponse(output)
```

Internacionalización (2)

- Cadenas de traducción en plantillas

```
<title>{% trans "This is the title." %}</title>

{% blocktrans %}
This string will have {{ value }} inside.
{% endblocktrans %}
```

Internacionalización (3)

- Traducciones en los lenguajes requeridos

```
django-admin.py makemessages -l es
```

- Activar el soporte para locale en Django

```
MIDDLEWARE_CLASSES = (
 'django.contrib.sessions.middleware.SessionMiddleware',
 'django.middleware.locale.LocaleMiddleware',
 'django.middleware.common.CommonMiddleware',
)
```

Referencias

- Documentación de Django
  `http://docs.djangoproject.com/en/dev`
- Libro de Django
  `http://www.djangobook.com`
- Documentación de Python
  `http://www.python.org/doc/`
- Tutorial sobre Django (e introducción a Django)
  `http://docs.djangoproject.com/en/dev/intro`

Hojas de estilo CSS

¿Qué es CSS?

- Es un lenguaje de hojas de estilos creado para **controlar el aspecto** o presentación de los documentos electrónicos definidos con HTML
- Es la mejor forma de **separar los contenidos y su presentación** y es imprescindible para crear páginas web complejas
  - Obliga a crear documentos HTML bien definidos y con significado completo (también llamados *documentos semánticos*)
  - Mejora la accesibilidad del documento
  - Reduce la complejidad de su mantenimiento
  - Permite visualizar el mismo documento en infinidad de dispositivos diferentes

Antes del CSS

```
<!DOCTYPE html>
<html>
<head>
 <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1"/>
 <title>Ejemplo de estilos sin CSS</title>
</head>

<body>
 <h1><font color="red" face="Arial" size="5">
    Titular de la página
 </font></h1>
 <p><font color="gray" face="Verdana" size="2">
   Un párrafo de texto no muy largo.
 </font></p>
</body>
</html>
```

Con CSS

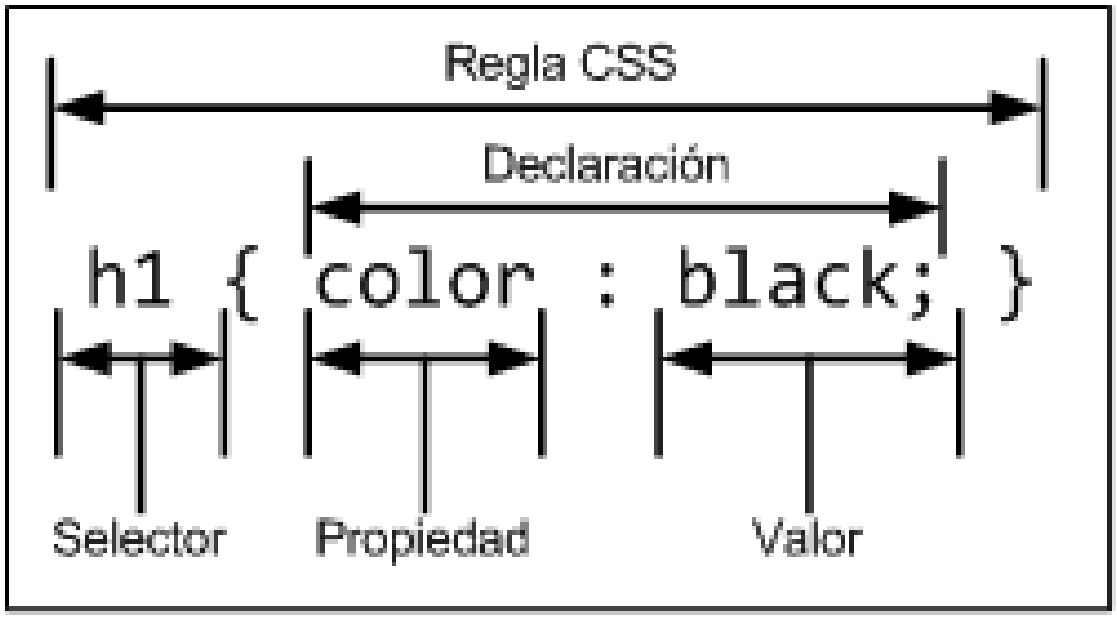
```
<!DOCTYPE html">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1"/>
<title>Ejemplo de estilos con CSS</title>
<style type="text/css">
  h1 { color: red;  font-family: Arial;   font-size: large;  }
  p  { color: gray; font-family: Verdana; font-size: medium; }
</style>
</head>

<body>
  <h1>Titular de la página</h1>
  <p>Un párrafo de texto no muy largo.</p>
</body>
</html>
```

CSS en un documento HTML

Se pueden integrar instrucciones CSS de varias maneras en un documento HTML:

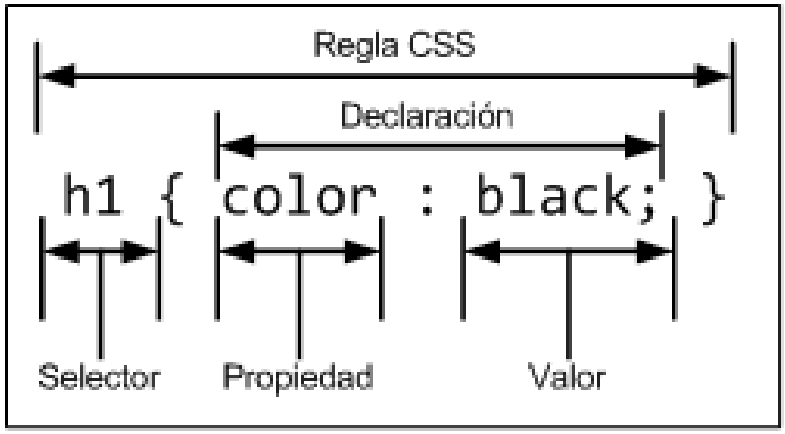
1. Incluir CSS en el mismo documento HTML
2. Definir CSS en un archivo externo
3. Incluir CSS en los elementos HTML

Glosario Básico (I)



- **Regla**: cada uno de los estilos que componen una hoja de estilos CSS. Cada regla está compuesta de una parte de "selectores", un símbolo de "llave de apertura" ({), otra parte denominada "declaración" y por último, un símbolo de "llave de cierre" (}).
- **Selector**: indica el elemento o elementos HTML a los que se aplica la regla CSS.

Glosario Básico (y II)



- **Declaración**: especifica los estilos que se aplican a los elementos. Está compuesta por una o más propiedades CSS.
- **Propiedad**: característica que se modifica en el elemento seleccionado, como por ejemplo su tamaño de letra, su color de fondo, etc.
- **Valor**: establece el nuevo valor de la característica modificada en el elemento.

CSS 2.1 define 115 propiedades, mientras que CSS 3 define 239 propiedades.

Selectores

- A un mismo elemento HTML se le pueden aplicar varias reglas
- Cada regla puede aplicarse a un número ilimitado de elementos
- Cuando el selector de dos o más reglas CSS es idéntico, se pueden agrupar las declaraciones de las reglas para hacer las hojas de estilos más eficientes
- Cuando se establece el valor de una propiedad CSS en un elemento, sus elementos descendientes heredan de forma automática el valor de esa propiedad

CSS 2.1 incluye una docena de tipos diferentes de selectores, que permiten seleccionar de forma muy precisa elementos individuales o conjuntos de elementos dentro de una página web.

Selectores básicos

1. Selector universal
2. Selector de tipo o etiqueta
3. Selector descendente
4. Selector de clase
5. Selector de identidad

Selector de tipo o etiqueta

- Selecciona todos los elementos de la página cuya etiqueta HTML coincide con el valor del selector
- Se pueden agrupar todas las reglas individuales en una sola regla con un selector múltiple.
- Buena práctica: agrupar las propiedades comunes de varios elementos en una única regla CSS y posteriormente definir las propiedades específicas de esos mismos elementos

```
h1, h2, h3 {
  color: #8A8E27;
  font-weight: normal;
  font-family: Arial, Helvetica, sans-serif;
}

h1 { font-size: 2em; }
h2 { font-size: 1.5em; }
h3 { font-size: 1.2em; }
```

Selector de clase más específico

- Combinando el selector de tipo y el selector de clase, se obtiene un selector mucho más específico.

```
p.destacado { color: red }
[...]
  <p class="destacado">Lorem ipsum dolor sit amet...</p>
  <p>Nunc sed lacus et <a href="#" class="destacado">est
adipiscing</a> accumsan...</p>
  <p>Class aptent taciti <em class="destacado">sociosqu
ad</em> litora...</p>
```

CSS: Consideraciones adicionales

CSS
Consideraciones Adicionales

(transparencias de referencia)

Selector Universal

- No se utiliza habitualmente
- Generalmente es equivalente para poner estilo a $< body >$
- Se suele combinar con otros selectores y además, forma parte de algunos hacks muy utilizados

```
* {
  margin: 0;
  padding: 0;
}
```

Selector descendente

- Selecciona los elementos que se encuentran dentro de otros elementos. Un elemento es descendiente de otro cuando se encuentra entre las etiquetas de apertura y de cierre del otro elemento.

```
p span { color: red; }
[...]
<p>
  ...
  <span>texto1</span>
  ...
  <a href="">...<span>texto2</span></a>
  ...
</p>
```

Al resto de elementos $< span >$ de la página que no están dentro de un elemento $< p >$, no se les aplica la regla CSS anterior.

Ejercicio

¿Qué elementos se seleccionarían con estos tipos de selectores?

- p a span em { text-decoration: underline; }
- p, a, span, em { text-decoration: underline; }
- p a { color: red; }
- p * a { color: red; }

Selector de clase

- Se utiliza el atributo class de HTML sobre ese elemento para indicar directamente la regla CSS que se le debe aplicar
- Se crea en el archivo CSS una nueva regla llamada destacado con todos los estilos que se van a aplicar al elemento
- Se prefija el valor del atributo class con un punto (.)

```
.destacado { color: red; }
[...]
  <p class="destacado">Lorem ipsum dolor sit amet...</p>
  <p>Nunc sed lacus et
<a href="#" class="destacado">est adipiscing</a>
  accumsan...</p>
  <p>Class aptent taciti <em class="destacado">sociosqu
ad</em> litora...</p>
```

Ejercicio

¿Qué elementos se seleccionarían con estos tipos de selectores?

- p.aviso { ... }
- p .aviso { ... }
- p, .aviso { ... }
- *.aviso { ... }

Selectores de identificador

- Aplica estilos CSS a un único elemento de la página
- El valor del atributo id no se puede repetir en dos elementos diferentes de una misma página

```
#destacado { color: red; }

<p>Primer párrafo</p>
<p id="destacado">Segundo párrafo</p>
<p>Tercer párrafo</p>
```

Ejercicio

¿Qué elementos se seleccionarían con estos tipos de selectores?

- p#aviso { ... }
- p #aviso { ... }
- p, #aviso { ... }
- *#aviso { ... }

¿Tienen sentido estos selectores? ¿Cuándo?

Ejercicio: Combinación de selectores

¿Qué elementos se seleccionarían con estos tipos de selectores?

- .aviso .especial { ... }
- div.aviso span.especial { ... }
- ul#menuPrincipal li.destacado a#inicio { ... }

Colisión de estilos (simplificado)

1. Cuanto más específico sea un selector, más importancia tiene su regla asociada.
2. A igual especificidad, se considera la última regla indicada.

Unidades de medida

- Unidades absolutas
  - in, cm, mm, pt, pc
- Unidades relativas
  - em, ex, px
- Porcentajes

En general, se recomienda el uso de unidades relativas siempre que sea posible

Normalmente se utilizan píxel y porcentajes para definir el layout del documento (básicamente, la anchura de las columnas y de los elementos de las páginas) y em y porcentajes para el tamaño de letra de los textos.

Colores

- Palabras clave
  - aqua, black, blue, fuchsia, gray...
- RGB decimal
- RGB porcentual
- RGB hexadecimal

Tipos de elementos (I)

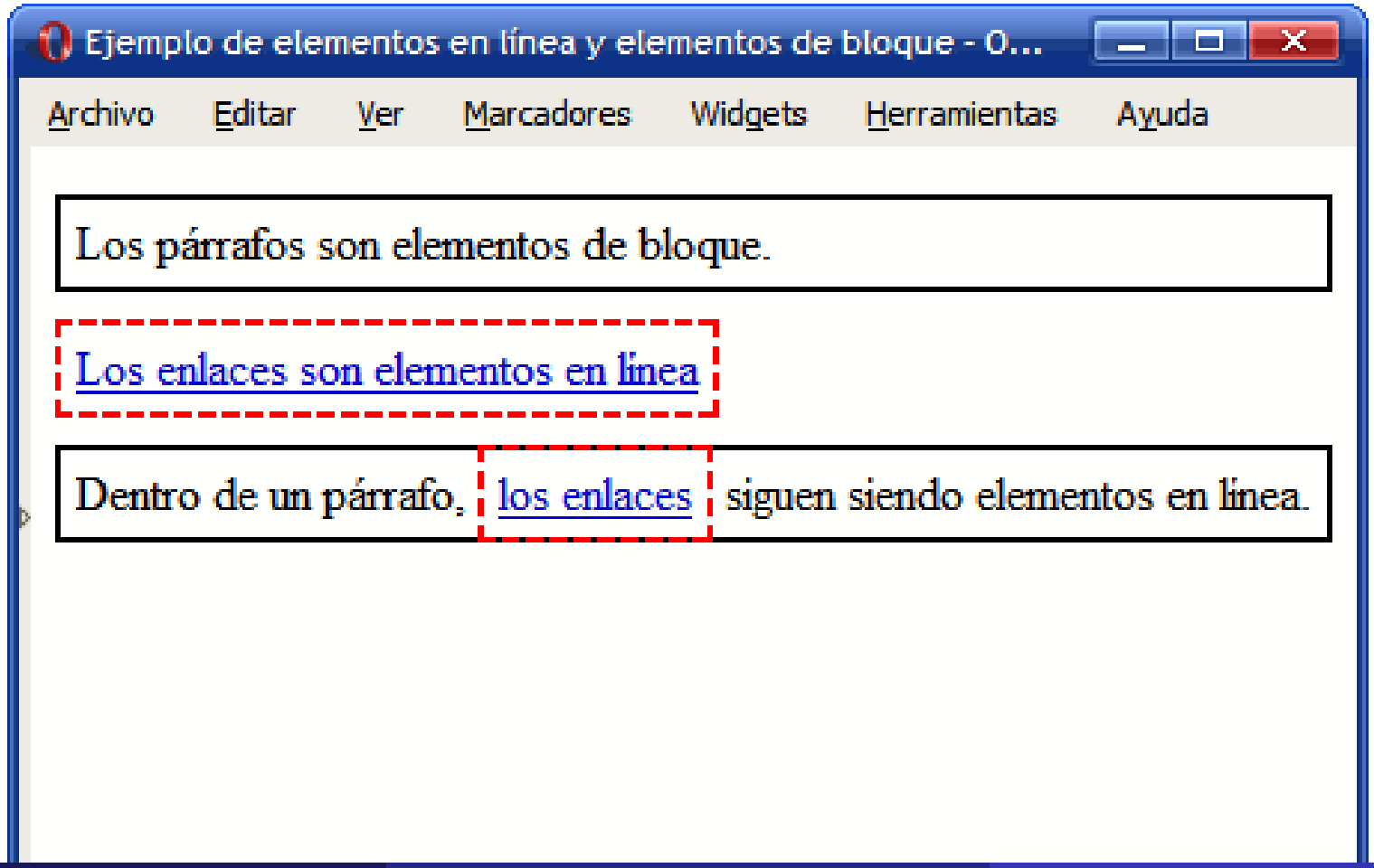
El estándar HTML clasifica a todos sus elementos en dos grandes grupos:
Elementos de línea:

- Los elementos en línea ("inline elements" en inglés) no empiezan necesariamente en nueva línea y sólo ocupan el espacio necesario para mostrar sus contenidos.

Tipos de elementos (II)

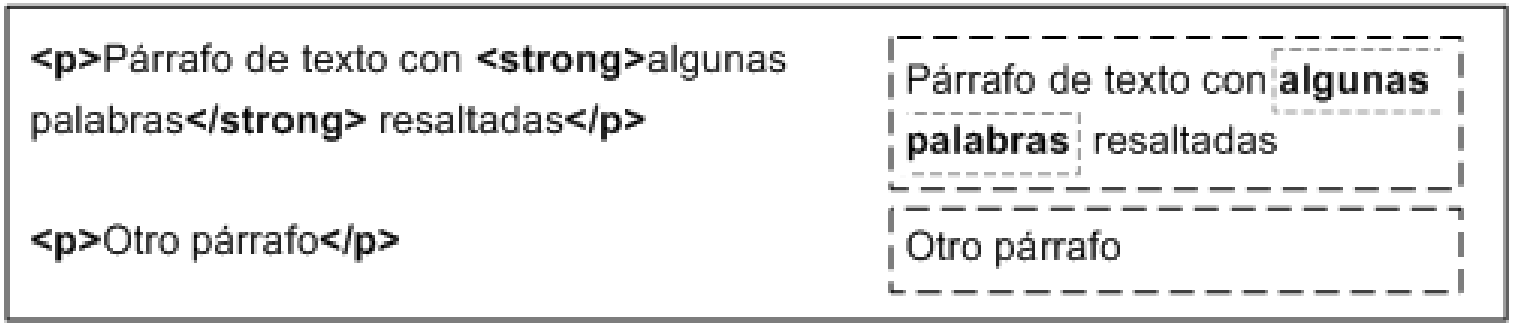
Elementos de bloque:

- Los elementos de bloque ("block elements" en inglés) siempre empiezan en una nueva línea y ocupan todo el espacio disponible hasta el final de la línea



El modelo de cajas

- Es el comportamiento de CSS que hace que todos los elementos de las páginas se representen mediante cajas rectangulares
- Cada vez que se inserta una etiqueta HTML, se crea una nueva caja rectangular que encierra los contenidos de ese elemento



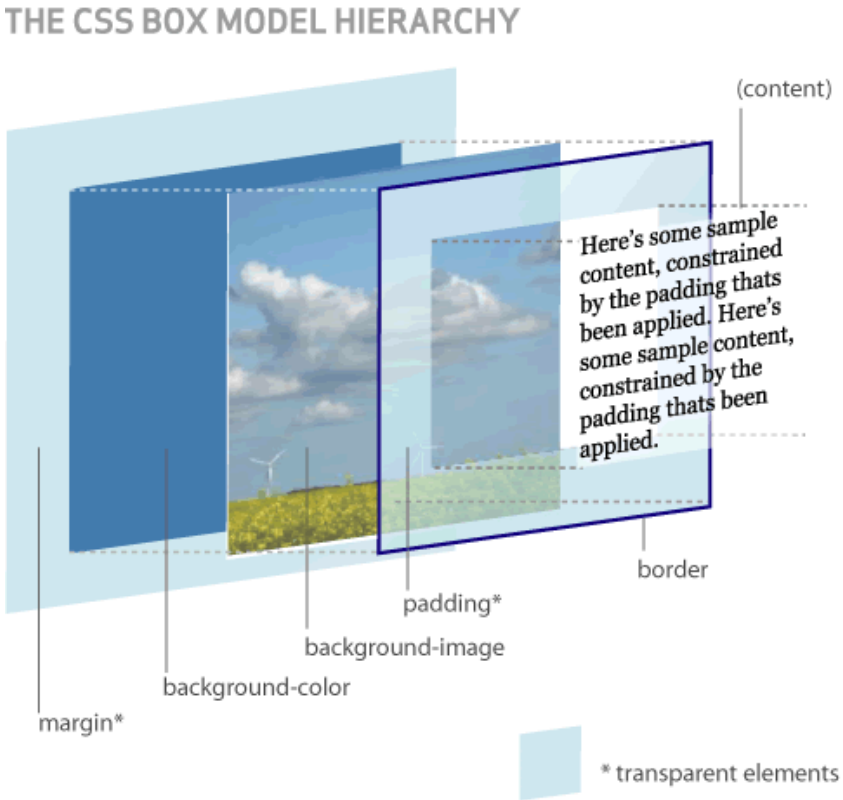
El modelo de cajas (II)

- No son visibles a simple vista porque inicialmente no muestran ningún color de fondo ni ningún borde

Ejemplo de http://www.alistapart.com/ después de forzar a que todas las cajas muestren su borde

El modelo de cajas (III)

- Los navegadores crean y colocan las cajas de forma automática, pero CSS permite modificar todas sus características. Cada una de las cajas está formada por seis partes, tal y como muestra la siguiente imagen:



Representación tridimensional del box model de CSS (Esquema utilizado con permiso de http://www.hicksdesign.co.uk/boxmodel/)

Partes que componen cada caja

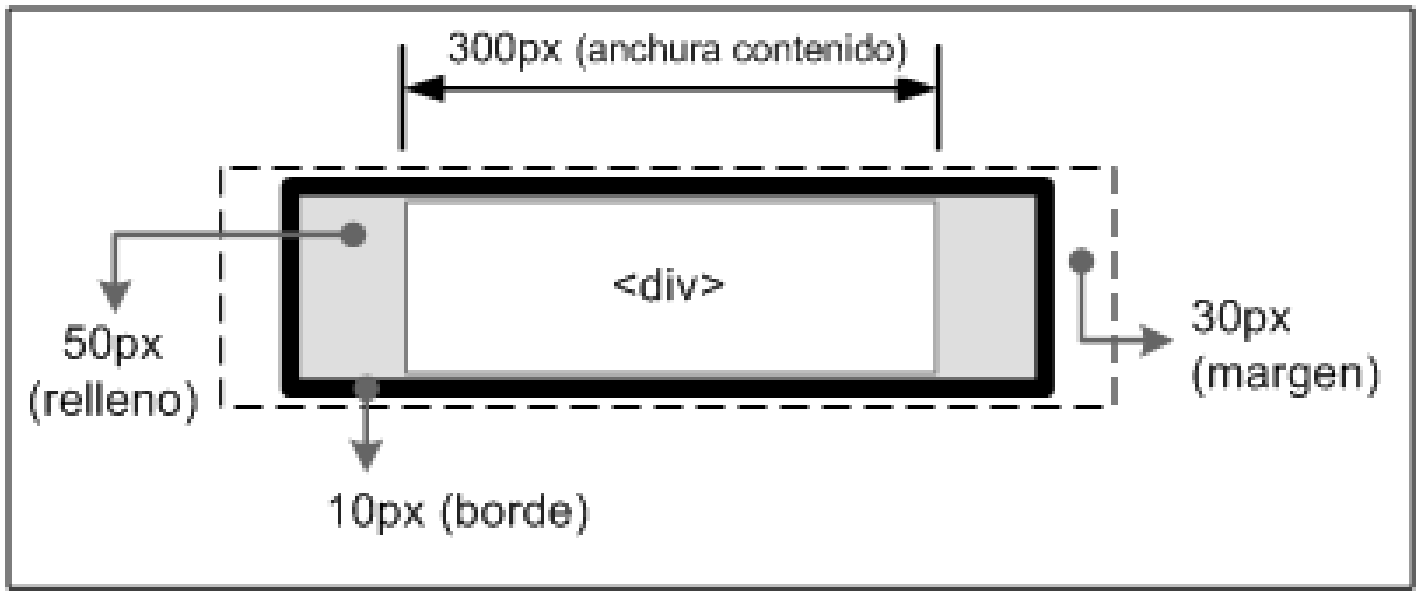
- **Contenido** (content): se trata del contenido HTML del elemento (las palabras de un párrafo, una imagen, el texto de una lista de elementos, etc.)
- **Relleno** (padding): espacio libre opcional existente entre el contenido y el borde.
- **Borde** (border): línea que encierra completamente el contenido y su relleno.
- **Imagen de fondo** (background image): imagen que se muestra por detrás del contenido y el espacio de relleno.
- **Color de fondo** (background color): color que se muestra por detrás del contenido y el espacio de relleno.
- **Margen** (margin): separación opcional existente entre la caja y el resto de cajas adyacentes.

Margen, relleno, bordes y modelo de cajas (I)

- El margen, el relleno y los bordes establecidos a un elemento determinan la anchura y altura final del elemento

```
div {
  width: 300px;
  padding-left:  50px;
  padding-right: 50px;
  margin-left:   30px;
  margin-right:  30px;
  border: 10px solid black;
}
```

Margen, relleno, bordes y modelo de cajas (y II)

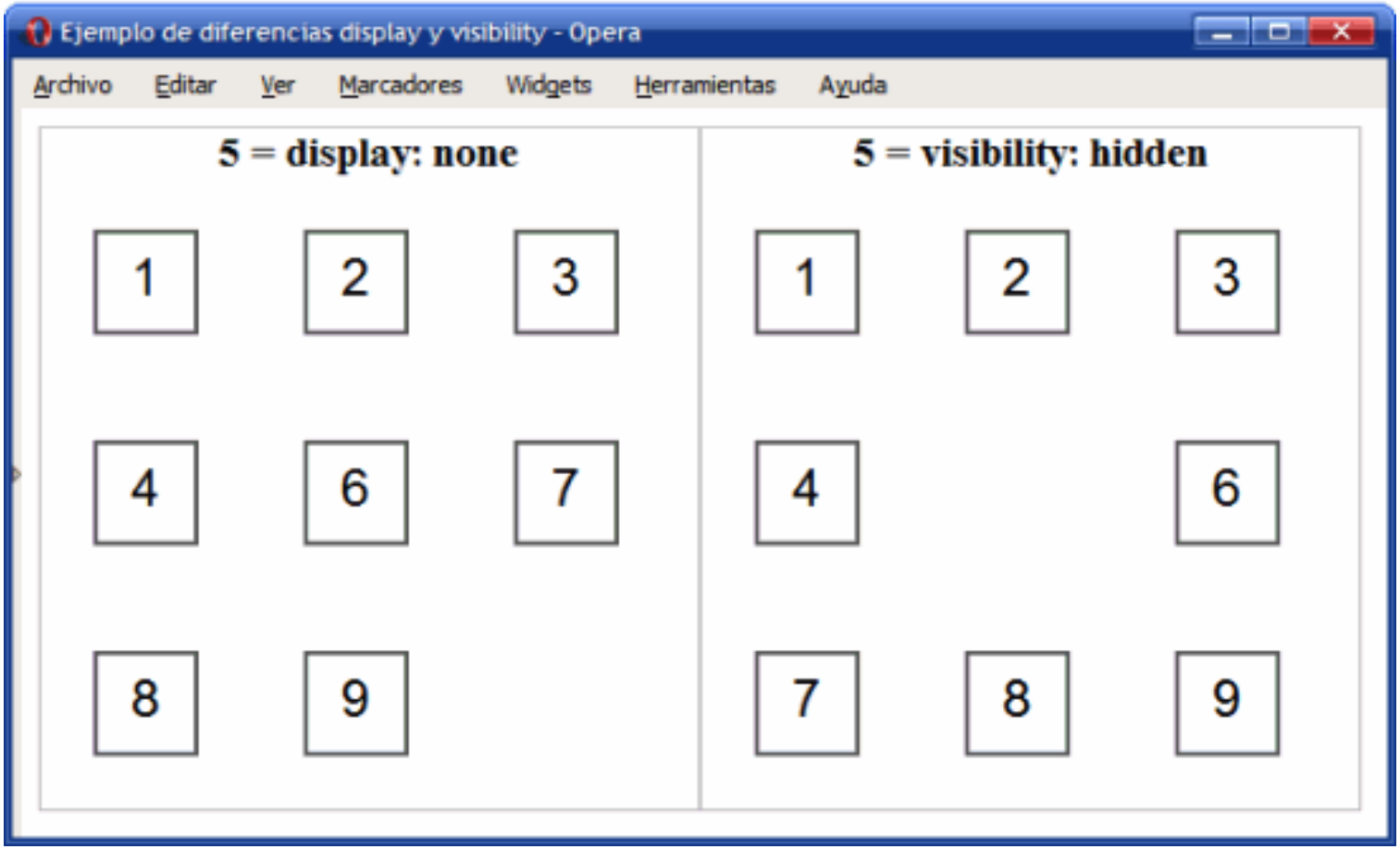


De esta forma, la anchura del elemento en pantalla sería igual a la suma de la anchura original, los márgenes, los bordes y los rellenos:
30px + 10px + 50px + 300px + 50px + 10px + 30px = 480 píxel

Visualización

- CSS define otras cuatro propiedades para controlar su visualización: display, visibility, overflow y z-index.
- La propiedad display permite ocultar completamente un elemento haciendo que desaparezca de la página. Como el elemento oculto no se muestra, el resto de elementos de la página se mueven para ocupar su lugar.
- La propiedad display también permite modificar el comportamiento de un elemento a bloque (block) o en línea (inline).
- La propiedad visibility permite hacer invisible un elemento, lo que significa que el navegador crea la caja del elemento pero no la muestra. En este caso, el resto de elementos de la página no modifican su posición, ya que aunque la caja no se ve, sigue ocupando sitio.

Diferencias entre display y visibility



Propiedades margin

Propiedades	**margin-top, margin-right, margin-bottom, margin-left**
Valores	< *medida* > — < *porcentaje* > — auto — inherit
Se aplica a	Todos los elementos, salvo margin-top y margin-bottom que sólo se aplican a los elementos de bloque y a las imágenes
Valor inicial	0
Descripción	Establece cada uno de los márgenes horizontales y verticales de un elemento

Cuadro: Definición de la propiedad *margin-top*, *margin-right*, *margin-bottom*, *margin-left* de CSS

Propiedad margin (propiedad *shorthand*)

Propiedad	**margin**
Valores	(< *medida* > — < *porcentaje* > — auto) 1, 4 — inherit
Se aplica a	Todos los elementos salvo algunos casos especiales de elementos mostrados como tablas
Valor inicial	-
Descripción	Establece de forma directa todos los márgenes de un elemento

Cuadro: Definición de la propiedad margin de CSS

Propiedad margin (propiedad *shorthand*) (y II)

La notación 1, 4 de la definición anterior significa que la propiedad margin admite entre uno y cuatro valores, con el siguiente significado:

- Si solo se indica un valor, todos los márgenes tienen ese valor.
- Si se indican dos valores, el primero se asigna al margen superior e inferior y el segundo se asigna a los márgenes izquierdo y derecho.
- Si se indican tres valores, el primero se asigna al margen superior, el tercero se asigna al margen inferior y el segundo valor se asigna los márgenes izquierdo y derecho.
- Si se indican los cuatro valores, el orden de asignación es: margen superior, margen derecho, margen inferior y margen izquierdo.

Orden de visualización

- El relleno y el margen son transparentes, por lo que en el espacio ocupado por el relleno se muestra el color o imagen de fondo (si están definidos)
- En el espacio ocupado por el margen se muestra el color o imagen de fondo de su elemento padre (si están definidos)
- Si ningún elemento padre tiene definido un color o imagen de fondo, se muestra el color o imagen de fondo de la propia página (si están definidos)
- Si una caja define tanto un color como una imagen de fondo, la imagen tiene más prioridad y es la que se visualiza
- Si la imagen de fondo no cubre totalmente la caja del elemento o si la imagen tiene zonas transparentes, también se visualiza el color de fondo.

Propiedad width

Propiedad	**width**
Valores	$<$ *medida* $>$ — $<$ *porcentaje* $>$ — auto — inherit
Se aplica a	Todos los elementos, salvo los elementos en línea que no sean imágenes, las filas de tabla y los grupos de filas de tabla
Valor inicial	auto
Descripción	Establece la anchura de un elemento

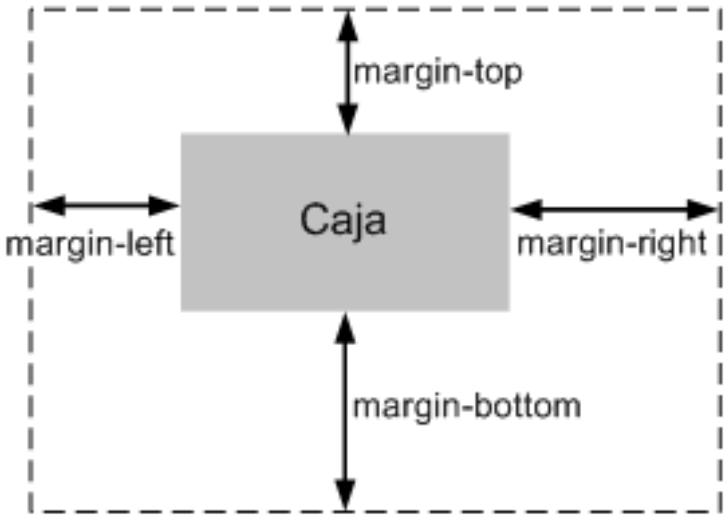
Cuadro: Definición de la propiedad *width* de CSS

Propiedad height

Propiedad	**height**
Valores	$<$ *medida* $>$ — $<$ *porcentaje* $>$ — auto — inherit
Se aplica a	Todos los elementos, salvo los elementos en línea que no sean imágenes, las columnas de tabla y los grupos de columnas de tabla
Valor inicial	auto
Descripción	Establece la altura de un elemento

Cuadro: Definición de la propiedad *height* de CSS

Márgenes



- En vez de utilizar la etiqueta $<$ *blockquote* $>$ de HTML, debería utilizarse la propiedad margin-left de CS
- Los márgenes verticales (margin-top y margin-bottom) sólo se pueden aplicar a los elementos de bloque y las imágenes, mientras que los márgenes laterales (margin-left y margin-right) se pueden aplicar a cualquier elemento

El margen vertical

Es algo peculiar:

- Cuando se juntan dos o más márgenes verticales, se fusionan de forma automática y la altura del nuevo margen será igual a la altura del margen más alto de los que se han fusionado.
- Si un elemento está contenido dentro de otro elemento, sus márgenes verticales se fusionan y resultan en un nuevo margen de la misma altura que el mayor margen de los que se han fusionado
- Si no se diera este comportamiento y se estableciera un determinado margen a todos los párrafos, el primer párrafo no mostraría un aspecto homogéneo respecto de los demás.

Relleno

Propiedades	**padding-top, padding-right, padding-bottom, padding-left**
Valores	< *medida* > — < *porcentaje* > — inherit
Se aplica a	Todos los elementos excepto algunos elementos de tablas como grupos de cabeceras y grupos de pies de tabla
Valor inicial	0
Descripción	Establece cada uno de los rellenos horizontales y verticales de un elemento

Cuadro: Definición de la propiedad padding-top, padding-right, padding-bottom, padding-left de CSS

Propiedad	**padding**
Valores	(< *medida* > — < *porcentaje* >) 1, 4 — inherit
Se aplica a	Todos los elementos excepto algunos elementos de tablas como grupos de cabeceras y grupos de pies de tabla
Valor inicial	-
Descripción	Establece de forma directa todos los rellenos de los elementos

Cuadro: Definición de la propiedad padding de CSS

Anchura de los bordes

Propiedades	**border-top-width, border-right-width, border-bottom-width, border-left-width**
Valores	(< *medida* > — thin — medium — thick) — inherit
Se aplica a	Todos los elementos
Valor inicial	Medium
Descripción	Establece la anchura de cada uno de los cuatro bordes de los elementos

Cuadro: Definición de la propiedad border-top-width, border-right-width, border-bottom-width, border-left-width de CSS

Anchura de los bordes (shorthand)

Propiedad	**border-width**
Valores	(< *medida* > — thin — medium — thick) 1, 4 — inherit
Se aplica a	Todos los elementos
Valor inicial	Medium
Descripción	Establece la anchura de todos los bordes del elemento

Cuadro: Definición de la propiedad border-width de CSS

Color de los bordes

Propiedades	**border-top-color, border-right-color, border-bottom-color, border-left-color**
Valores	< *color* > — transparent — inherit
Se aplica a	Todos los elementos
Valor inicial	-
Descripción	Establece el color de cada uno de los cuatro bordes de los elementos

Cuadro: Definición de la propiedad border-top-color, border-right-color, border-bottom-color, border-left-color de CSS

Color de los bordes (shorthand)

Propiedad	**border-color**
Valores	(< *color* > — transparent) 1, 4 — inherit
Se aplica a	Todos los elementos
Valor inicial	-
Descripción	Establece el color de todos los bordes del elemento

Cuadro: Definición de la propiedad border-color de CSS

Estilo de los bordes

Propiedades	**border-top-style, border-right-style, border-bottom-style, border-left-style**
Valores	none — hidden — dotted — dashed — solid — double — groove — ridge — inset — outset — inherit
Se aplica a	Todos los elementos
Valor inicial	none
Descripción	Establece el estilo de cada uno de los cuatro bordes de los elementos

Cuadro: Definición de la propiedad border-top-style, border-right-style, border-bottom-style, border-left-style de CSS

Estilo de los bordes *shorthand*

Propiedad	**border-style**
Valores	(none — hidden — dotted — dashed — solid — double — groove — ridge — inset — outset) 1, 4 — inherit
Se aplica a	Todos los elementos
Valor inicial	-
Descripción	Establece el estilo de todos los bordes del elemento

Cuadro: Definición de la propiedad border-style de CSS

Propiedades *shorthand* para bordes

Propiedades	**border-top, border-right, border-bottom, border-left**
Valores	(< *medida_borde* > —— < *color_borde* > —— < *estilo_borde* >) — inherit
Se aplica a	Todos los elementos
Valor inicial	-
Descripción	Establece el estilo completo de cada uno de los cuatro bordes de los elementos

Cuadro: Definición de la propiedad border-top, border-right, border-bottom, border-left de CSS

Propiedad *shorthand* para borde (global)

Propiedad	**border**
Valores	(< *medida_borde* > —— < *color_borde* > —— < *estilo_borde* >) — inherit
Se aplica a	Todos los elementos
Valor inicial	-
Descripción	Establece el estilo completo de todos los bordes de los elementos

Cuadro: Definición de la propiedad border de CSS

Más sobre bordes

- Como el valor por defecto de la propiedad border-style es none, si una propiedad shorthand no establece explícitamente el estilo de un borde, el elemento no muestra ese borde
- Cuando los cuatro bordes no son idénticos pero sí muy parecidos, se puede utilizar la propiedad border para establecer de forma directa los atributos comunes de todos los bordes y posteriormente especificar para cada uno de los cuatro bordes sus propiedades particulares:

```
h1 {
  border: solid #000;
  border-top-width: 6px;
  border-left-width: 8px;
}
```

Fondos

- Puede ser un color simple o una imagen.
- Solamente se visualiza en el área ocupada por el contenido y su relleno, ya que el color de los bordes se controla directamente desde los bordes y las zonas de los márgenes siempre son transparentes
- Se puede establecer de forma simultánea un color y una imagen de fondo. En este caso, la imagen se muestra delante del color, por lo que solamente si la imagen contiene zonas transparentes es posible ver el color de fondo.

Propiedad background-color

Propiedad	**background-color**
Valores	*color* > — transparent — inherit
Se aplica a	Todos los elementos
Valor inicial	transparent
Descripción	Establece un color de fondo para los elementos

Cuadro: Definición de la propiedad background-color de CSS

Propiedad background-image

Propiedad	**background-image**
Valores	*url* > — none — inherit
Se aplica a	Todos los elementos
Valor inicial	none
Descripción	Establece una imagen como fondo para los elementos

Cuadro: Definición de la propiedad background-image de CSS

Propiedad background-repeat

Propiedad	**background-repeat**
Valores	repeat — repeat-x — repeat-y — no-repeat — inherit
Se aplica a	Todos los elementos
Valor inicial	repeat
Descripción	Controla la forma en la que se repiten las imágenes de fondo

Cuadro: Definición de la propiedad background-repeat de CSS

Propiedad background-position

Propiedad	**background-position**
Valores	((< *porcentaje* > — < *medida* > — left — center — right) (< *porcentaje* > — < *medida* > — top — center — bottom)?) — ((left — center — right) —— (top — center — bottom)) — inherit
Se aplica a	Todos los elementos
Valor inicial	0 % 0 %
Descripción	Controla la posición en la que se muestra la imagen en el fondo del elemento

Cuadro: Definición de la propiedad background-position de CSS

Propiedad background-attachment

Propiedad	**background-attachment**
Valores	scroll — fixed — inherit
Se aplica a	Todos los elementos
Valor inicial	scroll
Descripción	Controla la forma en la que se visualiza la imagen de fondo: permanece fija cuando se hace scroll en la ventana del navegador o se desplaza junto con la ventana

Cuadro: Definición de la propiedad background-attachment de CSS

Propiedad *shorthand* background

Propiedad	**background**
Valores	(*background* – *color* —— *background* – *image* —— *background* – *repeat* —— *background* – *attachment* —— *background* – *position*) — inherit
Se aplica a	Todos los elementos
Valor inicial	-
Descripción	Establece todas las propiedades del fondo de un elemento

Cuadro: Definición de la propiedad background de CSS

Posicionamiento y visualización

- Los navegadores crean y posicionan de forma automática todas las cajas que forman cada página HTML
- El diseñador puede modificar la posición en la que se muestra cada caja.
- Existen **cinco tipos de posicionamiento** definidos para las cajas

Tipos de posicionamiento

1. **Normal o estático**: posicionamientosi no se indica lo contrario.
2. **Relativo**: consiste en posicionar una caja según el posicionamiento normal y después desplazarla respecto de su posición original.
3. **Absoluto**: la posición de una caja se establece de forma absoluta respecto de su elemento contenedor y el resto de elementos de la página ignoran la nueva posición del elemento.
4. **Fijo**: variante del posicionamiento absoluto que convierte una caja en un elemento inamovible, de forma que su posición en la pantalla siempre es la misma independientemente del resto de elementos e independientemente de si el usuario sube o baja la página en la ventana del navegador.
5. **Flotante**: desplaza las cajas todo lo posible hacia la izquierda o hacia la derecha de la línea en la que se encuentran.

Propiedad position

Propiedad	**position**
Valores	static — relative — absolute — fixed — inherit
Se aplica a	Todos los elementos
Valor inicial	static
Descripción	Selecciona el posicionamiento con el que se mostrará el elemento

Cuadro: Definición de la propiedad position de CSS

Significados propiedad position

- static: corresponde al posicionamiento normal o estático. Si se utiliza este valor, se ignoran los valores de las propiedades top, right, bottom y left que se verán a continuación.
- relative: corresponde al posicionamiento relativo. El desplazamiento de la caja se controla con las propiedades top, right, bottom y left.
- absolute: corresponde al posicionamiento absoluto. El desplazamiento de la caja también se controla con las propiedades top, right, bottom y left, pero su interpretación es mucho más compleja, ya que el origen de coordenadas del desplazamiento depende del posicionamiento de su elemento contenedor.
- fixed: corresponde al posicionamiento fijo. El desplazamiento se establece de la misma forma que en el posicionamiento absoluto, pero en este caso el elemento permanece inamovible en la pantalla.

Pseudo-clases

Como con los atributos id o class no es posible aplicar diferentes estilos a un mismo elemento en función de su estado, CSS introduce un nuevo concepto llamado pseudo-clases. Por ejemplo, en enlaces:

- **:link**: enlaces que apuntan a páginas o recursos que aún no han sido visitados por el usuario.
- **:visited**: enlaces que apuntan a recursos que han sido visitados anteriormente por el usuario. El historial de enlaces visitados se borra automáticamente cada cierto tiempo y el usuario también puede borrarlo manualmente.
- **:hover**: enlace sobre el que el usuario ha posicionado el puntero del ratón.
- **:active**: enlace que está pinchando el usuario. Los estilos sólo se aplican desde que el usuario pincha el botón del ratón hasta que lo suelta.

Propiedades top, right, bottom, left

Propiedades	**top**, **right**, **bottom**, **left**
Valores	< *medida* > — < *porcentaje* > — auto — inherit
Se aplica a	Todos los elementos posicionados
Valor inicial	auto
Descripción	Indican el desplazamiento horizontal y vertical del elemento respecto de su posición original

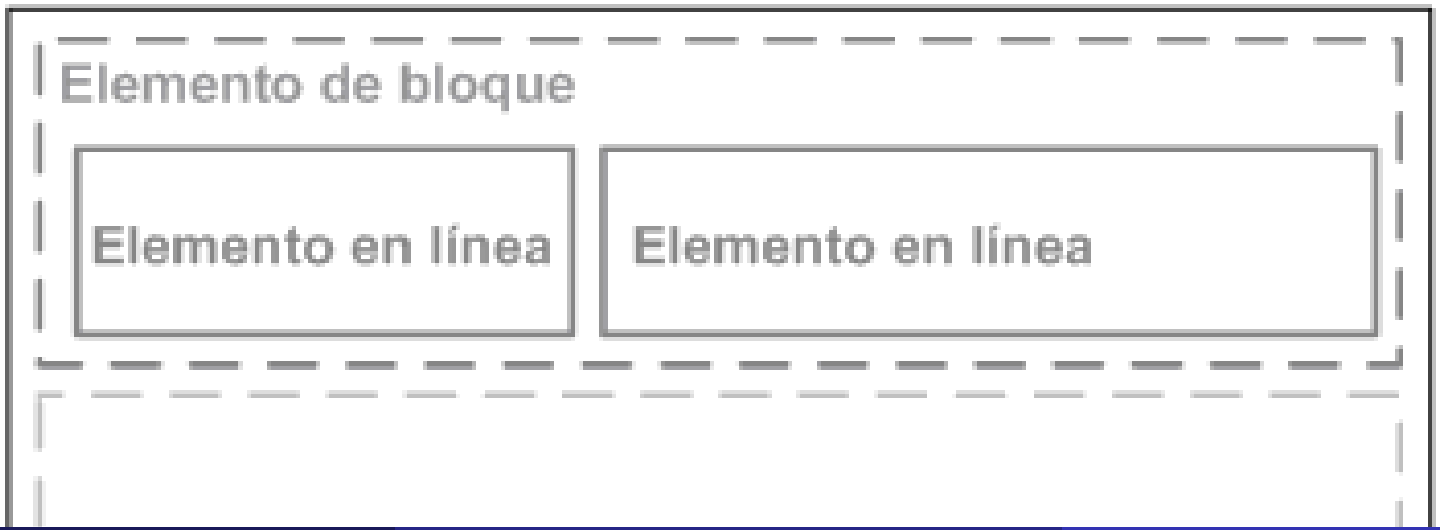
Cuadro: Definición de la propiedad top, right, bottom, left de CSS

Posicionamiento normal (o estático)

- Utilizado por defecto por los navegadores
- Sólo se tiene en cuenta si el elemento es de bloque o en línea, sus propiedades width y height y su contenido.
- Las cajas se muestran una debajo de otra comenzando desde el principio del elemento contenedor. La distancia entre las cajas se controla mediante los márgenes verticales.
- Si un elemento se encuentra dentro de otro, el elemento padre se llama "elemento contenedor" y determina tanto la posición como el tamaño de todas sus cajas interiores.

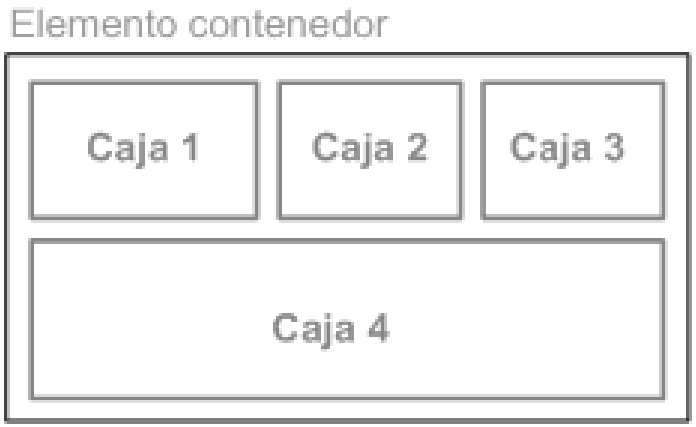
Posicionamiento normal (o estático) (y II)

- Los elementos en línea forman los "contextos de formato en línea". Las cajas se muestran una detrás de otra de forma horizontal comenzando desde la posición más a la izquierda de su elemento contenedor.
- Si las cajas en línea ocupan más espacio del disponible en su propia línea, el resto de cajas se muestran en las líneas inferiores.
- Si las cajas en línea ocupan un espacio menor que su propia línea, se puede controlar la distribución de las cajas mediante la propiedad text-align para centrarlas, alinearlas a la derecha o justificarlas.

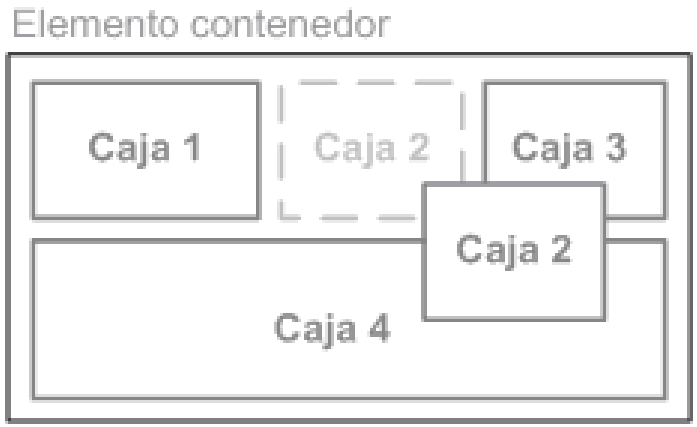


Posicionamiento relativo

- Desplaza una caja respecto de su posición original establecida mediante el posicionamiento normal. El desplazamiento de la caja se controla con las propiedades top, right, bottom y left.
- la propiedad top se emplea para mover las cajas de forma descendente, la propiedad bottom mueve las cajas de forma ascendente, la propiedad left se utiliza para desplazar las cajas hacia la derecha y la propiedad right mueve las cajas hacia la izquierda.



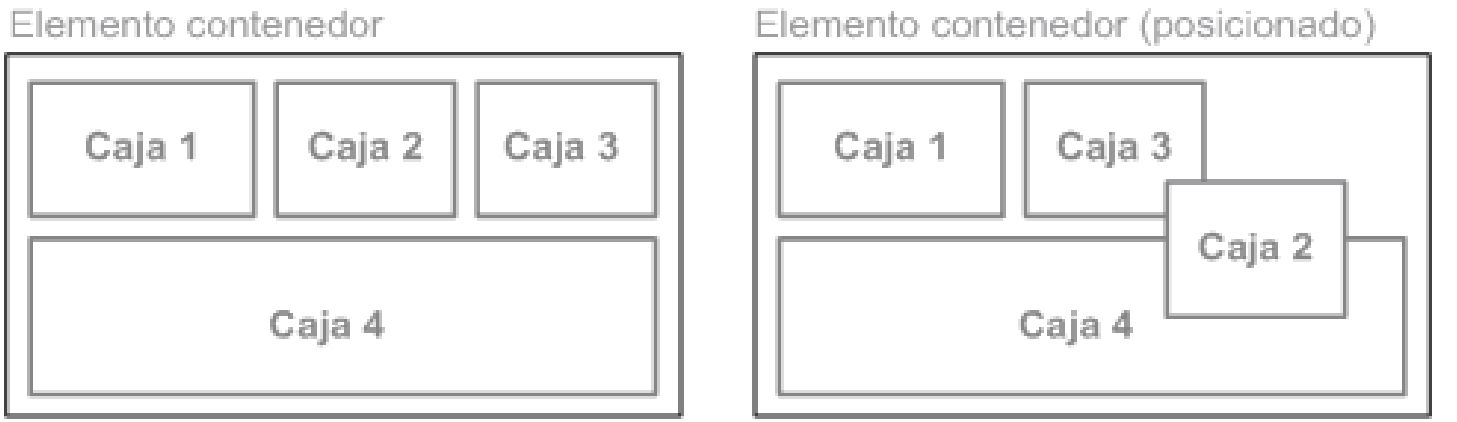
Posicionamiento normal



Posicionamiento relativo de la caja 2

Posicionamiento absoluto

- Se emplea para establecer de forma exacta la posición en la que se muestra la caja de un elemento.
- Cuando una caja se posiciona de forma absoluta, el resto de elementos de la página se ven afectados y modifican su posición.

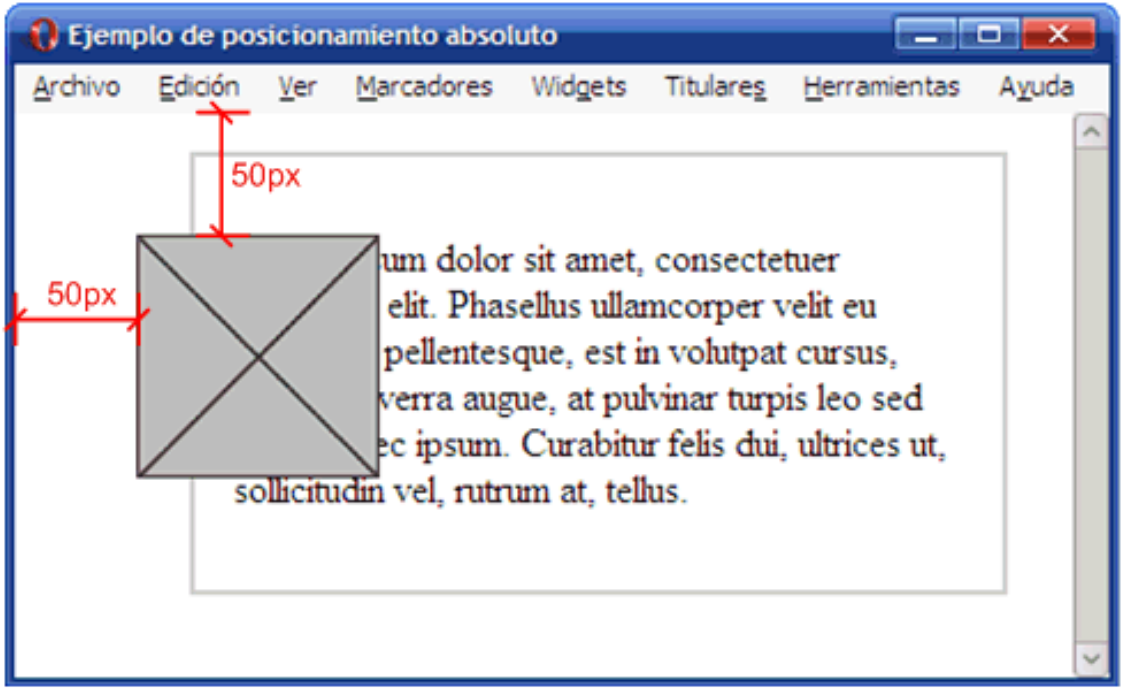


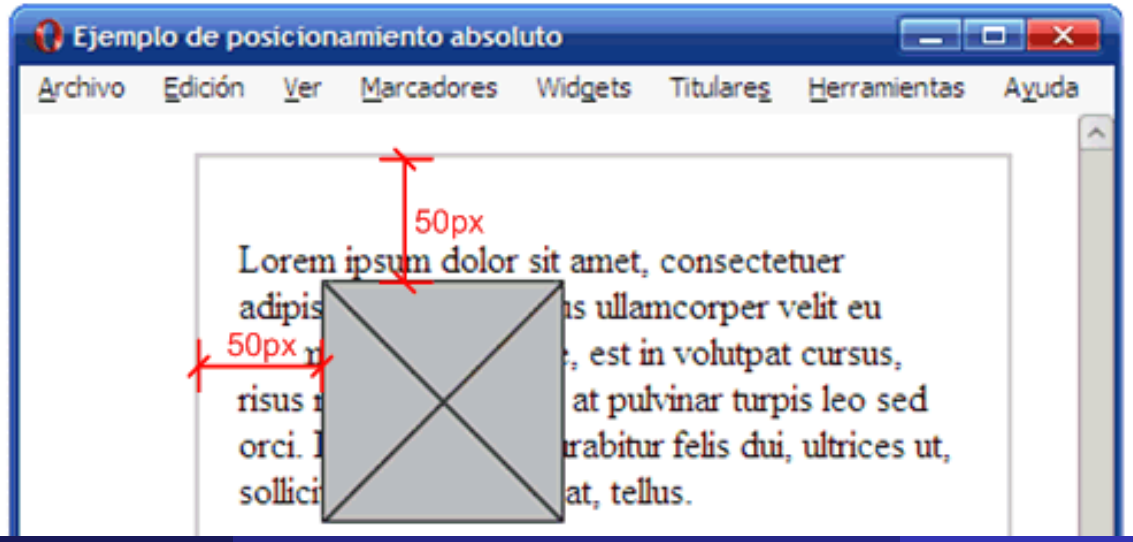
Posicionamiento normal **Posicionamiento absoluto de la caja 2**

Posicionamiento absoluto

- El primer elemento contenedor que esté posicionado de cualquier forma diferente a position: static se convierte en la referencia que determina la posición de la caja posicionada de forma absoluta.
- Si ningún elemento contenedor está posicionado, la referencia es la ventana del navegador, que no debe confundirse con el elemento $< body >$ de la página.
- Una vez determinada la referencia del posicionamiento absoluto, la interpretación de los valores de las propiedades top, right, bottom y left se realiza como sigue:
  - Top: desplazamiento desde el borde superior del elemento contenedor que se utiliza como referencia.
  - Right: ídem pero desde el borde derecho al borde derecho.
  - Left: ídem pero desde el borde izquierdo al borde izquierdo.
  - Bottom: ídem pero desde el borde inferior al borde inferior.

Diferencias entre posicionamiento absoluto y relativo



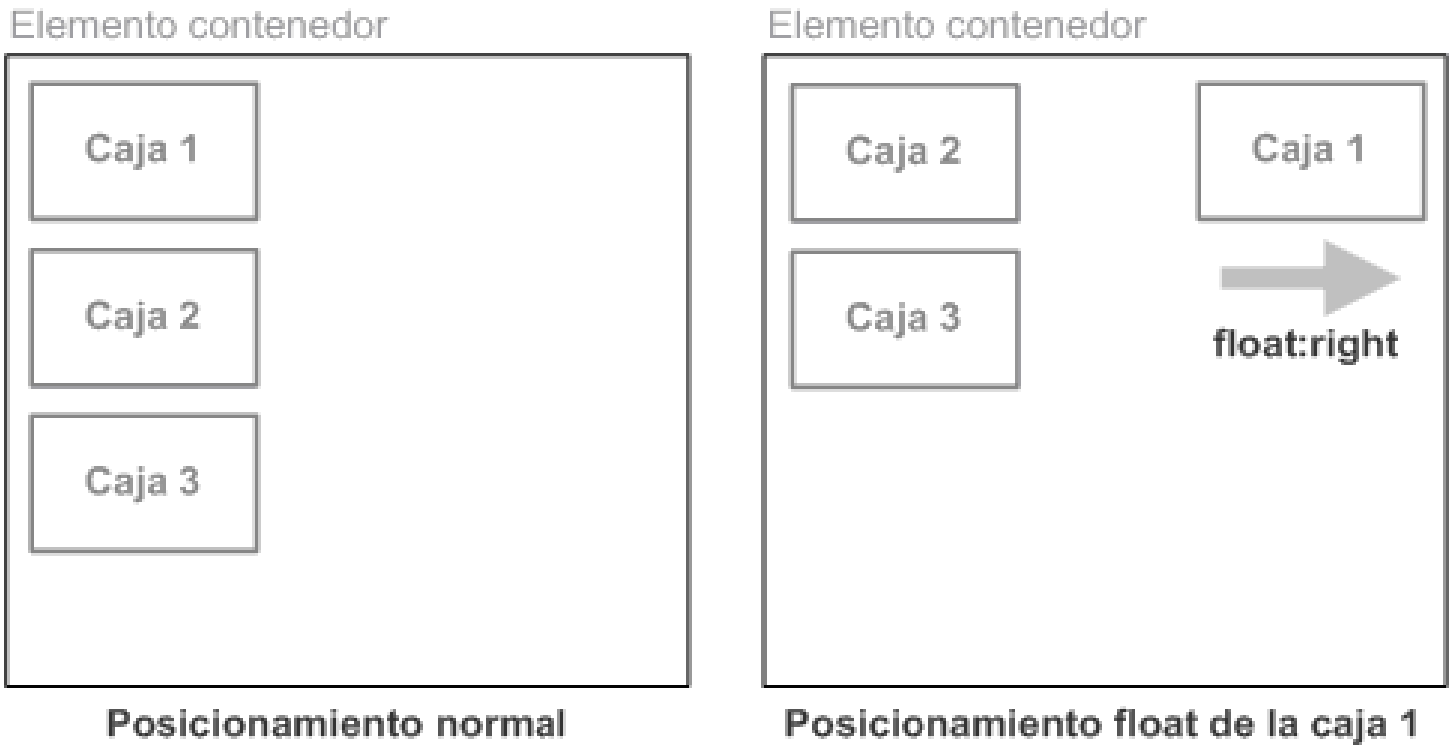


Posicionamiento fijo

- Es un caso particular del posicionamiento absoluto, ya que sólo se diferencian en el comportamiento de las cajas posicionadas.
- La principal característica de una caja posicionada de forma fija es que su posición es inamovible dentro de la ventana del navegador.
- El posicionamiento fijo hace que las cajas no modifiquen su posición ni aunque el usuario suba o baje la página en la ventana de su navegador.
- Si la página se visualiza en un medio paginado (por ejemplo en una impresora) las cajas posicionadas de forma fija se repiten en todas las páginas.

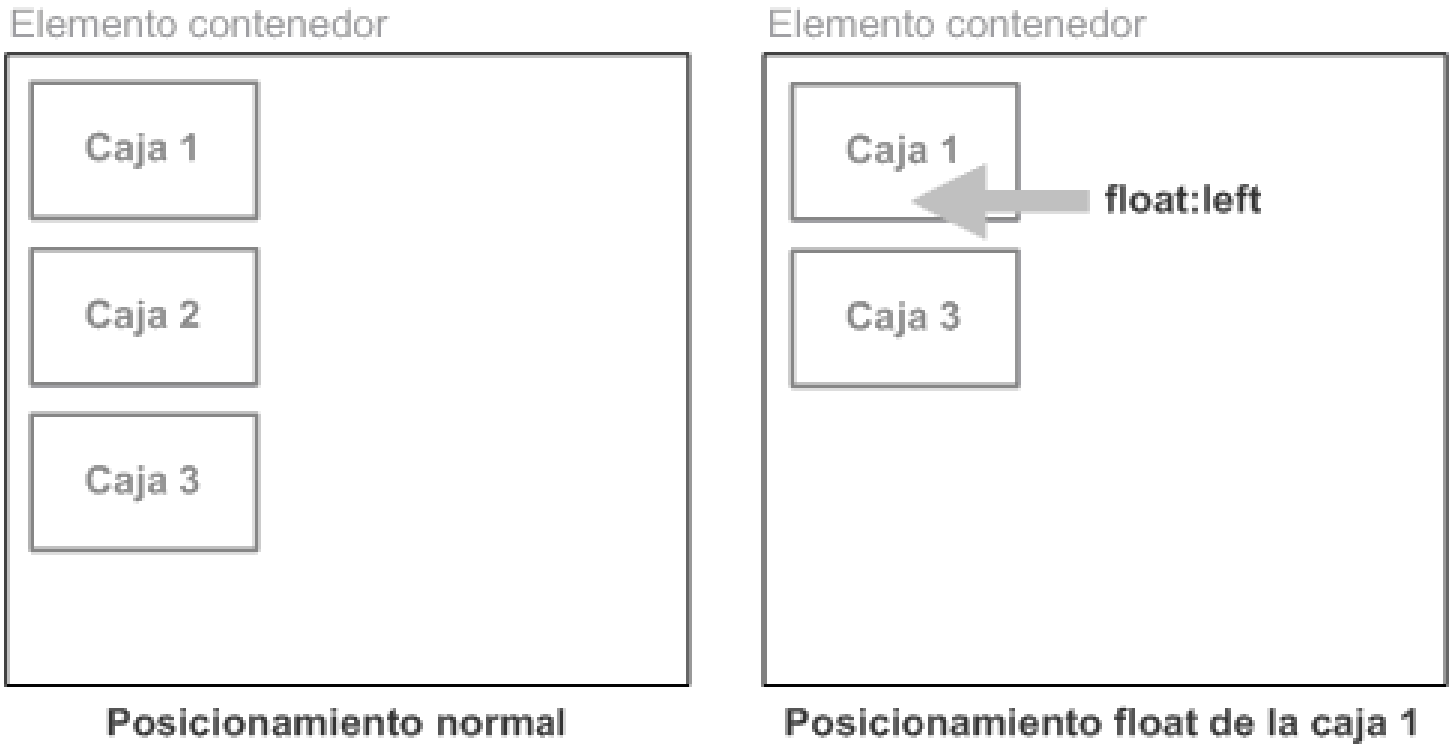
Posicionamiento flotante

- Cuando una caja se posiciona con el modelo de posicionamiento flotante, automáticamente se convierte en una caja flotante, lo que significa que se desplaza hasta la zona más a la izquierda o más a la derecha de la posición en la que originalmente se encontraba.



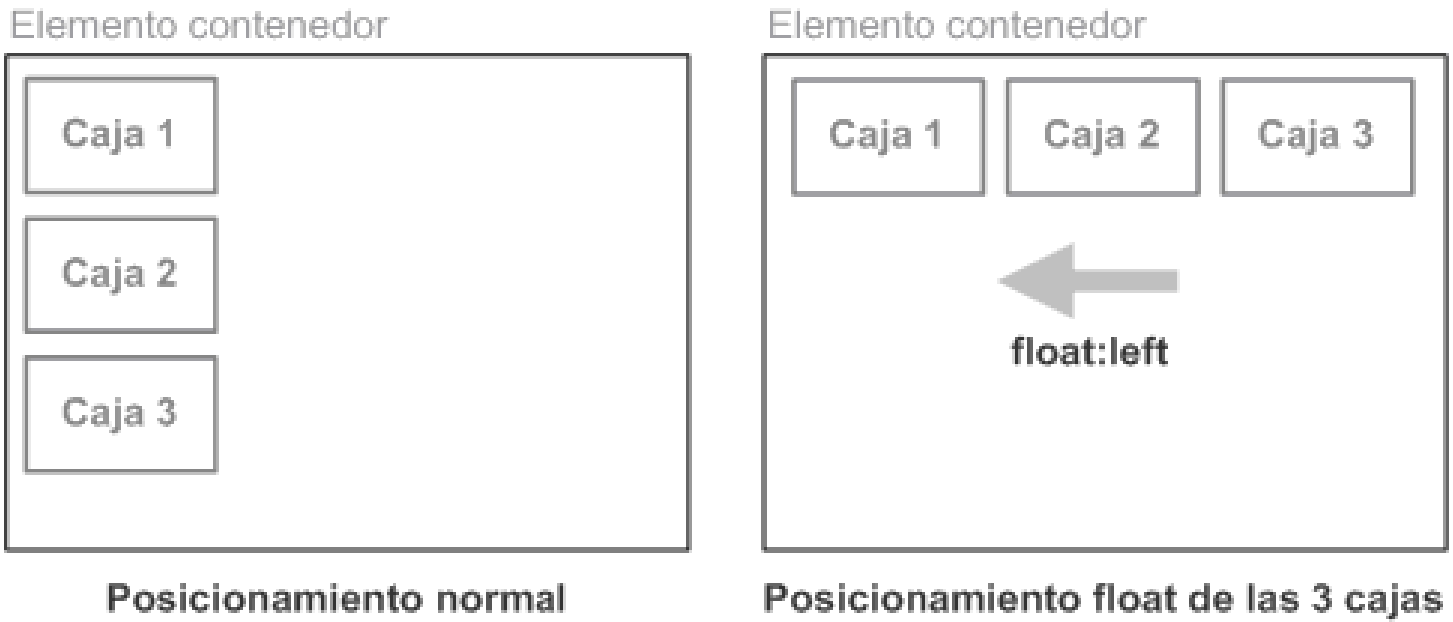
Posicionamiento flotante (y II)

- Cuando se posiciona una caja de forma flotante:
  - La caja deja de pertenecer al flujo normal de la página, lo que significa que el resto de cajas ocupan el lugar dejado por la caja flotante.
  - La caja flotante se posiciona lo más a la izquierda o lo más a la derecha posible de la posición en la que se encontraba originalmente.



Posicionamiento flotante (y III)

- Si existen otras cajas flotantes, al posicionar de forma flotante otra caja, se tiene en cuenta el sitio disponible.
- Si no existiera sitio en la línea actual, la caja flotante baja a la línea inferior hasta que encuentra el sitio necesario para mostrarse lo más a la izquierda o lo más a la derecha posible en esa nueva línea



Propiedad float

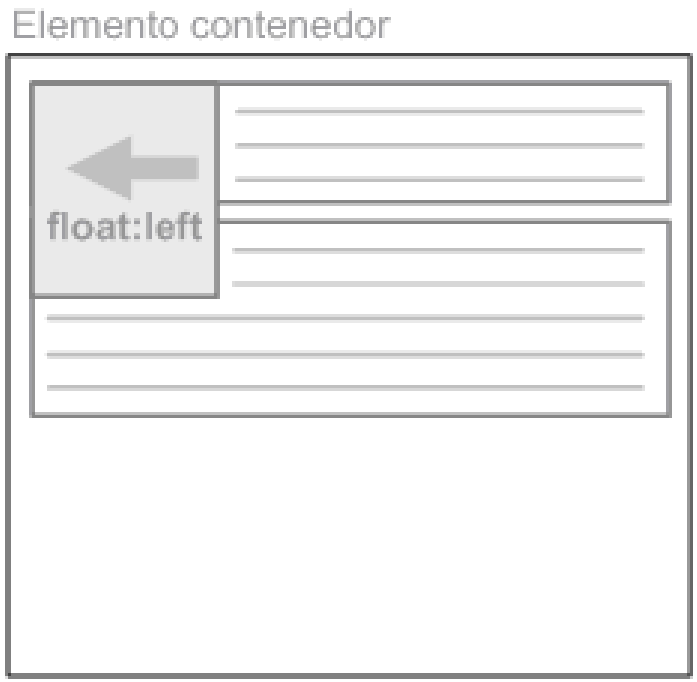
Propiedad	**float**
Valores	left — right — none — inherit
Se aplica a	Todos los elementos
Valor inicial	none
Descripción	Establece el tipo de posicionamiento flotante del elemento

Cuadro: Definición de la propiedad float de CSS

Posicionamiento flotante (y IV)

- Los elementos que se encuentran alrededor de una caja flotante adaptan sus contenidos para que fluyan alrededor del elemento posicionado
- Uno de los principales motivos para la creación del posicionamiento float fue precisamente la posibilidad de colocar imágenes alrededor de las cuales fluye el texto.

Posicionamiento normal



Posicionamiento float de la imagen

Propiedad clear

- La propiedad clear indica el lado del elemento HTML que no debe ser adyacente a ninguna caja posicionada de forma flotante. Si se indica el valor left, el elemento se desplaza de forma descendente hasta que pueda colocarse en una línea en la que no haya ninguna caja flotante en el lado izquierdo.
- La especificación oficial de CSS explica este comportamiento como "un desplazamiento descendente hasta que el borde superior del elemento esté por debajo del borde inferior de cualquier elemento flotante hacia la izquierda".

Propiedad clear

Propiedad	**clear**
Valores	none — left — right — both — inherit
Se aplica a	Todos los elementos de bloque
Valor inicial	none
Descripción	Indica el lado del elemento que no debe ser adyacente a ninguna caja flotante

Cuadro: Definición de la propiedad clear de CSS

Propiedad display

Propiedad	**display**
Valores	inline — block — none — list-item — run-in — inline-block — table — inline-table — table-row-group — table-header-group — table-footer-group — table-row — table-column-group — table-column — table-cell — table-caption — inherit
Se aplica a	Todos los elementos
Valor inicial	inline
Descripción	Permite controlar la forma de visualizar un elemento e incluso ocultarlo

Cuadro: Definición de la propiedad display de CSS

Propiedad visibility

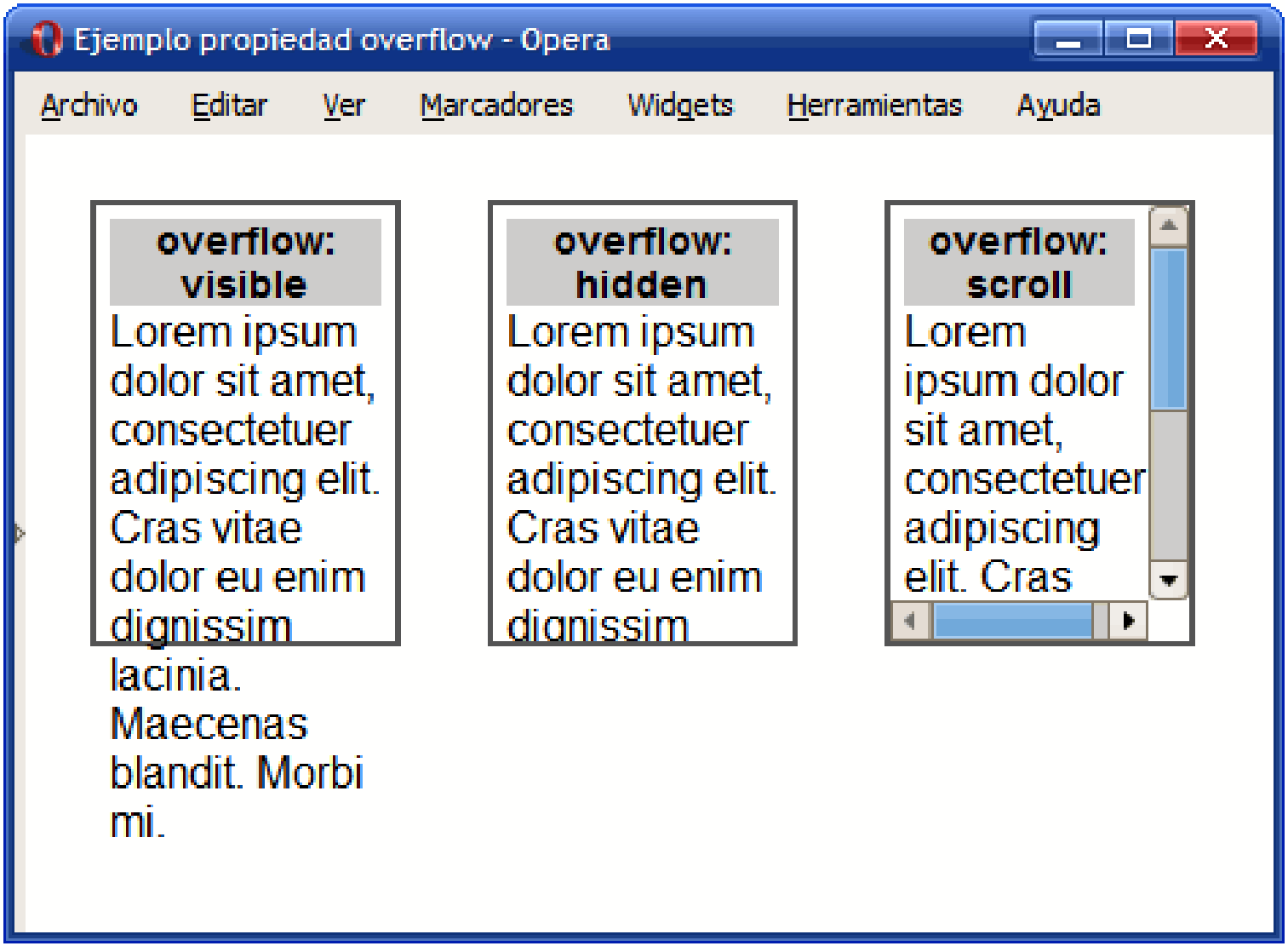
Propiedad	**visibility**
Valores	visible — hidden — collapse — inherit
Se aplica a	Todos los elementos
Valor inicial	visible
Descripción	Permite hacer visibles e invisibles a los elementos

Cuadro: Definición de la propiedad visibility de CSS

Propiedad overflow

- En algunas ocasiones el contenido de un elemento no cabe en el espacio reservado para ese elemento y se desborda.
- La situación más habitual en la que el contenido sobresale de su espacio reservado es cuando se establece la anchura y/o altura de un elemento mediante la propiedad width y/o height.
- Los valores de la propiedad overflow tienen el siguiente significado:
  - visible: el contenido no se corta y se muestra sobresaliendo la zona reservada para visualizar el elemento. Este es el comportamiento por defecto.
  - hidden: el contenido sobrante se oculta y sólo se visualiza la parte del contenido que cabe dentro de la zona reservada para el elemento.
  - scroll: solamente se visualiza el contenido que cabe dentro de la zona reservada para el elemento, pero también se muestran barras de scroll que permiten visualizar el resto del contenido.
  - auto: el comportamiento depende del navegador, aunque normalmente es el mismo que la propiedad scroll.

Propiedad overflow (y II)



Propiedad overflow (y III)

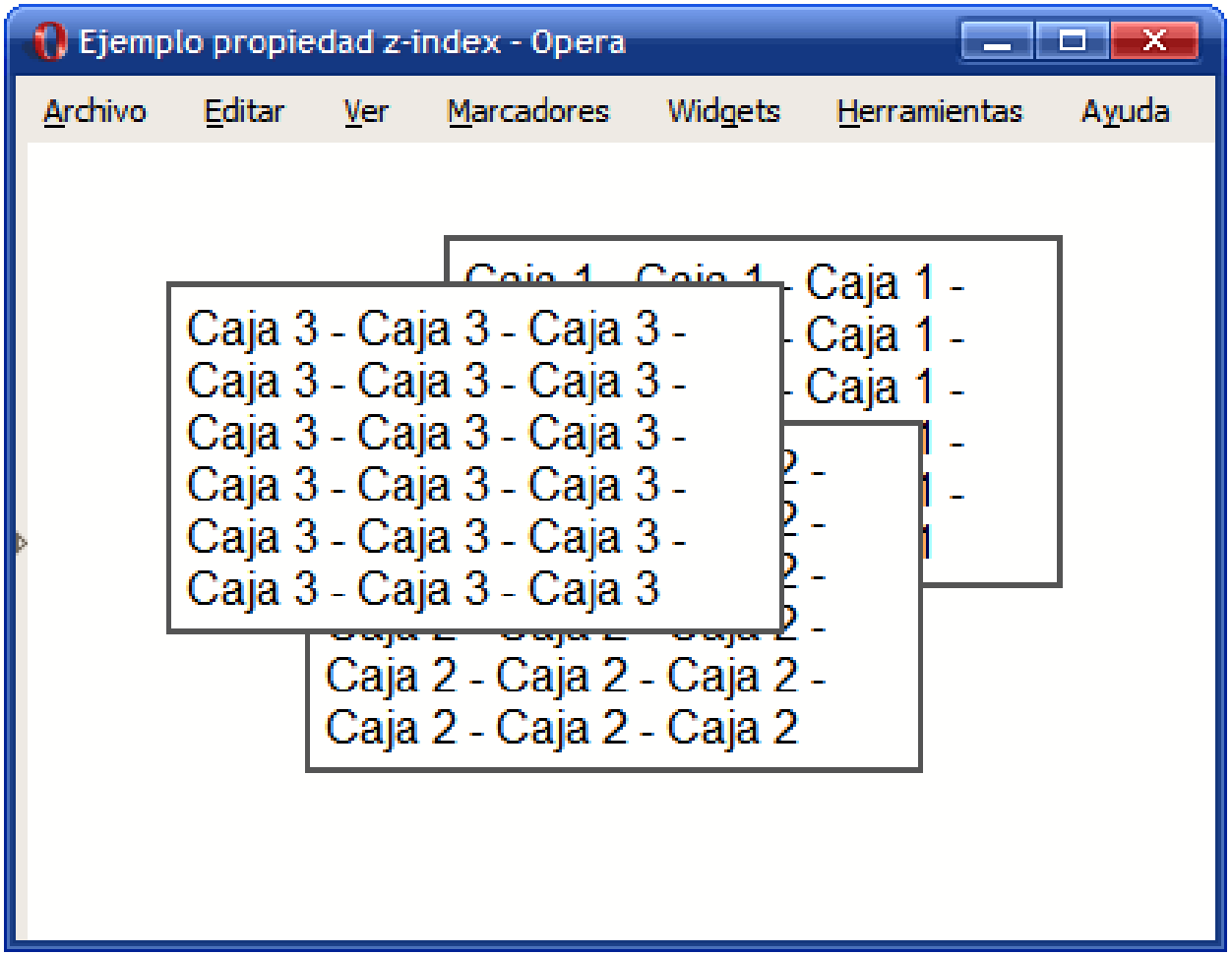
Propiedad	**overflow**
Valores	visible — hidden — scroll — auto — inherit
Se aplica a	Elementos de bloque y celdas de tablas
Valor inicial	visible
Descripción	Permite controlar los contenidos sobrantes de un elemento

Cuadro: Definición de la propiedad overflow de CSS

Propiedad z-index

- CSS permite controlar la posición tridimensional de las cajas posicionadas
- Es posible indicar las cajas que se muestran delante o detrás de otras cajas cuando se producen solapamientos.
- Cuanto más alto sea el valor numérico, más cerca del usuario se muestra la caja.

Propiedad z-index (II)



Propiedad z-index (y III)

Propiedad	**z-index**
Valores	auto — < *numero* > — inherit
Se aplica a	Elementos que han sido posicionados explícitamente
Valor inicial	auto
Descripción	Establece el nivel tridimensional en el que se muestra el elemento

Cuadro: Definición de la propiedad z-index de CSS

Selectores avanzados

1. Selector de hijos

```
p > span { color: blue; }
```

2. Selector adyacente

```
p + p { text-indent: 1.5em; }
```

3. Selector de atributos

```
a[class="externo"] { color: blue; }
a[class~="externo"] { color: blue; }
*[lang=en] { ... }
*[lang|="es"] { color : red }
```

Colisión de estilos

El método seguido por CSS para resolver las colisiones de estilos se muestra a continuación:

1. Determinar todas las declaraciones que se aplican al elemento para el medio CSS seleccionado.
2. Ordenar las declaraciones según su origen (CSS de navegador, de usuario o de diseñador) y su prioridad (palabra clave !important).
3. Ordenar las declaraciones según lo específico que sea el selector. Cuanto más genérico es un selector, menos importancia tienen sus declaraciones.
4. Si después de aplicar las normas anteriores existen dos o más reglas con la misma prioridad, se aplica la que se indicó en último lugar.

Medios CSS

- Permiten definir diferentes estilos para diferentes medios o dispositivos: pantallas, impresoras, móviles, proyectores, etc.
- Define algunas propiedades específicamente para determinados medios: la paginación y los saltos de página para los medios impresos o el volumen y tipo de voz para los medios de audio.
- Ejemplos:
  - screen: Pantallas de ordenador
  - print: Impresoras y navegadores en el modo "Vista Previa para Imprimir"
  - handheld: Dispositivos de mano: móviles, PDA, etc.

Formas de indicar el medio

1. Reglas de tipo @media

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

2. Reglas de tipo @import

```
@import url("estilos_basicos.css") screen;
@import url("estilos_impresora.css") print;
```

3. Medios definidos con la etiqueta

```
<link rel="stylesheet" type="text/css" media="screen" href="basico.css
<link rel="stylesheet" type="text/css" media="print, handheld" href="e
```

4. Medios definidos mezclando varios métodos

```
<link rel="stylesheet" type="text/css"  media="screen" href="basico.cs
@import url("estilos_seccion.css") screen;
@media print {
  /* Estilos específicos para impresora */
```

Comentarios

- El comienzo de un comentario se indica mediante los caracteres /* y el final del comentario se indica mediante */

```
/* Este es un comentario en CSS */
```

- Pueden ocupar tantas líneas como sea necesario, pero no se puede incluir un comentario dentro de otro comentario

```
/* Este es un
   comentario CSS de varias
   lineas */
```

Tipografía

- CSS define numerosas propiedades para modificar la apariencia del texto
- color se utiliza para establecer el color de la letra
- Como el valor de la propiedad color se hereda, normalmente se establece la propiedad color en el elemento body para establecer el color de letra de todos los elementos de la página
- font-family se utiliza para indicar el tipo de letra con el que se muestra el texto
- Suele definirse como una lista de tipos de letra alternativos separados por comas. El último valor de la lista es el nombre de la familia tipográfica genérica que más se parece al tipo de letra que se quiere utilizar.

Propiedad color

Propiedad	**color**
Valores	< *color* > — inherit
Se aplica a	Todos los elementos
Valor inicial	Depende del navegador
Descripción	Establece el color de letra utilizado para el texto

Cuadro: Definición de la propiedad color de CSS

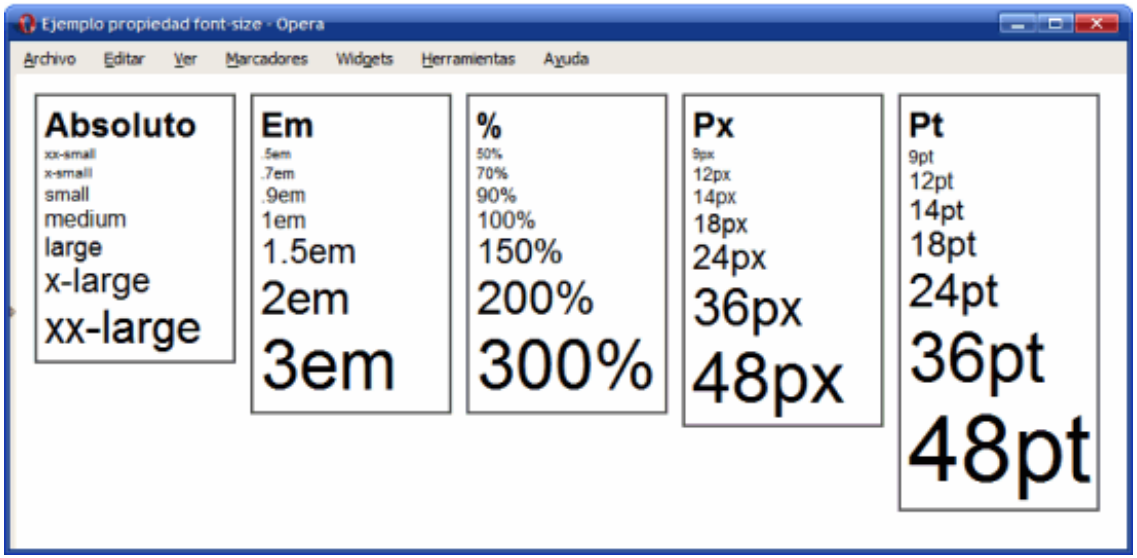
Propiedad font-family

Propiedad	**font-family**
Valores	((< *nombre_familia* > — < *familia_generica* >) (,*nombre_familia* > — < *familia_generica*)*) — inherit
Se aplica a	Todos los elementos
Valor inicial	Depende del navegador
Descripción	Establece el tipo de letra utilizado para el texto

Cuadro: Definición de la propiedad font-family de CSS

Propiedad font-size (I)

- Además de medida relativas, absolutas y de porcentajes, CSS permite utilizar una serie de palabras clave para indicar el tamaño de letra del texto:
  - tamaño_absoluto: indica el tamaño de letra de forma absoluta mediante alguna de las siguientes palabras clave: xx-small, x-small, small, medium, large, x-large, xx-large.
  - tamaño_relativo: indica de forma relativa el tamaño de letra del texto mediante dos palabras clave (larger, smaller) que toman como referencia el tamaño de letra del elemento padre.



Propiedad font-size (y II)

Propiedad	**font-size**
Valores	$<$ *tamano_absoluto* $>$ — $<$ *tamano_relativo* $>$ — $<$ *medida* $>$ — $<$ *porcentaje* $>$ — inherit
Se aplica a	Todos los elementos
Valor inicial	medium
Descripción	Establece el tamaño de letra utilizado para el texto

Cuadro: Definición de la propiedad font-size de CSS

Propiedad font-weight

Propiedad	**font-weight**
Valores	normal — bold — bolder — lighter — 100 — 200 — 300 — 400 — 500 — 600 — 700 — 800 — 900 — inherit
Se aplica a	Todos los elementos
Valor inicial	normal
Descripción	Establece la anchura de la letra utilizada para el texto

Cuadro: Definición de la propiedad font-weight de CSS

Propiedad font-style

Propiedad	**font-style**
Valores	normal — italic — oblique — inherit
Se aplica a	Todos los elementos
Valor inicial	normal
Descripción	Establece el estilo de la letra utilizada para el texto

Cuadro: Definición de la propiedad font-style de CSS

Propiedad font-variant

Propiedad	**font-variant**
Valores	normal — small-caps — inherit
Se aplica a	Todos los elementos
Valor inicial	normal
Descripción	Establece el estilo alternativo de la letra utilizada para el texto

Cuadro: Definición de la propiedad font-variant de CSS

Propiedad *short-hand* font

Propiedad	**font**
Valores	((< *font − style* > —— < *font − variant* > —— < *font − weight* >)? < *font − size* > (/ < *line − height* >)? < *font − family* >) — caption — icon — menu — message-box — small-caption — status-bar — inherit
Se aplica a	Todos los elementos
Valor inicial	-
Descripción	Permite indicar de forma directa todas las propiedades de la tipografía de un texto

Cuadro: Definición de la propiedad font de CSS

Propiedad *short-hand* font (y II)

- El orden en el que se deben indicar las propiedades del texto es el siguiente:
  - En primer lugar y de forma opcional se indican el font-style, font-variant y font-weight en cualquier orden.
  - A continuación, se indica obligatoriamente el valor de font-size seguido opcionalmente por el valor de line-height.
  - Por último, se indica obligatoriamente el tipo de letra a utilizar.

```
font: bold 1em "Trebuchet MS",Arial,Sans-Serif;
font: normal 0.9em "Lucida Grande", Verdana, Arial, Helvetica, sans-serif;
font: normal 1.2em/1em helvetica, arial, sans-serif;
```

Texto

- Además de las propiedades relativas a la tipografía del texto, CSS define numerosas propiedades que determinan la apariencia del texto en su conjunto.
- Estas propiedades adicionales permiten controlar
  - la alineación del texto,
  - el interlineado,
  - la separación entre palabras,
  - etc.

Propiedad text-align

Propiedad	**text-align**
Valores	left — right — center — justify — inherit
Se aplica a	Elementos de bloque y celdas de tabla
Valor inicial	left
Descripción	Establece la alineación del contenido del elemento

Cuadro: Definición de la propiedad text-align de CSS

Propiedad line-height

Propiedad	**line-height**
Valores	normal — < *numero* > — < *medida* > — < *porcentaje* > — inherit
Se aplica a	Todos los elementos
Valor inicial	normal
Descripción	Permite establecer la altura de línea de los elementos

Cuadro: Definición de la propiedad line-height de CSS

Propiedad text-decoration

Propiedad	**text-decoration**
Valores	none — (underline —— overline —— line-through —— blink) — inherit
Se aplica a	Todos los elementos
Valor inicial	none
Descripción	Establece la decoración del texto (subrayado, tachado, parpadeante, etc.)

Cuadro: Definición de la propiedad text-decoration de CSS

Propiedad text-transform

Propiedad	**text-transform**
Valores	capitalize — uppercase — lowercase — none — inherit
Se aplica a	Todos los elementos
Valor inicial	none
Descripción	Transforma el texto original (lo transforma a mayúsculas, a minúsculas, etc.)

Cuadro: Definición de la propiedad text-transform de CSS

Propiedad vertical-align

Propiedad	**vertical-align**
Valores	baseline — sub — super — top — text-top — middle — bottom — text-bottom — < *porcentaje* > — < *medida* > — inherit
Se aplica a	Elementos en línea y celdas de tabla
Valor inicial	baseline
Descripción	Determina la alineación vertical de los contenidos de un elemento

Cuadro: Definición de la propiedad vertical-align de CSS

Propiedad text-indent

Propiedad	**text-indent**
Valores	< *medida* > — < *porcentaje* > — inherit
Se aplica a	Los elementos de bloque y las celdas de tabla
Valor inicial	0
Descripción	Tabula desde la izquierda la primera línea del texto original

Cuadro: Definición de la propiedad text-indent de CSS

Propiedad letter-spacing

Propiedad	**letter-spacing**
Valores	normal — < *medida* > — inherit
Se aplica a	Todos los elementos
Valor inicial	normal
Descripción	Permite establecer el espacio entre las letras que forman las palabras del texto

Cuadro: Definición de la propiedad letter-spacing de CSS

Propiedad word-spacing

Propiedad	**word-spacing**
Valores	normal — < *medida* > — inherit
Se aplica a	Todos los elementos
Valor inicial	normal
Descripción	Permite establecer el espacio entre las palabras que forman el texto

Cuadro: Definición de la propiedad word-spacing de CSS

Propiedad white-space

Propiedad	**white-space**
Valores	normal — pre — nowrap — pre-wrap — pre-line — inherit
Se aplica a	Todos los elementos
Valor inicial	normal
Descripción	Establece el tratamiento de los espacios en blanco del texto

Cuadro: Definición de la propiedad white-space de CSS

Propiedad white-space (y II)

El significado de cada uno de los valores es el siguiente:

- normal: comportamiento por defecto de HTML.
- pre: se respetan los espacios en blanco y las nuevas líneas (exactamente igual que la etiqueta $< pre >$). Si la línea es muy larga, se sale del espacio asignado para ese contenido.
- nowrap: elimina los espacios en blanco y las nuevas líneas. Si la línea es muy larga, se sale del espacio asignado para ese contenido.
- pre-wrap: se respetan los espacios en blanco y las nuevas líneas, pero ajustando cada línea al espacio asignado para ese contenido.
- pre-line: elimina los espacios en blanco y respeta las nuevas líneas, pero ajustando cada línea al espacio asignado para ese contenido.

Propiedad list-style-type

Propiedad	**list-style-type**
Valores	disc — circle — square — decimal — decimal-leading-zero — lower-roman — upper-roman — lower-greek — lower-latin — upper-latin — armenian — georgian — lower-alpha — upper-alpha — none — inherit
Se aplica a	Elementos de una lista
Valor inicial	disc
Descripción	Permite establecer el tipo de viñeta mostrada para una lista

Cuadro: Definición de la propiedad list-style-type de CSS

Propiedad list-style-position

Propiedad	**list-style-position**
Valores	inside — outside — inherit
Se aplica a	Elementos de una lista
Valor inicial	outside
Descripción	Permite establecer la posición de la viñeta de cada elemento de una lista

Cuadro: Definición de la propiedad list-style-type de CSS

Propiedad list-style-image

Propiedad	**list-style-image**
Valores	$< url >$ — none — inherit
Se aplica a	Elementos de una lista
Valor inicial	none
Descripción	Permite reemplazar las viñetas automáticas por una imagen personalizada

Cuadro: Definición de la propiedad list-style-image de CSS

Propiedad *shorthand* list-style

Propiedad	**list-style**
Valores	($< list-style-type >$ —— $< list-style-position >$ —— $< list-style-image >$) — inherit
Se aplica a	Elementos de una lista
Valor inicial	-
Descripción	Propiedad que permite establecer de forma simultánea todas las opciones de una lista

Cuadro: Definición de la propiedad list-style de CSS

Imágenes según el estilo del enlace

- En ocasiones, puede resultar útil incluir un pequeño icono al lado de un enlace para indicar el tipo de contenido que enlaza.
- Este tipo de imágenes son puramente decorativas en vez de imágenes de contenido, por lo que se deberían añadir con CSS y no con elementos de tipo $< img >$. Utilizando la propiedad background (y background-image) se puede personalizar el aspecto de los enlaces para que incluyan un pequeño icono a su lado.
- La técnica consiste en mostrar una imagen de fondo sin repetición en el enlace y añadir el padding necesario al texto del enlace para que no se solape con la imagen de fondo.

Imágenes según el estilo del enlace (II)

```
a { margin: 1em 0; float: left; clear: left; }

.rss {
  color: #E37529;
  padding: 0 0 0 18px;
  background: #FFF url(imagenes/rss.gif) no-repeat left center;
}

.pdf {
  padding: 0 0 0 22px;
  background: #FFF url(imagenes/pdf.png) no-repeat left center;
}
```

```
<a href="#">Enlace con el estilo por defecto</a>
<a class="rss" href="#">Enlace a un archivo RSS</a>
<a class="pdf" href="#">Enlace a un documento PDF</a>
```

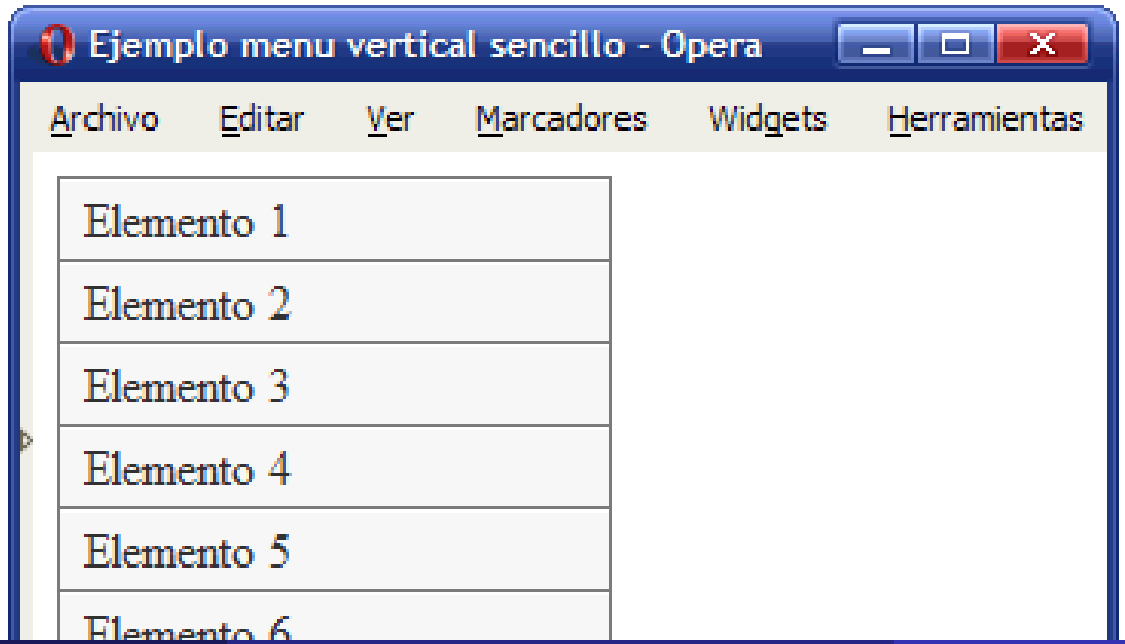
Imágenes según el estilo del enlace (y III)

Mostrar los enlaces como si fueran botones

```
a { margin: 1em 0; float: left; clear: left; }
a.boton {
  text-decoration: none;
  background: #EEE;
  color: #222;
  border: 1px outset #CCC;
  padding: .1em .5em;
}
a.boton:hover {
  background: #CCB;
}
a.boton:active {
  border: 1px inset #000;
}
<a class="boton" href="#">Guardar</a>
<a class="boton" href="#">Enviar</a>
```

Crear un menú vertical

1. Definir anchura del menú
2. Eliminar las viñetas automáticas y todos los márgenes y espaciados aplicados por defecto
3. Añadir un borde al menú de navegación y establecer el color de fondo y los bordes de cada elemento del menú
4. Aplicar estilos a los enlaces: mostrarlos como un elemento de bloque para que ocupen todo el espacio de cada <li¿ del menú, añadir un espacio de relleno y modificar los colores y la decoración por defecto



Menú horizontal básico

1. Aplicar los estilos CSS básicos para establecer el estilo del menú (similares a los del menú vertical)
2. Establecer la anchura de los elementos del menú. Si el menú es de anchura variable y contiene cinco elementos, se asigna una anchura del 20 % a cada elemento
3. Establecer los bordes de los elementos que forman el menú
4. Se elimina el borde derecho del último elemento de la lista, para evitar el borde duplicado

Elemento 1	Elemento 2	Elemento 3	Elemento 4	Elemento 5

Bootstrap

¿Qué es Bootstrap?

- Bootstrap es un framework libre para desarrollo web
- Realizado por ingenieros de Twitter
- Incluye plantillas HTML y CSS con tipografías, formas, botones, cuadros, barras de navegación, carruseles de imágenes y muchas otras
- También existe la posibilidad de utilizar plugins de JavaScript
- Aunque su preferencia es *mobile first*, permite crear diseños que se ven bien en múltiples dispositivos (*responsive design*)

Ventajas de Bootstrap (para ingenieros)

- Es sencillo y rápido
- Se adapta a distintos dispositivos (*responsive design*)
- Proporciona un diseño consistente
- Es compatible con los navegadores modernos
- Es software libre

Ficheros de Bootstrap

- Se pueden descargar y servir como cualquier otro fichero estático
- O se pueden referenciar desde un CDN
  - CDN: Content Delivery Network
- Con un CDN (Content Delivery Network) no hace falta tener Bootstrap en nuestros archivos. Además, si un usuario ya ha descargado esas URLs, probablemente las tenga ya en la caché del navegador (con el consiguiente ahorro de tiempo).

La plantilla de Bootstrap

```
<!doctype html>
<html lang="en">
  <head>
    <!-- Required meta tags -->
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">

    <!-- Bootstrap CSS -->
    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.0/dist/css/boots

    <title>Hello, world!</title>
  </head>
  <body>
      <h1>Hello, world!</h1>

    <!-- Option 1: Bootstrap Bundle with Popper -->
    <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.0.0/dist/js/boots
</html>
```

Mobile first

- En los navegadores, el *viewport* es la parte visible de un documento
- Si el documento es mayor que el área de visualización, el usuario puede cambiar la vista desplazándose
- Con una propiedad de etiqueta meta, podemos indicar la escala inicial del *viewport*

```
<meta name="viewport" content="width=device-width, initial-scale=1">
```

- Se puede inhabilitar el zoom en dispositivos móviles con `user-scalable=no`
- Los usuarios sólo podrán hacer *scroll* y tendrá una apariencia nativa.
- Usar con precaución. No vale para todas las aplicaciones

```
<meta name="viewport" content="width=device-width, initial-scale=1,
maximum-scale=1, user-scalable=no">
```

Contenedores

- Bootstrap requiere tener un elemento contenedor que cubra contenidos y el sistema de rejilla.
- Para un contenedor responsivo de tamaño fijo, usa `.container`

```
<div class="container">
  ...
</div>
```

- Si se desea un contenedor con el ancho total (del *viewport*), se ha de usar `.container-fluid`

```
<div class="container-fluid">
  ...
</div>
```

El sistema de rejilla (I)

El diseño de páginas basado en rejilla se realiza mediante filas y columnas donde se colocan los contenidos. Así funciona la rejilla de Bootstrap:

- Filas, dentro un contenedor, agrupan horizontalmente varias columnas, que son las que tienen contenido
- La pantalla se divide en 12 columnas
- Las columnas definen su anchura especificando cuántas columnas de la fila ocupan
- Hay clases CSS (como por ejemplo .row y .col-xs-4) para crear rejillas rápidamente
- Hay padding entre columnas. En la primera y última columnas, las filas (elementos .row) aplican márgenes negativos

El sistema de rejilla (y II)

Three equal columns

Get three equal-width columns **starting at desktops and scaling to large desktops**. On mobile devices, tablets and below, the columns will automatically stack.

.col-md-4	.col-md-4	.col-md-4

Three unequal columns

Get three columns **starting at desktops and scaling to large desktops** of various widths. Remember, grid columns should add up to twelve for a single horizontal block. More than that, and columns start stacking no matter the viewport.

.col-md-3	.col-md-6	.col-md-3

Two columns

Get two columns **starting at desktops and scaling to large desktops**.

.col-md-8	.col-md-4

Responsive design

Responsive design en Bootstrap

Móviles y escritorio:

.col-xs-12 .col-md-8	.col-xs-6 .col-md-4

.col-xs-6 .col-md-4	.col-xs-6 .col-md-4	.col-xs-6 .col-md-4

.col-xs-6	.col-xs-6

Móvil, tableta y escritorio:

.col-xs-12 .col-sm-6 .col-lg-8	.col-xs-6 .col-lg-4

.col-xs-6 .col-sm-4	.col-xs-6 .col-sm-4	.col-xs-6 .col-sm-4

Breakpoints

Bootstrap incluye seis puntos de interrupción predeterminados, a veces denominados niveles de cuadrícula, para construir de manera *responsive*:

Breakpoint	**Class infix**	**Dimensions**
X-Small	None	$< 576px$
Small	sm	$>= 576px$
Medium	md	$>= 768px$
Large	lg	$>= 992px$
Extra large	xl	$>= 1200px$
Extra extra large	xxl	$>= 1400px$

Otros sitios de interés

- Listado de recursos sobre Bootstrap
  `http://bootsnipp.com/resources`
- Bootsnipp: Cientos de componentes adicionales
  `http://www.bootsnipp.com`
- Startboostrap: Plantillas y temas Bootstrap (gratis)
  `http://startbootstrap.com/`
- WrapBoostrap: Plantillas y temas Boostrap (de pago)
  `https://wrapbootstrap.com/`
- BootTheme: Generador (no libre) de diseños Bootstrap
  `http://www.boottheme.com/`