



SISTEMAS DISTRIBUIDOS BASADOS EN LA WORLD WIDE WEB

Sistemas informáticos I



2.2 WEB HIPERTEXTO

Sistemas distribuidos basados en la World Wide Web

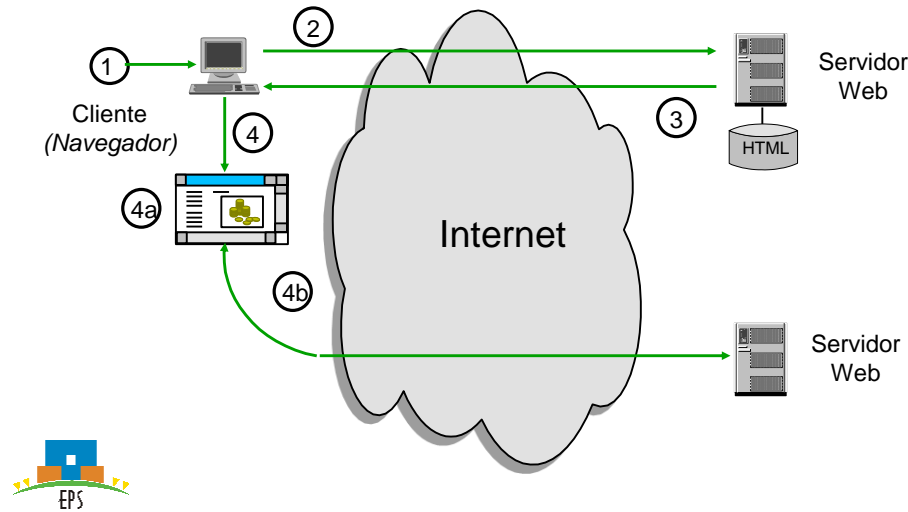


WEB HIPERTEXTO

- Es el modelo **original** y más sencillo de la Web
- Documentos distribuidos en servidores en Internet
 - Identificados por una URI
- Pueden contener otros objetos o documentos incrustados
- Enlaces desde cualquier documento a cualquier otro en la red
- El documento se define atendiendo a los elementos que contiene y no a la forma de presentarlo
 - Tendencia actual hacia una web semántica → metadatos y ontologías
 - Qué vs. Cómo
- Clientes estándar para visualizar documentos



FUNCIONAMIENTO BÁSICO



FUNCIONAMIENTO BÁSICO

1. El usuario solicita un recurso mediante su URL en un navegador (cliente)
2. El navegador genera una petición HTTP y la envía al servidor Web
3. El servidor Web recibe la petición y envía el recurso solicitado
 - Una sesión TCP por cada solicitud
 - Los documentos se codifican utilizando HTML
4. El cliente interpreta y muestra el documento recibido
 - Puede tener asociados nuevos elementos en su interior
 - Nueva petición HTTP a los servidores que los contienen para su recuperación
 - Originalmente nueva sesión TCP, el protocolo HTTP actual permite reutilizar sesiones



HTTP: HYPERTEXT TRANSFER PROTOCOL

- Protocolo de comunicaciones entre cliente y servidor Web
- Transporte: Conexión TCP sobre el puerto 80 (por defecto, 443 HTTPS)
- Intercambio de mensajes ASCII entre ambos:
 - Cliente realiza una petición

```
GET /hypertext/www/TheProject.html HTTP/1.0
```

- El servidor responde con un mensaje MIME (*Multipurpose Internet Mail Extensions*):

```
HTTP/1.0 200 Document follows
MIME-Version: 1.0
Server:CERN/3.0
Content-Type: text/html
Content-Length: 8247

<HEAD><TITLE>The World Wide Web Consortium (W3C)</TITLE></HEAD>
<BODY>
<H1><IMG ALIGN=MIDDLE ALT="W3C" SRC="icons/WWW(w3c_96x67.gif)"
The World Wide Web Consortium</H1><P>
...
```

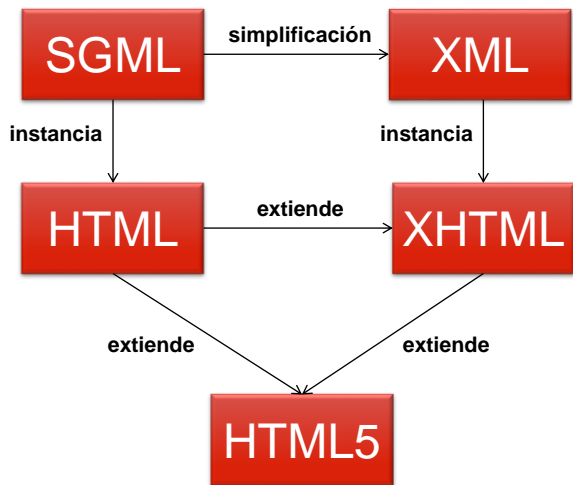


PETICIONES HTTP

- Formato de las peticiones: [método] URI [protocolo]
- URI: Identifica el objeto sobre el que aplicar el método
- Protocolo: El del mensaje enviado: HTTP/1.0, HTTP/1.1
- Método: Acción a realizar:
 - **GET**: Lectura del objeto
 - **HEAD**: Lectura de la cabecera del objeto
 - **PUT**: Almacenamiento del objeto
 - **POST**: Añadir al objeto la información enviada
 - **DELETE**: Eliminar el objeto
 - **LINK**: Conectar el objeto con otro determinado
 - **UNLINK**: Desconectar el objeto de otro determinado



FAMILIA DE LENGUAJES DE LA WWW



SGML: STANDARD GENERALIZED MARKUP LANGUAGE

- Estándar ISO-8879-1/1986 para la definición de texto electrónico independiente de dispositivos, sistemas y aplicaciones
- Proviene de GML (IBM, 70's)
- **Metalinguaje** para definir lenguajes de diseño descriptivos
- Proporciona un medio de codificar mensajes cuyo destino sea el intercambio directo entre sistemas o aplicaciones
- Múltiples lenguajes definidos mediante SGML
 - *Hypertext Markup Language, HTML*
- Evolución de SGML: *Extended Markup Language, XML*



SGML - CARACTERÍSTICAS

- SGML tiene cinco características principales:
 - Permite crear lenguajes de codificación descriptivos
 - Códigos incluidos en el propio texto definen los elementos que lo componen
 - Define una estructura de documento jerárquica, con elementos y componentes interconectados
 - Proporciona una especificación formal completa del documento
 - Contenida en la Document Type Definition, *DTD*
 - No tiene un conjunto implícito de convenciones de señalización. Soporta, por tanto, un conjunto flexible de juegos de etiquetas
 - SGML es un metalenguaje. Definición de etiquetas para cada lenguaje en su *DTD*
 - **Los documentos generados con él son legibles por personas**



EJEMPLO SGML – DTD DE HTML 4

```
<!-- Parameter Entities -->
<ENTITY % head.misc "SCRIPT | STYLE | META | LINK" -- repeatable head elements -->

<ENTITY % heading "H1 | H2 | H3 | H4 | H5 | H6">

<ENTITY % list "UL | OL | DIR | MENU">

<ENTITY % preformatted "PRE">

....

<ELEMENT FONT -- (%inline)* -- local change to font -->
<!ATTLIST FONT
  size      CDATA      #IMPLIED -- [+]nn e.g. size="+1", size=4 --
  color     CDATA      #IMPLIED -- #RRGGBB in hex, e.g. red: "#FF0000" --
  face      CDATA      #IMPLIED -- comma separated list of font names --
>
```



HTML: HYPERTEXT MARKUP LANGUAGE

- Aplicación del estándar SGML. <https://www.w3.org/MarkUp/>
- Lenguaje de definición de **formato** de documentos
 - Define los elementos que pueden aparecer dentro del documento
 - Se especifican mediante etiquetas (*Tags*) de comienzo y final de documento
 - Texto ASCII encerrado entre < y >
 - Sólo se define el tipo de elemento y no la forma de representarlo
- Lenguaje estándar utilizado en la Web para el intercambio de documentos hipermedia que incluyen:
 - Texto
 - Imágenes (Fotos, Vídeo)
 - Audio
 - Vínculos (Links)
- Estándar especificado por el *World Wide Web Consortium*, **W3C**



ESTRUCTURA DE UN DOCUMENTO HTML

- Identificación SGML
 - Permite identificar la DTD adecuada para procesarlo
- Delimitación del documento HTML
 - Identificado con la etiqueta <HTML>
- Cabecera. Información asociada al documento
 - Identificado por la etiqueta <HEAD>
 - Contiene información descriptiva del documento y definiciones de formato para los elementos del resto del mismo
- Cuerpo del documento
 - Identificado con la etiqueta <BODY>
 - Contiene los elementos de presentación del documento



EJEMPLO DE DOCUMENTO HTML

```
1 <html>
2   <head>
3     <title>
4       Ejemplo de página web
5     </title>
6   </head>
7
8   <body>
9     <div class="container">
10      <h1>Título de la página</h1>
11      <p>Primer párrafo...</p>
12    </div>
13  </body>
14 </html>
```

Diagrama de estructura de un documento HTML:

- Root Element <html>
 - Element <head>
 - Element <title>
 - Text "Ejemplo de página web"
 - Element <body>
 - Element <div> (Attribute "class")
 - Element <h1>
 - Text "Título de la página"
 - Element <p>
 - Text "Primer párrafo..."



<https://www.w3schools.com/html>



COMPONENTES DEL LENGUAJE HTML

- Etiquetas/Elementos
 - No vacías:
 - La mayoría de elementos comienzan por un tag, luego contenido, finalizando por un tag de terminación
`<TAG> contenido </TAG>`
 - Vacías: por ejemplo ``, `
`, etc.
- Atributos:
 - Específicos de cada etiqueta
 - Texto ASCII entre comillas (doble o simple) dentro de la etiqueta de comienzo
- Existen multitud de etiquetas que no vamos a ver en detalle, aquí solo vamos a tratar algunos (muy pocos) ejemplos representativos. Consultar el manual de referencia de el W3C
- HTML es insensible a Mayúsculas/Minúsculas:
 - `<TAG>` es equivalente a `<tag>` o `<TaG>`
- Espacios en blanco, tabulaciones y retornos de carros no son significativos



FRAMES (MARCOS)

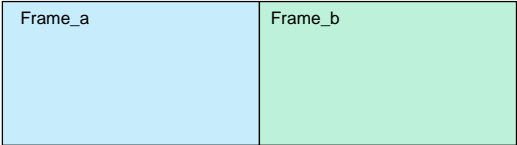
- Con los frames (iframes en HTML5) varios documentos HTML pueden ser mostrados en la misma ventana del navegador
- Cada frame es independiente de los otros
- Han caído en desuso: seguridad y AJAX



EJEMPLO DE *IFRAMES*

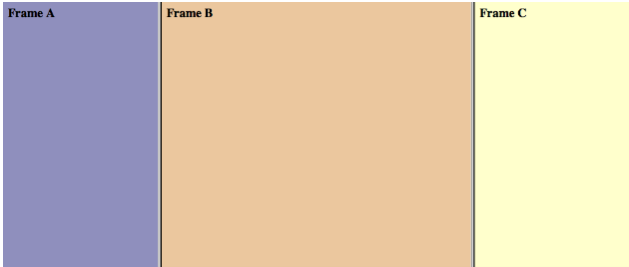
```
<html>
  <body>
    <p>Ejemplo de página con iframes</p>
    <table width="100%">
      <tr>
        <td>
          <iframe src="frame_a.htm" height="200px" width="100%"></iframe>
        </td>
        <td>
          <iframe src="frame_b.htm" height="200px" width="100%"></iframe>
        </td>
      </tr>
    </table>
  </body>
</html>
```

Ejemplo de página con iframes



EJEMPLO DE *FRAMES*

```
<html>
  <frameset cols="25%,50%,25%">
    <frame src="frame_a.htm" />
    <frame src="frame_b.htm" />
    <frame src="frame_c.htm" />
  </frameset>
</html>
```



LAYOUT EN HTML

- Dos formas de organizar dónde va cada cosa:
 - Tablas
 - Etiqueta <div>
 - CSS



Main Title of Web Page	
Menu HTML CSS JavaScript	Content goes here
Copyright © 2011 W3Schools.com	

LAYOUT UTILIZANDO <TABLE>


- El único disponible en las primeras versiones de HTML
- Solía combinarse con el uso de frames, actualmente es más habitual el uso de AJAX
- Desventajas:
 - Obliga a crear ficheros muy grandes y complejos
 - Obliga a descargar información de presentación con cada página que se descarga
 - El rediseño de sitios y contenido existente es más laboriosos (y costoso)
 - Es difícil (y costoso) mantener consistencia visual en un sitio
 - Las páginas basadas en tablas son menos accesibles a usuarios con necesidades especiales y a los que usan teléfonos móviles, PDAs o tabletas para acceder




LAYOUT UTILIZANDO <TABLE>

```
<html>
<body>

<table width="500" border="0">
<tr>
<td colspan="2" style="background-color:#FFA500; height: 200px; width: 400px; text-align: top;">
<h1>Main Title of Web Page</h1>
</td>
</tr>
<tr>
<td style="background-color:#FFD700; width: 100px; text-align: top;">
<b>Menu</b><br />
HTML<br />
CSS<br />
JavaScript
</td>
<td style="background-color:#EEEEEE; height: 200px; width: 400px; text-align: top;">
Content goes here</td>
</tr>
</table>
</body>
</html>
```





LAYOUT UTILIZANDO <DIV>

```
<html>
<body>

<div id="container" style="width: 500px">


<div id="header" style="background-color: #FFA500;">
<h1 style="margin-bottom: 0;">
Main Title of Web Page</h1></div>

<div id="menu" style="background-color: #FFD700; height: 200px; width: 100px; float: left;">
<b>Menu</b><br />
HTML<br />
CSS<br />
JavaScript</div>

<div id="content" style="background-color: #EEEEEE; height: 200px; width: 400px; float: left;">
Content goes here</div>

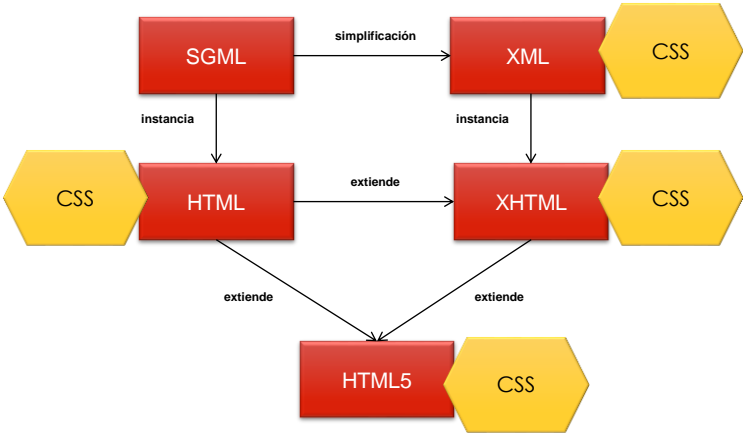
<div id="footer" style="background-color: #FFA500; clear: both; text-align: center;">
Copyright © 2011 W3Schools.com</div>

</div>
</body>
</html>
```



http://www.w3schools.com/tags/tag_div.asp

CSS: CASCADE STYLE SHEETS



SEPARACIÓN DE RESPONSABILIDADES

- Una regla fundamental del diseño de software es que funciones distintas deben ser llevadas a cabo por entidades distintas
 - *Separation of concerns*
- HTML originalmente fue diseñado para definir contenido de documentos...

```
<h1>This is a heading</h1>
<p>This is a paragraph.</p>
```
- ... pero no para definir formato, la forma en que se presentan los contenidos
 - Cuando se agregaron etiquetas como `` (a partir de HTML 3.2) la cosa empezó a complicarse
 - Se mezcla el contenido (el qué) con el formato (el cómo)!!!



CSS: CASCADE STYLE SHEETS

- Usado para describir la semántica de presentación de un documento escrito con un lenguaje de marcado
 - El uso más común es darle estilo a páginas escritas en HTML y XHTML
- El objetivo es separar el contenido del documento (HTML o similar) de cómo se presenta (layout, colores, fuentes, etc.)
- Ventajas:
 - Forma rápida y sencilla de cambiar el aspecto del documento
 - Ofrecer accesibilidad
 - Más control y flexibilidad de presentación
 - Compartir formato entre múltiples páginas
 - Distinta presentación para distinto tipo de dispositivo (ordenador, móvil, impreso...) o entre distinto tipo de usuarios de a aplicación (p.ej., usuarios de distintas compañías)




Heading 1

This is some text in a paragraph.

This is another paragraph.

Heading 2

Name	E-mail	Phone
Doe, John	jdoe@example.com	555-789-7222
Smith, Eva	esmith@example.com	555-324-3693



http://www.w3schools.com/Css/demo_default.htm

Heading 1

This is some text in a paragraph.

This is another paragraph.

Heading 2

Name	E-mail	Phone
Doe, John	jdoe@example.com	555-789-7222
Smith, Eva	esmith@example.com	555-324-3693

Heading 1

This is some text in a paragraph.

This is another paragraph.

Heading 2

Name	E-mail	Phone
Doe, John	jdoe@example.com	555-789-7222
Smith, Eva	esmith@example.com	555-324-3693

Heading 1

This is some text in a paragraph.

This is another paragraph.

Heading 2

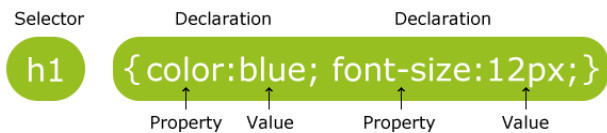
Name	E-mail	Phone
Doe, John	jdoe@example.com	555-789-7222
Smith, Eva	esmith@example.com	555-324-3693

SINTAXIS CSS

- Un estilo CSS es una combinación de propiedades predefinidas a las que se asigna un valor (*propiedad:valor*). Cada declaración separada por “;”
- *Estilos inline* (mala práctica, solo en casos muy específicos): aplican directamente sobre un elemento del documento

```
<p style="color:sienna;margin-left:20px">This is a paragraph.</p>
```

- Lo más habitual es definir hojas de estilos compuestas por reglas de aplicación
- Las reglas constan de un selector y las declaraciones del estilo entre llaves



- Se pueden definir:
 - Hojas de estilo internas: en el propio documento como contenido del tag <style>
 - Hojas de estilo externas: en ficheros externos vinculados al documento



EJEMPLO CSS

```
<html>
<head>
<link rel="stylesheet" type="text/css"
href="ex2.css" />
</head>

<body>

<h1>This is a header 1</h1>
<hr />

<p>You can see that the style
sheet formats the text</p>

<p><a href="http://www.w3schools.com"
target="_blank">This is a link</a></p>

</body>
</html>
```

```
body {background-color:tan;}
h1 {color:maroon;font-size:20pt;}
hr {color:navy;}
p {font-size:11pt;margin-left:15px;}
a:link {color:green;}
a:visited {color:yellow;}
a:hover {color:black;}
a:active {color:blue;}
```

This is a header 1

You can see that the style sheet formats the text

[This is a link](#)

TIPOS DE SELECTORES

- Selector de elemento
 - Usado para especificar el estilo de los elementos de un tipo determinado
 - Se define con el nombre del elemento
- Selector *id*
 - Usado para especificar el estilo de un elemento con un *id* determinado
 - Usa el atributo *id* de los elementos HTML y se define con "#"
- Selector *class*
 - Permite especificar un estilo para todos los elementos de una determinada "clase"
 - Utiliza el atributo *class* de HTML y se define con "."
- Combinación de selectores:
 - Reglas de selección en función de la relación entre selectores:
 - Espacio: selector de descendientes
 - >: selector de hijos
 - ~: selector de hermanos
 - +: selector de hermanos adyacentes

https://www.w3schools.com/css/css_combinators.asp



EJEMPLO DE SELECTOR ID

```
<html>
<head>
<style type="text/css">
#para1 {
text-align:center;
color:red;
}
</style>
</head>

<body>
<p id="para1">Hello World!</p>
<p>This paragraph is not affected by the style.</p>
</body>
</html>
```



Hello World!

This paragraph is not affected by the style.



EJEMPLO SELECTOR CLASS

```
<html>
<head>
<style type="text/css">
.center {
text-align:center;
}
</style>
</head>

<body>
<h1 class="center">Center-aligned heading</h1>
<p class="center">Center-aligned paragraph.</p>
</body>
</html>
```



Center-aligned heading

Center-aligned paragraph.

CASCADA...

- ... lo que por fin explica el nombre
- Sobre un elemento determinado pueden aplicar distintos estilos definidos en distintos "sitios"
- Se impone un orden de precedencia que determina qué selector prevalece según tres criterios (de mayor a menor):
 - Importancia (!important)
 - Especificidad:
 - Estilo *inline*
 - Selector id
 - Selector class
 - Selector de elemento
 - Orden

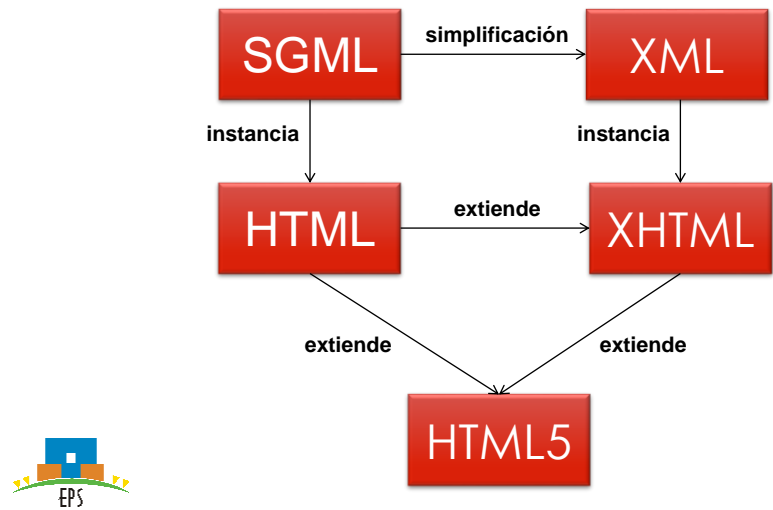


EJERCICIO

- Hacer una **página web** personal **estática**, recomendablemente añadiendo hojas de estilo CSS, que al menos incluya los siguientes elementos:
 - Un enlace
 - Una imagen (no es necesario que sea de una altísima calidad)
 - Una tabla
- Entregar de forma individual a través de Moodle



XML: EXTENSIBLE MARKUP LANGUAGE



XML: EXTENSIBLE MARKUP LANGUAGE

- Especificación del W3C (www.w3.org/XML) para crear lenguajes descriptivos de propósito específico
 - Fecha de primera especificación: 1998
 - <https://www.w3.org/TR/1998/REC-xml-19980210>
 - https://www.w3schools.com/xml/xml_what.asp
 - XML is a software- and hardware-independent tool for storing and transporting data
 - **XML Does Not DO Anything**
- Procede de SGML reduciendo su complejidad
- Se puede ver como una generalización de HTML para modelar estructuras de datos de cara al procesamiento automático de información
 - Lenguaje de meta-marcado
 - Diseñado para describir estructuras de datos
 - Utilizado para realizar intercambios de datos entre sistemas
 - Etiquetas, elementos y atributos con estructura de árbol
- En los últimos años hay una tendencia a sustituir XML por JSON



EJEMPLO DE DOCUMENTO XML

Root

```
<Person birth = "1912" death = "1954">
  <name>
    <firstName> Alan </firstName>
    <lastName> Turing </lastName>
  </name>
  <profession> computer scientist </profession>
  <profession> mathematician </profession>
  <profession> cryptographer </profession>
</Person>
```

--Atributo

--Elemento



SINTAXIS XML

- El documento XML es de texto, normalmente *Unicode*
- Cada documento XML debe tener un único elemento raíz
- Todo elemento no vacío debe tener etiqueta de comienzo (`<elemento>`) y final (`</elemento>`)
- Todo elemento que no lleve contenido puede ser marcado con una etiqueta de elemento vacío (`<elemento/>`)
- Todos los valores de atributos deben ir entre comillas dobles
- Los elementos se pueden anidar, pero no solapar con otros
- Los nombres de los elementos son sensibles a mayúsculas y minúsculas
- Permite el uso de distintos espacios de nombres tomados de distintos vocabularios en el mismo documento (`namespace:nombre`)



VALIDACIÓN CON DTD

- *Document Type Definition*
- No es obligatorio
- Especifican un sublenguaje de XML
 - Lista de etiquetas permitidas
 - Etiquetas y atributos permitidos dentro de cada etiqueta
- Pueden ser muy complejos
 - Para SVG (Scalable Vector Graphics) más de 1000 líneas
 - Para XHTML 1.0 más de 1500 líneas
 - Para DocBook más de 11000 líneas
- Alternativa: XML Schema (versión 1.0, mayo 2001)

https://www.w3schools.com/xml/xml_dtd_examples.asp



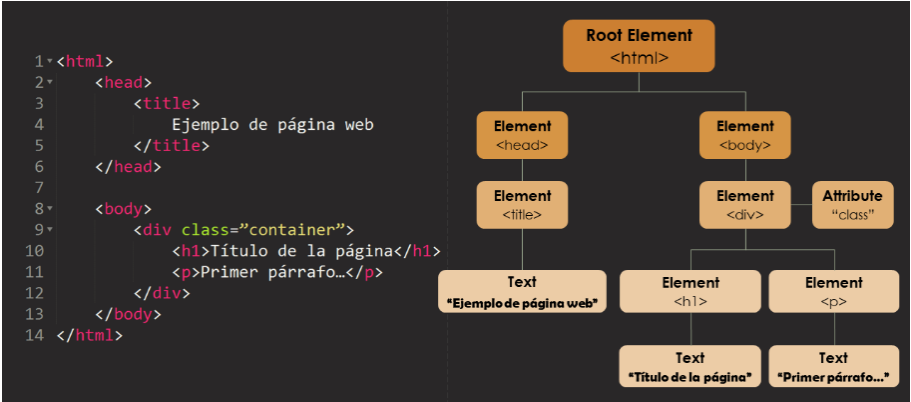
PARSEO DE DOCUMENTOS XML

SAX: *Simple API for XML*

- Orientado a eventos
- El programador proporciona métodos callback que son invocados por el parser a medida que lee el documento XML
- Es rápido y eficiente, pero difícil de usar
 - Cada nodo es tratado independientemente de lo que había antes o después
 - No se puede "volver atrás" en el procesado
- DOM: *Document Object Model*
 - Los documentos se estructuran como árboles
 - También sirve para otros lenguajes como HTML
 - Estándar del W3C
 - XML DOM:
 - Modelo de objetos estándar para XML
 - Una API estándar para XML, aunque no todos los parser DOM siguen el estándar
 - Independiente de la plataforma y del lenguaje de programación



ÁRBOL DOM



API DOM

- Todo en DOM son nodos:
 - El documento completo es un document node
 - Cada elemento XML es un element node
 - El texto en elementos XML son text nodes
 - Cada atributo es un attribute node
 - Los comentarios son comment nodes
- La API DOM define una serie de propiedades y métodos que debe ofrecer cualquier implementación de DOM para acceder y manipular nodos en las distintas dimensiones del árbol DOM
 - x.nodeName: el nombre de x
 - x.nodeValue: el valor de x
 - x.parentNode: el nodo padre de x
 - x.childNodes: los nodos hijos de x
 - x.attributes: los nodos atributos de x

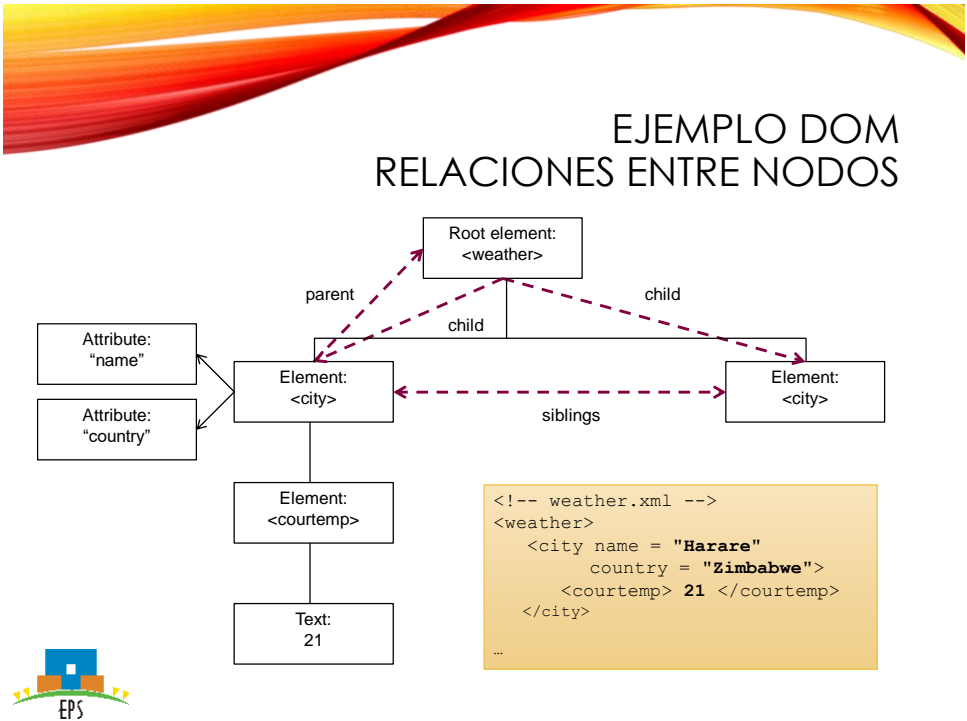


XPATH

- XPATH es un mecanismo para definir expresiones para navegar por las distintas dimensiones del árbol DOM
 - En cierta medida análogo a la definición de rutas en un sistema de archivos
- Recomendación del W3C independiente del lenguaje de programación
 - Hay implementaciones en múltiples lenguajes
 - En particular, es uno de los principales elementos de XSLT
- Distingue entre 7 tipos de nodos:
 - Element
 - Attribute
 - Text
 - Comment
 - Document
 - Namespace
 - Processing-instruction
- Relaciones entre nodos:
 - Parent
 - Children
 - Siblings
 - Ancestors
 - Descendants



https://www.w3schools.com/xml/xml_xpath.asp



EJEMPLOS XPATH

- /bookstore/book[1]
- /bookstore/book[last()]
- /bookstore/book[last()-1]
- /bookstore/book[position()<3]
- //title[@lang]
- //title[@lang='en']
- /bookstore/book[price>35.00]
- /bookstore/book[price>35.00]/title



XSL: EXTENSIBLE STYLESHEET LANGUAGE

- CSS dice cómo mostrar un documento HTML
 - HTML tiene un conjunto predefinido de etiquetas
 - Decirle a un navegador cómo mostrar un elemento en una fuente o color especial es sencillo
- XML en cambio permite definir cualquier etiqueta
 - ¿Qué significa TABLE?
 - ¿Cómo debe mostrarlo el cliente?



- XSL le dice al navegador cómo se debe mostrar el documento XML
- Una plantilla XSL es un documento XML



EJEMPLO DE TRANSFORMACIÓN XSL

transformacion.xsl

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0"
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:template match="/">

<html>
<head>
  <title>
    Ejemplo de transformacion - Texto de plantilla
  </title>
</head>
<body>
  <h1>Hola</h1>
  <ul>
    <xsl:for-each select="helloworld/person">
      <li>Hello <xsl:value-of select="name"/></li>
    </xsl:for-each>
  </ul>
</body>
</html>
</xsl:template>
</xsl:stylesheet>
```

datos.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE helloworld SYSTEM "dtd-datos.dtd">
<?xml-stylesheet type="text/xsl"
  href="transformacion.xsl" ?>

<helloworld>
  <person>
    <name>Pedro</name>
  </person>
  <person>
    <name>Juan</name>
  </person>
  <person>
    <name>Miguel</name>
  </person>
</helloworld>
```

dtd-datos.dtd

```
<!DOCTYPE helloworld
[
  <!ELEMENT helloworld (person)>
  <!ELEMENT person (name)>
  <!ELEMENT name (#PCDATA)>
]>
```





JSON: JAVASCRIPT OBJECT NOTATION

- Según el W3C:
 - JSON is a syntax for storing and exchanging data
 - JSON is text, written with *JavaScript* object notation
- Muy popular: a día de hoy es seguramente el formato más utilizado para intercambiar datos entre aplicaciones web
- Es legible por personas e interpretable por algoritmos
- Almacena los datos siguiendo la notación de **objetos** y **arrays JS**



```
{
  "artists" : [
    {
      "artistname" : "Deep Purple",
      "albums" : [
        {
          "albumname" : "Machine Head",
          "year" : "1972",
          "genre" : "Rock"
        },
        {
          "albumname" : "Stormbringer",
          "year" : "1974",
          "genre" : "Rock"
        }
      ]
    }
  ]
}
```



SINTAXIS JSON

- Un objeto JS es un conjunto desordenado de pares nombre/valor entre llaves {}
 - El nombre es una cadena de caracteres que identifica el atributo
 - El valor puede ser un objeto, un array, un número, cadena de caracteres, true, false o null



```
{ }

{ "artistname" : "Pink Floyd" }

{
  "artistname" : "Pink Floyd",
  "formed" : "1964"
}

{
  "artistname" : "Pink Floyd",
  "formed" : "1964",
  "origin" : "Cambridge, United Kingdom"
}
```


SINTAXIS JSON

- Un array es una colección ordenada de valores
- Permite construir listas de valores



```
{
  "artists" : [
    {
      "artistname" : "Pink Floyd",
      "formed" : "1964"
    },
    {
      "artistname" : "Bruce Springsteen",
      "born" : "1949"
    },
    {
      "artistname" : "U2",
      "formed" : "1976"
    }
  ]
}
```

DATOS ANIDADOS



```
{
  "artists" : [
    {
      "artistname" : "Pink Floyd",
      "formed" : "1964",
      "albums" : [
        {
          "albumname" : "The Dark Side of the Moon",
          "year" : "1973",
          "genre" : "Rock"
        },
        {
          "albumname" : "Wish You Were Here",
          "year" : "1975",
          "genre" : "Rock"
        }
      ]
    }
  ]
}
```

XML VS. JSON

```
<employees>
  <employee>
    <firstName>John</firstName> <lastName>Doe</lastName>
  </employee>
  <employee>
    <firstName>Anna</firstName> <lastName>Smith</lastName>
  </employee>
  <employee>
    <firstName>Peter</firstName> <lastName>Jones</lastName>
  </employee>
</employees>

{"employees": [
  { "firstName": "John", "lastName": "Doe" },
  { "firstName": "Anna", "lastName": "Smith" },
  { "firstName": "Peter", "lastName": "Jones" }
]}
```



https://www.w3schools.com/js/js_json_xml.asp

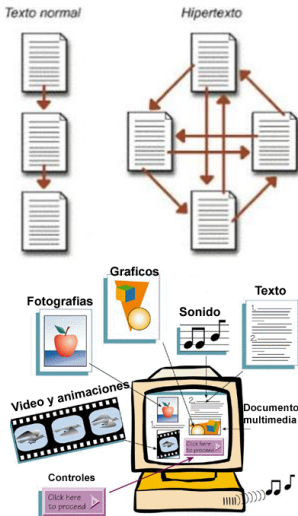
XML VS. JSON

- En principio, XML y JSON sirven para lo mismo
- Sus diferencias son:
 - JSON no utiliza tags
 - JSON es más corto
 - JSON es más rápido de leer y escribir
 - JSON utiliza arrays
 - XML debe ser parseado por un parser (tradicionalmente DOM), mientras que JSON puede serlo por el interprete JavaScript como objetos y arrays JS "listos para utilizar"
- Los mecanismos de validación XML
 - <https://json-schema.org/>
 - Actualmente en versión draft





VISIÓN HISTÓRICA



VISIÓN HISTÓRICA



<http://getbootstrap.com/>

