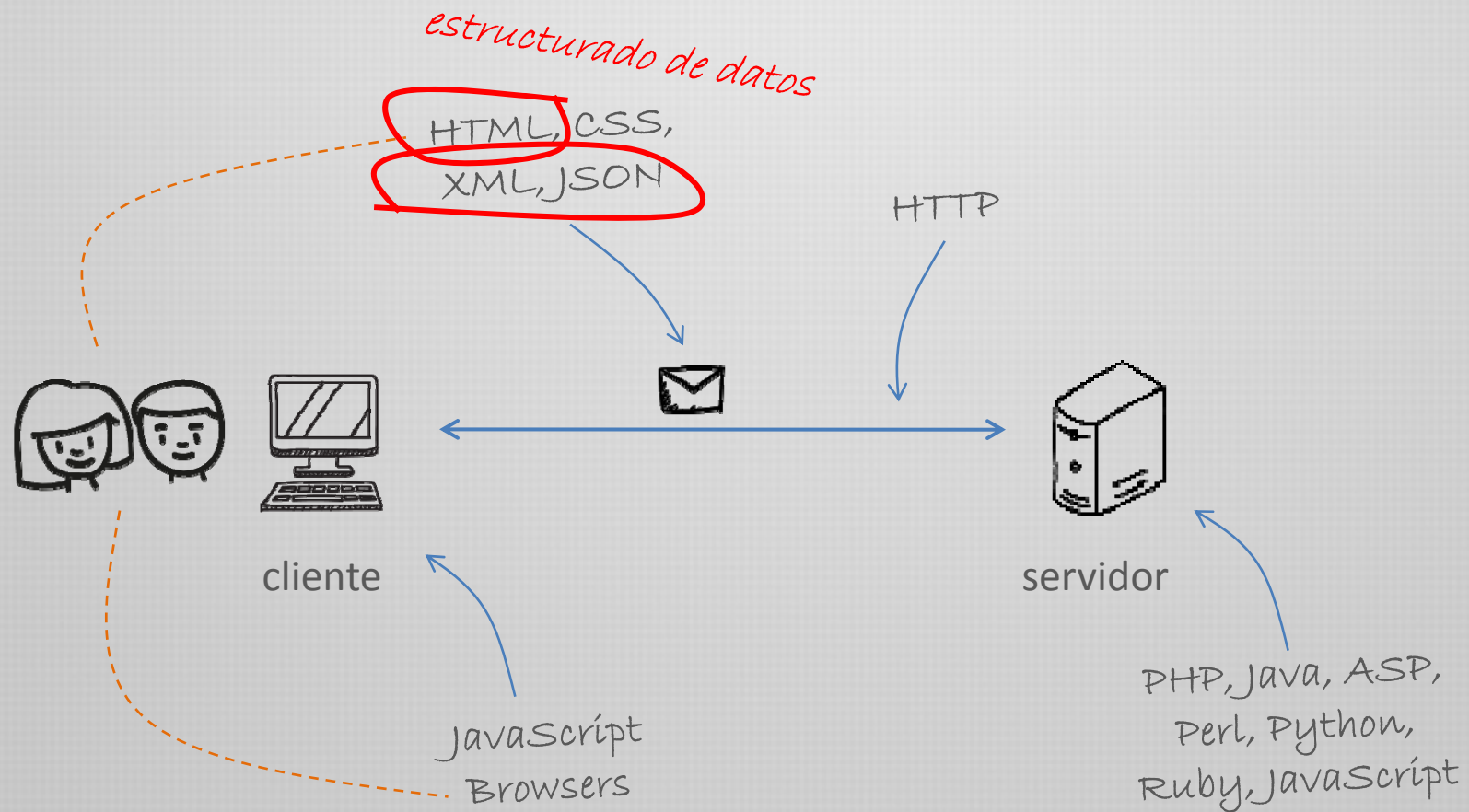


Ingeniería de Aplicaciones Web

Diego C. Martínez

Departamento de Ciencias e Ingeniería de la Computación
Universidad Nacional del Sur

Tecnologías web



XML

XML es el **lenguaje de marcado extensible**. (*eXtensible Markup Language*)

Es extensible porque los tags son definibles por el usuario

- Es la base de la *interoperabilidad* de muchos sistemas.
- Es el estándar para muchas tecnologías web (Ajax, Servicios web, etc)
- Es una recomendación de la W3C desde 1998.

La idea central detrás de XML es proveer una forma de

- **estructurar** información general,
- con **independencia** de la plataforma.

Facilita intercambio, procesamiento, claridad de datos.

```
<materia>
  <curricular/>
  <nombre>Historia</nombre>
  <profesor>
    Horacio Montesano
  </profesor>
  <horarios>
    <horario dia="lunes" desde="8" hasta="10"/>
    <horario dia="martes" desde="8" hasta="10"/>
  </horarios>
</materia>
```

XML - historia

XML no es el primer lenguaje de marcado.

GML
IBM - 1973

Generalized Markup Language
Goldfarb, Mosher, Lorie



SGML
ANSI - 1986

Standard Generalized Markup Language
Un meta-lenguaje para definir lenguajes de marcas.



HTML
W3C - 1990

Características distintivas:

- *Enfasis en marcado descriptivo, no procedural*
- *Concepto de tipo de documento*
- *Independencia del sistema*

XML - historia

XML nace entre la complejidad de SGML y las limitaciones de HTML

Los objetivos principales son:

- 1. XML debe usarse sin complicaciones sobre Internet.*
- 2. XML debe soportar una gran variedad de aplicaciones.*
- 3. XML debe ser compatible con SGML.*
- 4. Debe ser fácil escribir programas que procesan documentos XML.*
- 5. El número de características opcionales en XML debe mantenerse al mínimo.*
- 6. Los documentos XML deben ser legibles por el humano y razonablemente claros.*
- 7. El diseño XML debería ser preparado rápidamente.*
- 8. El diseño de XML debe ser formal y conciso.*
- 9. Los documentos XML deben ser fáciles de crear.*
- 10. El laconismo es de importancia mínima.*

XML - caracteres

XML utiliza caracteres Unicode (conjunto de caracteres de 32 bits <http://www.unicode.org>).
Principalmente UTF-8.

Algunos caracteres no pueden usarse literalmente.

Character reference ----> codificación del caracter literal por su equivalente numérico

A = A © = © é = é

Caracteres obligatoriamente no literales:

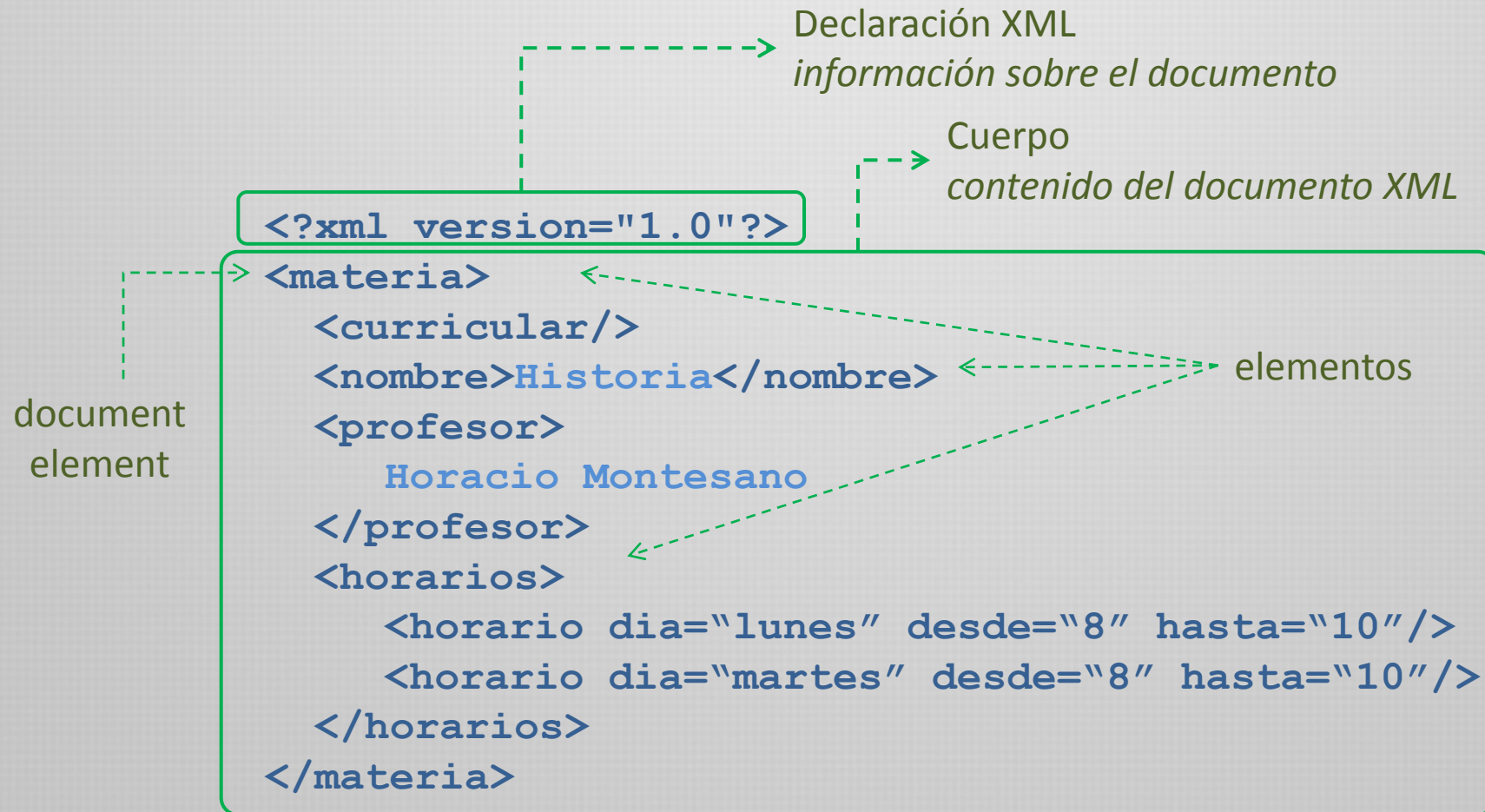
< < <
& & &
> > >
" " "
' ' '

Algunos caracteres no visibles tienen una importancia especial en XML...

<i>Whitespaces</i>	----	space	
		tab		
		carriage return	
		line feed	

XML es case-sensitive.

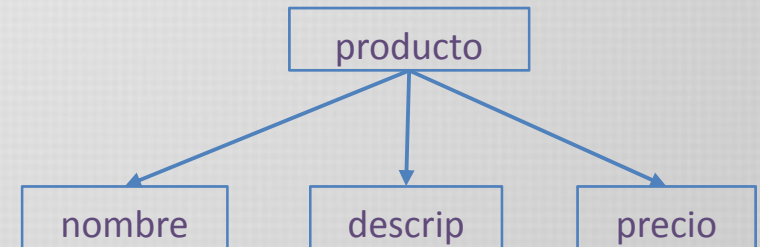
XML - anatomía



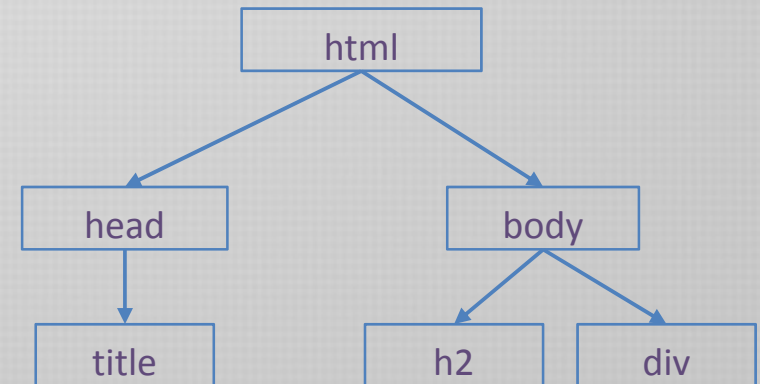
Estructurando documentos

HTML, XML y XHTML son estándares orientados principalmente a la representación y estructuración de información con algún fin particular

```
<producto>
  <nombre>PS3</nombre>
  <descrip>Buenisima</descrip>
  <precio>3000</precio>
</producto>
```



```
<html>
  <head>
    <title></title>
  </head>
  <body>
    <h2>Blog</h2>
    <div>Bla bla bla</div>
  </body>
</html>
```



HTML incluye algunos tags con semántica de presentación incorporada

JSON - una alternativa

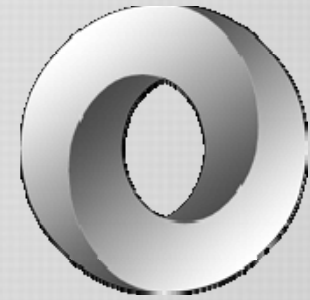
JSON = JavaScript Object Notation

Es un formato de intercambio de datos.

Minimal y textual.

Es una popular alternativa a XML.

*Es un **formato**, no un lenguaje*



Basado en un subconjunto de JavaScript, Standard ECMA-262

A veces denominados inicializadores en JavaScript

Usa convenciones de la familia de lenguajes C.

C, C++, Java, C#, JavaScript, Perl, PHP, Python, ActionScript.

No es un formato de serialización de datos.

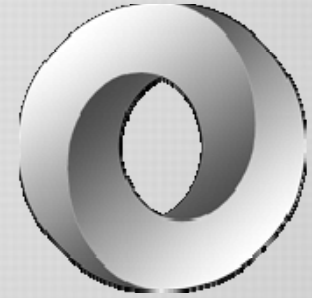
No admite estructuras cíclicas ni la inclusión de funciones

json.org

JSON

JSON es un buen formato de intercambio de datos

- Es fácilmente **interpretable** por el humano y por las computadoras
- **Poca decoración** a los datos, lo que facilita su transporte e interpretación algorítmica.
- Soporte *Unicode*, lo que lo hace **universal** en su contenido
- Representación de **estructuras de datos elementales**.
- Valores simples universales
- Facilidad de mapeo de estructuras nativas a notación JSON.



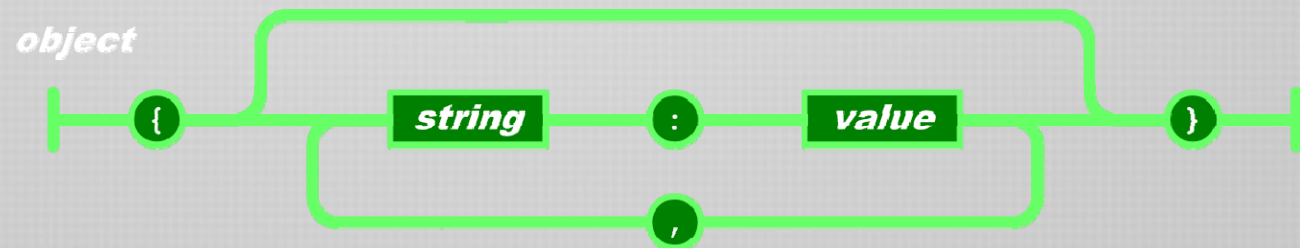
JSON

Consiste básicamente de dos estructuras de datos *universales*:

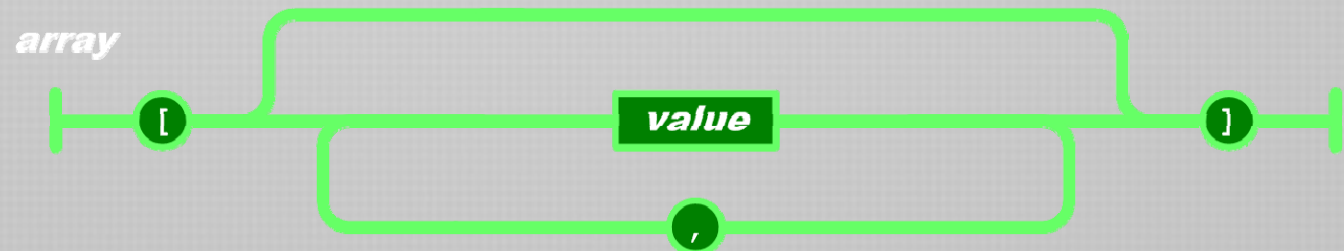
- Colección de pares nombre-valor
En muchos lenguajes: objetos, records, structs, etc
- Lista ordenada de valores
En muchos lenguajes: listas, arreglos, vectores, etc

En JSON, estas estructuras son:

- Objetos

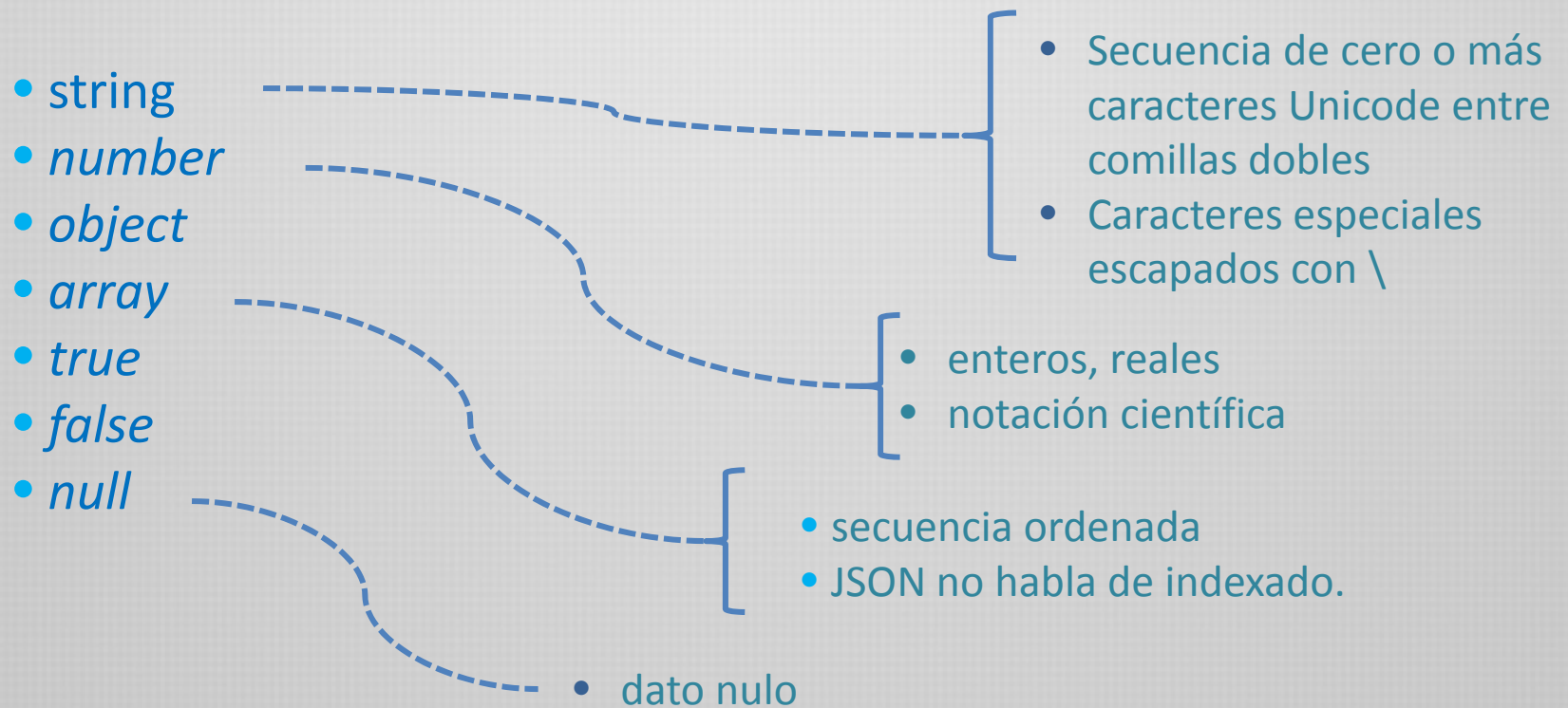


- Arrays



JSON

Un valor JSON puede ser



```
var unObjeto = {"id": 1234, "nombre": "Juan  
Fulano", "alumnoRegular": true, "ultimosCursos": [7145, 7854, 7321  
], "origen": {"ciudad": "Bahia Blanca", "provincia": "Buenos Aires"}}
```

JSON - arrays

```
var arreglo = [ "Argentina",  
                "Brasil",  
                "Chile",  
                "Uruguay",  
                "Ecuador",  
                "Colombia"  
              ]
```

```
var horarios = [  
  ["martes", "jueves"],  
  [14, 14],  
  [18, 18]  
]
```


JSON - Objetos

```
var unAlumno = {  
  "id": 1234,  
  "nombre": "Juan Fulano",  
  "alumnoRegular": true,  
  "ultimosCursos": [  
    7145,  
    7854,  
    7321  
  ],  
  "origen": {  
    "ciudad": "Bahia Blanca",  
    "provincia": "Buenos Aires"  
  }  
}
```

JSON - Objetos

```
var unAlumno = {  
  "id": 1234,  
  "nombre": "Juan Fulano",  
  "alumnoRegular": true,  
  "ultimosCursos": [  
    7145,  
    7854,  
    7321  
  ],  
  "origen": {  
    "ciudad": "Bahia Blanca",  
    "provincia": "Buenos Aires"  
  }  
}
```

Los campos de los
objetos van entre
comillas dobles

JSON - Objetos

```
var unAlumno = {  
  "id": 1234,  
  "nombre": "Juan Fulano",  
  "alumnoRegular": true,  
  "ultimosCursos": [  
    7145,  
    7854,  
    7321  
  ],  
  "origen": {  
    "ciudad": "Bahia Blanca",  
    "provincia": "Buenos Aires"  
  }  
}
```

Los valores son
marcados de
acuerdo a su tipo

JSON - Objetos

```
var unAlumno = {  
  "id": 1234,  
  "nombre": "Juan Fulano",  
  "alumnoRegular": true,  
  "ultimosCursos": [  
    7145,  
    7854,  
    7321  
  ],  
  "origen": {  
    "ciudad": "Bahia Blanca",  
    "provincia": "Buenos Aires"  
  }  
}
```

Un arreglo de enteros

JSON - Objetos

```
var unAlumno = {  
  "id": 1234,  
  "nombre": "Juan Fulano",  
  "alumnoRegular": true,  
  "ultimosCursos": [  
    7145,  
    7854,  
    7321  
  ],  
  "origen": {  
    "ciudad": "Bahia Blanca",  
    "provincia": "Buenos Aires"  
  }  
}
```

Un objeto con dos
campos

JSON - Objetos

```
var unAlumno = {
    "id": 1234,
    "nombre": "Juan Fulano",
    "alumnoRegular": true,
    "ultimosCursos": [
        7145,
        7854,
        7321
    ],
    "origen": {
        "ciudad": "Bahia Blanca",
        "provincia": "Buenos Aires"
    }
}
```

`unAlumno.id`
`unAlumno.nombre`
`unAlumno.ultimosCursos[0]`
`unAlumno.origen.ciudad`

MIME Media Type

Como JSON se utiliza como **formato de transmisión de datos** serializados, es necesario indicar su **tipo** cuando es transmitido (HTTP messages)

Content-Type: -----> **application/json**

A veces llamado **MIME Type**

-----> ***Multipurpose Internet Mail Extensions***

Inicialmente referido al protocolo SMTP y la transmisión de datos no-ASCII agregados al mail.

Hoy propagado a otros protocolos.

Interpretando JSON

El lenguaje utilizado debe ser capaz de comprender y generar estructuras JSON.

Los *decoders* aceptan descripciones JSON bien formadas y las traducen a estructuras de datos nativas del lenguaje.

Los *encoders* traducen estructuras nativas a expresiones JSON.

JavaScript es un *decoder* natural de JSON :)
No se emplea tiempo de parsing ni transformación

Otros lenguajes ofrecen librerías para operar con datos JSON.

JSON

JSON no posee **validadores**, a diferencia de XML.

JSON permite una flexibilidad mayor que XML en algunos aspectos

Permite agregar campos a una estructura sin perturbar el código existente

JSON ha demostrado ser eficiente en *el intercambio de datos entre servidores y clientes*, con tecnologías heterógeneas

Estructura vs. presentación

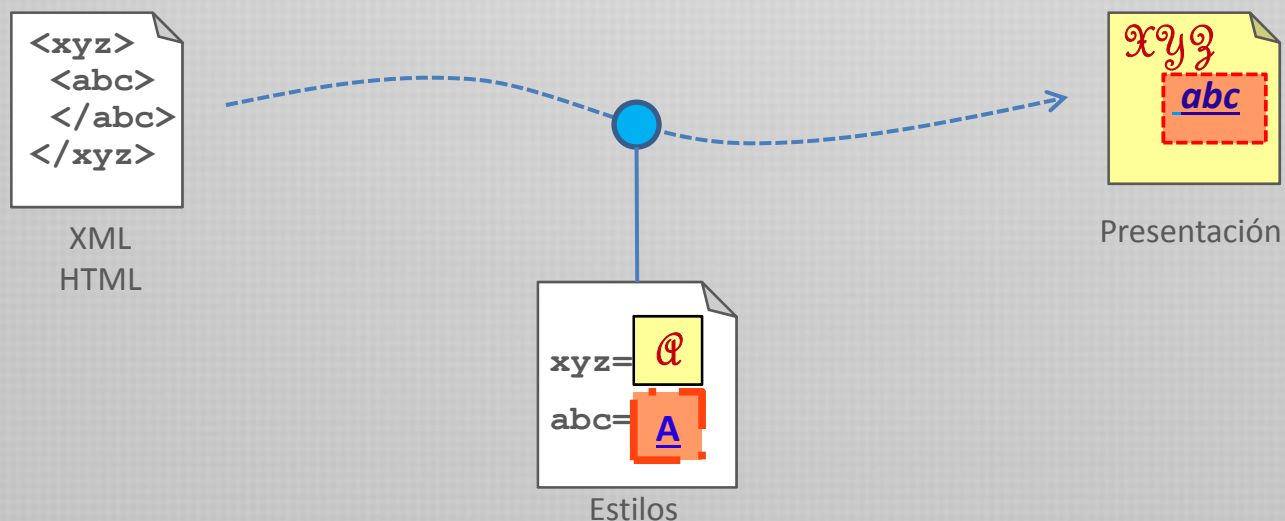
La W3C presenta *algunas* guías para la visualización de documentos HTML, basada en la interpretación de algunos tags.

- un elemento `strong` se verá en font **bold**
- un elemento `h1` se verá de mayor tamaño que uno de `h2`.

El estándar XML no incluye nada de aspectos de presentación , pues es un lenguaje de marcado de propósito general.

Tal vez no sea información destinada a la visualización.

Sin embargo, cuando estructuramos información destinada (de alguna manera u otra) a una interfaz visual, es deseable poder estilizarla adecuadamente.



Estilos

Existen dos mecanismos para agregar estilos a documentos de tags.

- *CSS – Cascading Style Sheets*

- Documento de texto con sintaxis simple.
- Es aplicable a HTML y a XML.
- Permite aplicar formato a elementos (tags) – por tipo, nombre o identificador
- Permite heredar formatos de elementos contenedores.

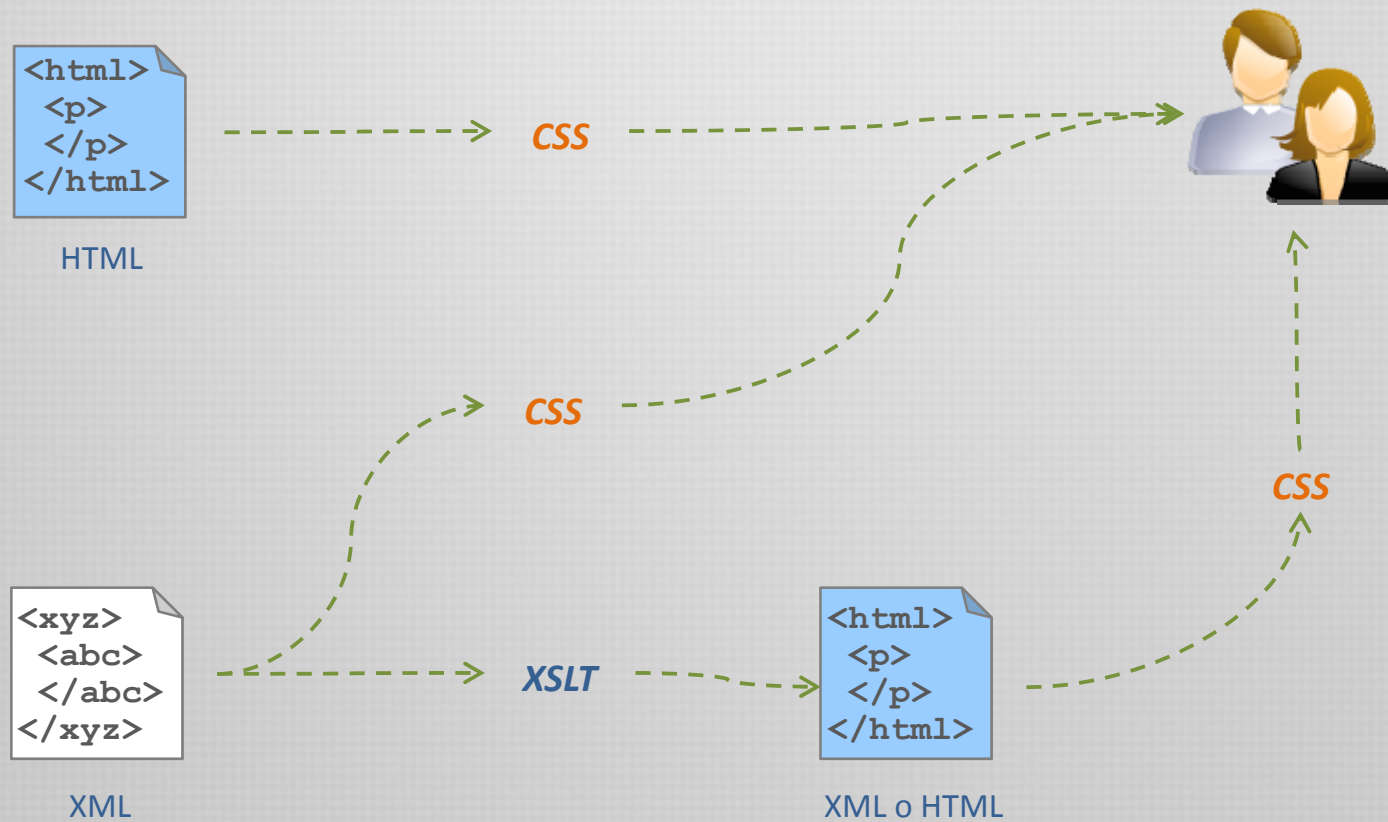
```
p {  
  margin-top: 8pt;  
  margin-bottom: 8pt;  
  font-size: 75%  
}
```

- *XSL – Extensible Stylesheet Language*

- Documento de texto con sintaxis XML.
- Más que un mecanismo de estilos: transforma documentos XMLs.
- Por ser un mecanismo de transformación, se centra más en el uso de templates que en formatos de elementos individuales.

```
<xsl:template match="p">  
  <fo:block  
    space-before="8pt"  
    space-after="8pt"  
    font-size="75%">  
    <xsl:apply-templates/>  
  </fo:block>  
</xsl:template>
```

Estilos



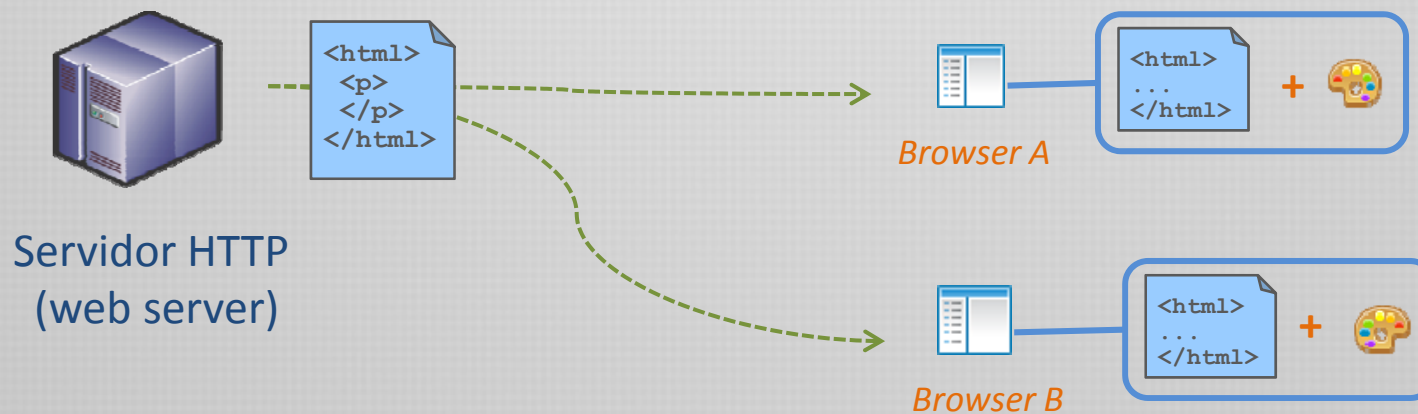
CSS – Cascading Style Sheets

Los estilos CSS nacen varios años después que HTML, a mediados de los 90.

Sin embargo, la separación *HTML-estilos de presentación* ya era considerada por Berners-Lee desde un comienzo.

El primer browser implementaba una lista de estilos universales. Berners-Lee suponía que todos los futuros browsers harían lo mismo.

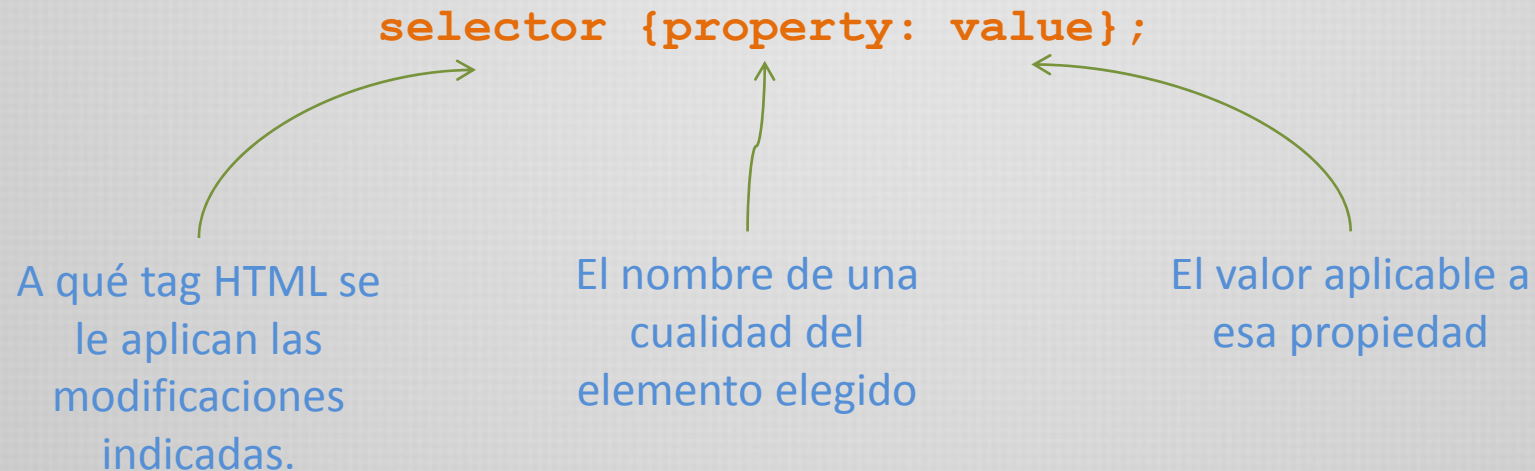
NCSA Mosaic permitía ciertas personalizaciones de estilo



En 1994 sale el primer borrador de CSS, hojas de estilo aplicables a HTML.

Cascading Style Sheets

Un documento CSS es simplemente una colección de reglas de la forma:



Por ejemplo,

```
p {  
  color: blue;  
  text-align: right;  
  font-family:  
  courier;  
}
```

```
hr {  
  color : #91B2C5;  
  height : 1px;  
  width : 100%;  
}
```

```
td, tr, div {  
  font-family : Arial;  
  font-size : x-small;  
  color : #000;  
}
```

Son características *heredables*.

Se puede indicar un font para un párrafo, y aparte un color para los links.

Luego, cada link usa sus propiedades y las del párrafo.

Cascading Style Sheets

Pueden categorizarse diferentes usos de un mismo elemento:

```
p.bodytext {color: black;}  
p.alert {color: red;}
```

Luego se indica, para ese elemento, de qué categoría es:

`<p class="bodytext">`
Esto va de color negro
`</p>`

`<p class="alert">`
Esto va de color rojo.
`</p>`

También pueden definirse elementos aplicables a cualquier tag. En este caso no se nombra el tag específico sino la *clase* de formato aplicada:

```
.highlight {color: blue;}
```

Esto se puede aplicar a cualquier tag!

```
<p class="highlight">Esto es texto azul</p>  
<h2 class="highlight">Esto es un encabezado azul</h2>
```


Cascading Style Sheets

CSS se puede aplicar de formas diferentes:

- Directamente en el elemento que se quiere formatear.
- Incluido al comienzo en el documento HTML (con el tag `<style>`), o combinado con los tags `<div>` y ``.
- En archivos separados del documento HTML.

● Inline style

Se utiliza el atributo `style`, aplicable a cualquier tag HTML:

```
<b style="color: blue;">Esto es bold y azul</b> y esto no.  
<span style="color: green;">Esto es verde</span> y esto no.
```

● Por tipo de elemento

Se utiliza el tag `style`:

```
<style type="text/css">  
b { color: blue }  
</style>
```

Todas las ocurrencias del tag bold `` también cambiará el texto a azul.

Procesando XML/HTML - modelo web - scripting

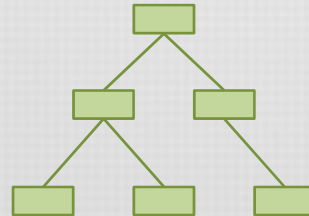
Documento

```
<html>  
<p>  
</p>  
</html>
```

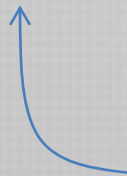
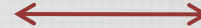
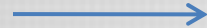
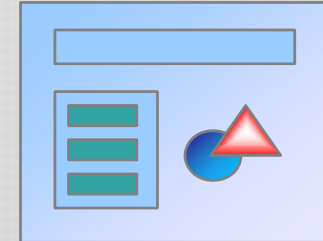


```
if()  
then  
else  
while
```

Estructura del documento



Vista del documento



simple, dinámico

JavaScript

El más popular de los lenguajes de script para la programación del lado cliente.



Creado en sus comienzos por *Netscape* y luego en conjunto con *Sun*. *Microsoft implementó también el lenguaje, pero como es habitual lo hizo a su manera (Browsers Wars!).*

Se llamaba inicialmente *Mocha*, luego *Livescript*.

JavaScript es en realidad una implementación de ECMAScript, la creación original de Brendan Eich.

Brendan Eich

A blue hand-drawn arrow originates from the text 'Brendan Eich' and points towards the portrait of Brendan Eich on the right.

- Sintaxis simple, sin tipos de datos.
- Puede modificar dinámicamente el documento en el que se encuentra, por medio de una interfaz especial denominada DOM.
- Puede reaccionar ante eventos ocurriendo en la página en la que se encuentra (*onMouseOver, onClick, onLoad, etc*).
- Puede ser utilizado para la validación de datos.
- Puede detectar el browser del visitante y ayudar a optimizar la visualización
- Puede realizar pedidos adicionales a un servidor, en *background*, en forma asincrónica (AJAX).

JavaScript

```
<html>
  <head>
    <title>Pagina</title>
  </head>
  <body>
    <h2>Bienvenido</h2>
    <div> etc </div>

    <script>
      //sentencias
    </script>

  </body>
</html>
```

*sentencias de
alto nivel*

El código será **interpretado por el navegador**.

El elemento tiene el código o una referencia a un recurso con código.

JavaScript - ejemplo

```
<HTML>
  <HEAD>
    <TITLE>Ejemplo JS</TITLE>
  </HEAD>

  <BODY>

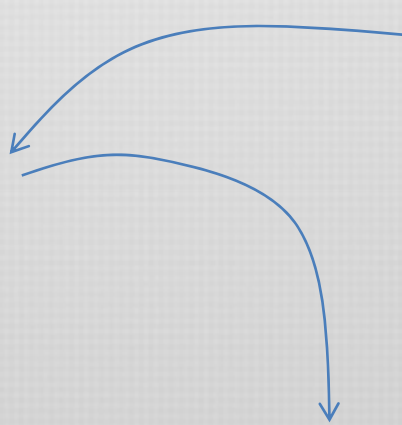
    <SCRIPT>
      var mostrar=true
    </SCRIPT>

    <p>Bienvenidos!</p>

    <SCRIPT LANGUAGE="JavaScript">
      if(mostrar==true){ document.write("Hola! "); }
    </SCRIPT>

    <p>Fin.</p>

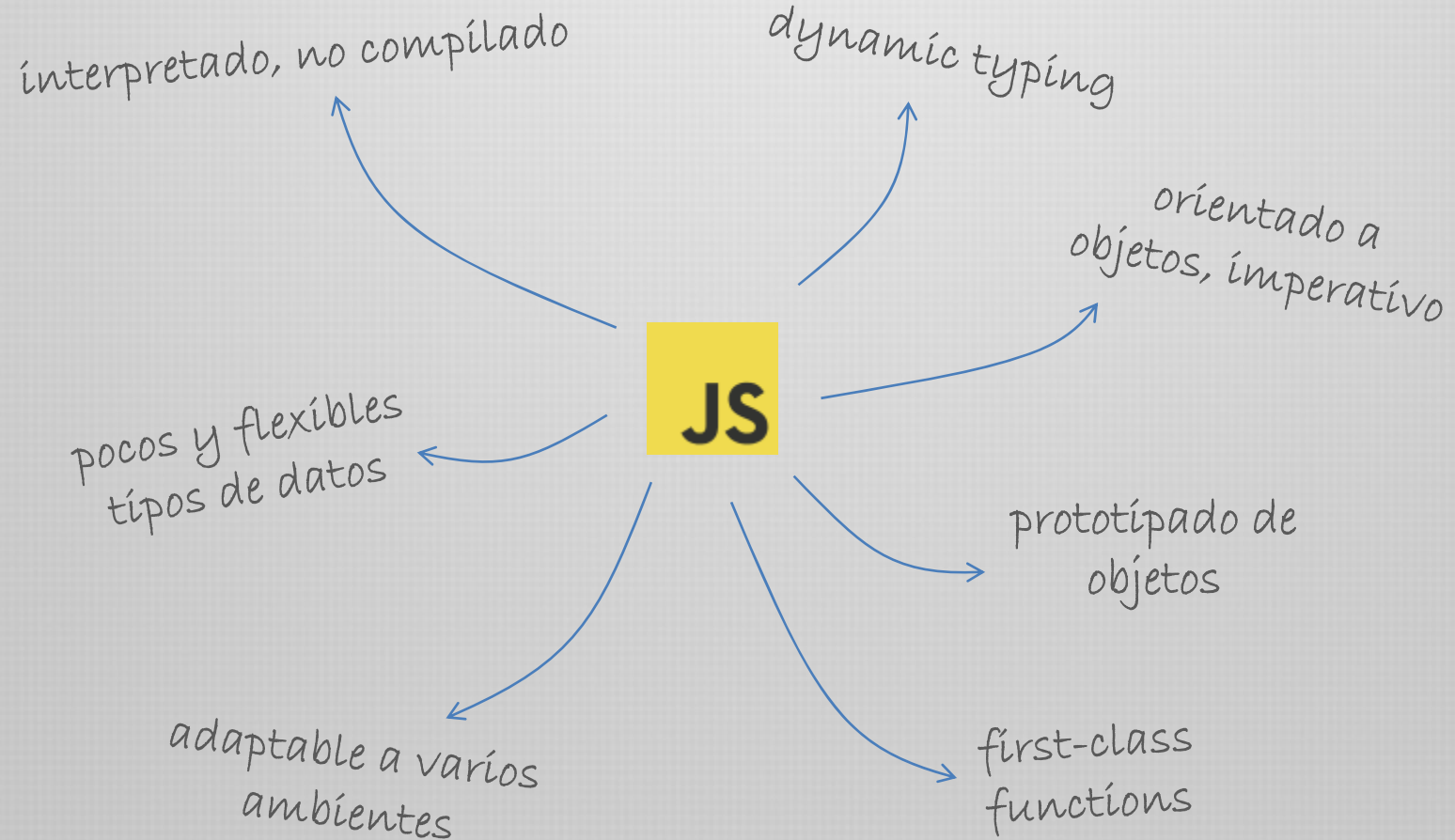
  </BODY>
</HTML>
```



continuidad

The diagram consists of two blue curved arrows. The first arrow starts at the right side of the first script block (the one with 'var mostrar=true') and points to the right side of the second script block (the one with the 'if' statement). The second arrow starts at the right side of the second script block and points down towards the 'Fin.' paragraph. The word 'continuidad' is written in a cursive script to the right of the first arrow, indicating the flow of execution.

JavaScript



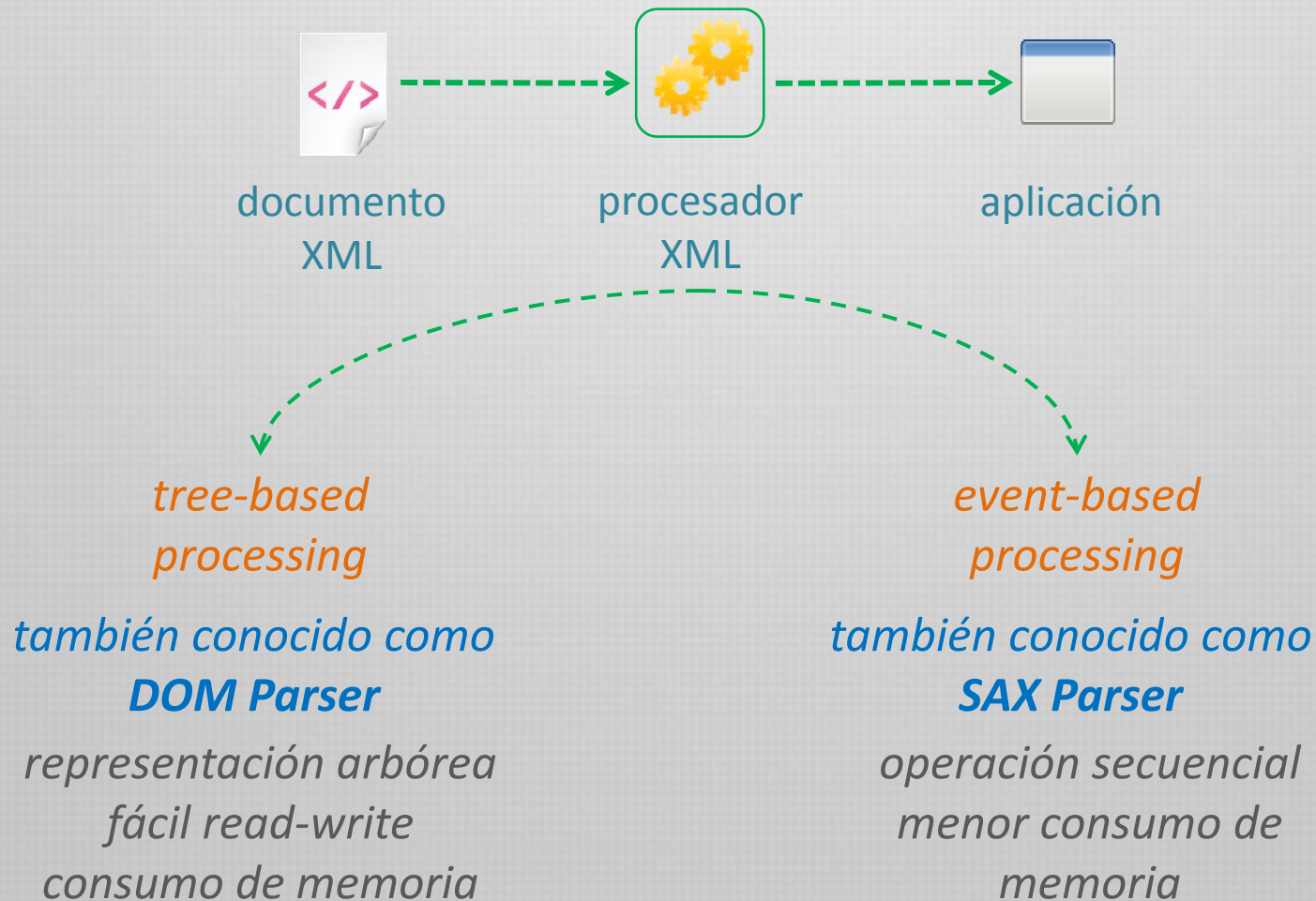
JavaScript

Ventajas principales del uso de JavaScript

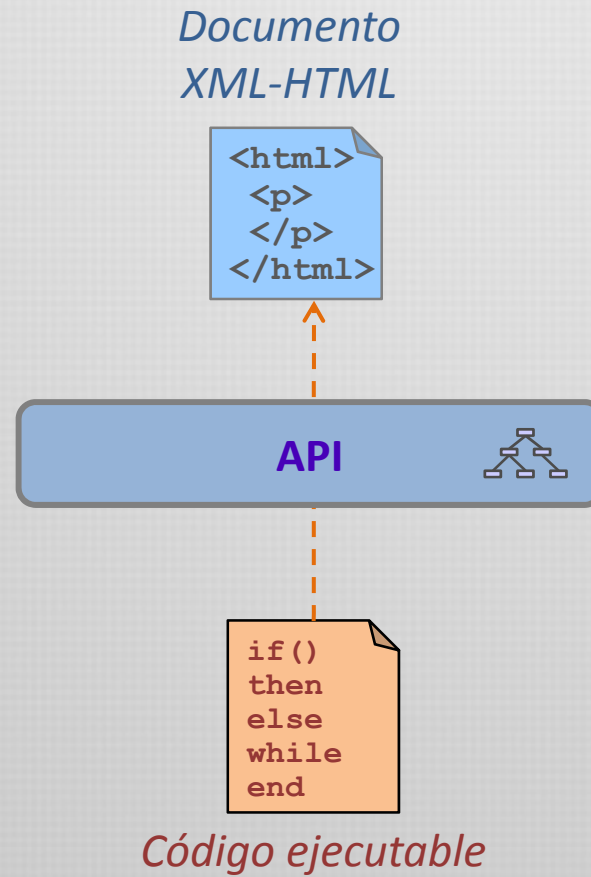
- Menos interacción con el servidor
Pueden validarse datos antes de enviar el request al servidor.
- Respuesta inmediata al usuario
Parte del procesamiento puede “adelantarse” al lado cliente
- Reparación automática y de superficie de errores menores
Principalmente formato de datos, como las fechas
- Mejor experiencia de uso
Menues desplegables o colapsables, ayudas contextuales, etc
- Mayor interacción con el usuario
Le otorga mayor reacción al uso del mouse.
- Interfaces enriquecidas
Drag-and-drop, sliders, etc.
- Optimización de la interacción cliente-servidor
Parte de la información puede obtenerse en background, actualizando parcialmente la página en vista.

Procesando XML

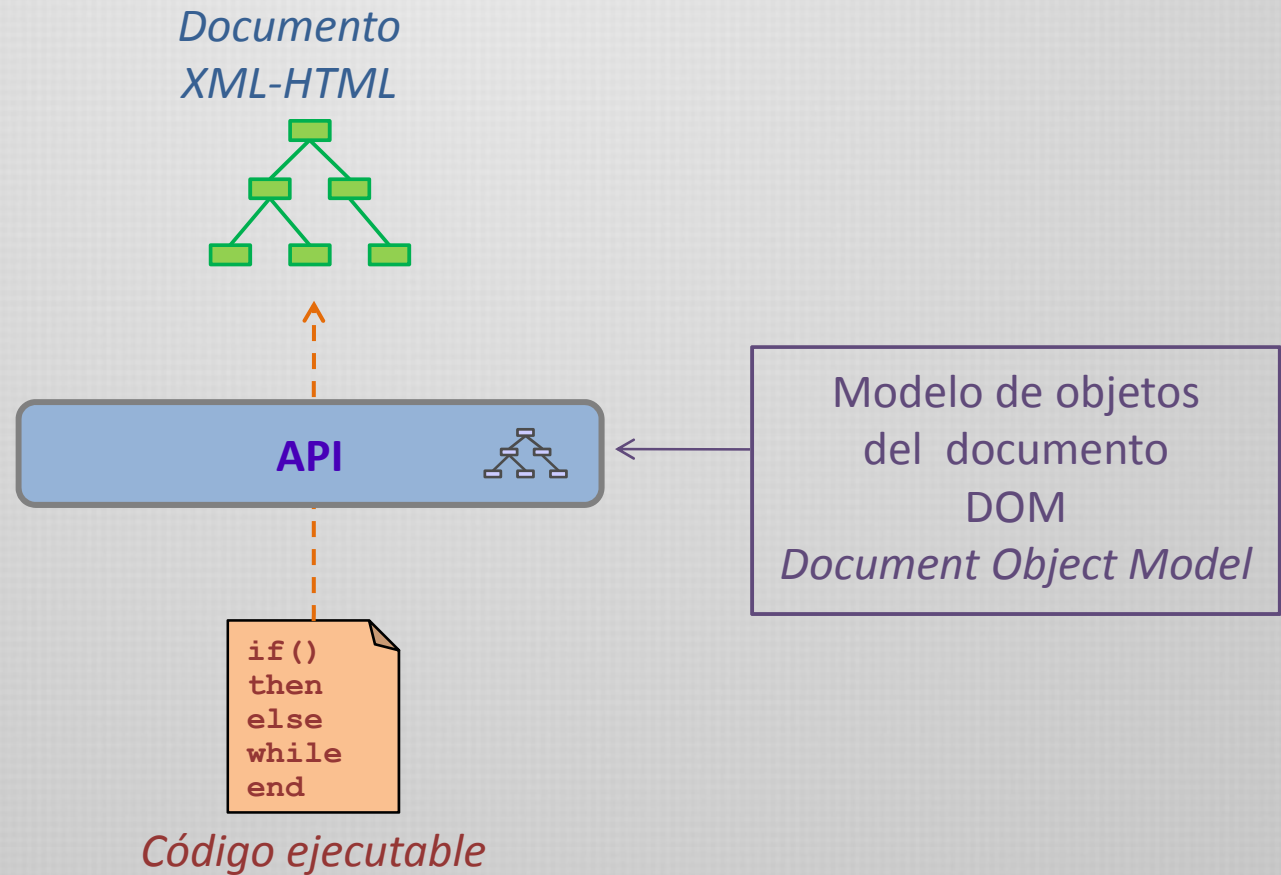
La especificación XML asume que el documento será procesado de manera especial



Procesando XML/HTML



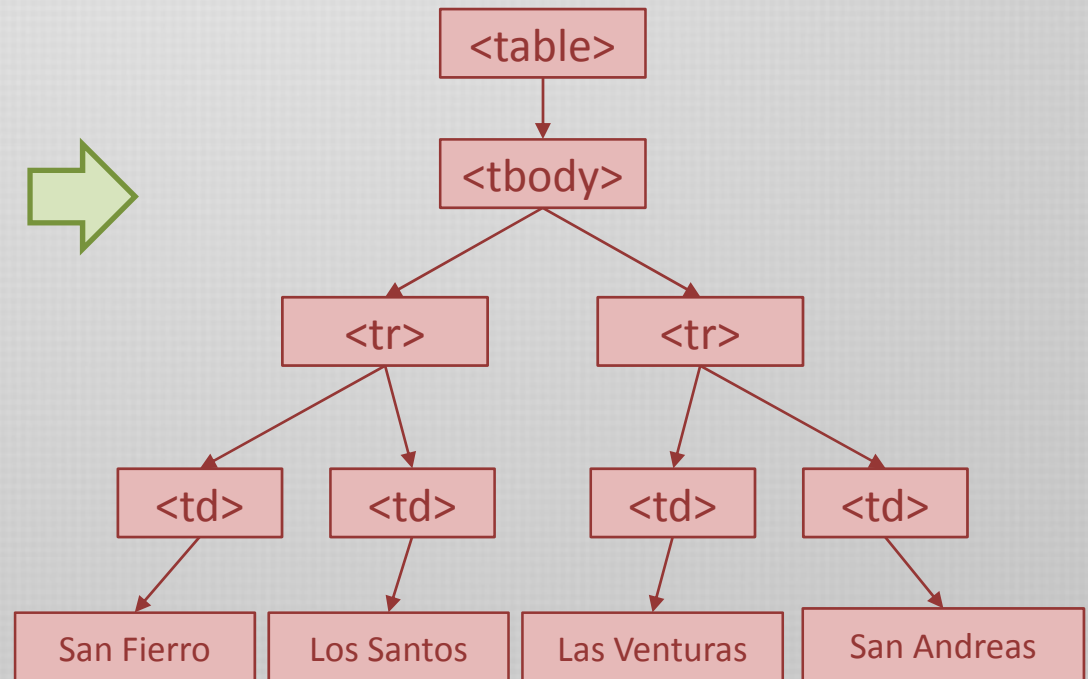
Procesando XML/HTML



Document Object Model - DOM

DOM es una interfaz de acceso y manipulación de documentos XML y HTML bien formados.
Es independiente de la plataforma, del navegador y del lenguaje que lo utiliza.
Es un estándar de la W3C desde 1998.

```
<table>
  <tbody>
    <tr>
      <td>San Fierro</td>
      <td>Los Santos</td>
    </tr>
    <tr>
      <td>Las Venturas</td>
      <td>San Andreas</td>
    </tr>
  </tbody>
</table>
```



Document Object Model - DOM

El estándar DOM identifica:

- Las interfaces y objetos usados para representar y manipular un documento.
- La semántica de estas interfaces y objetos (comportamiento y atributos)
- La relación y las colaboraciones entre estas interfaces y objetos



El estándar DOM no es una especificación binaria de implementación.

No es una forma de persistir objetos a XML.

No define la semántica interna de los XML.

El estándar DOM no es una estructura de datos

Document Object Model - ejemplo

```
<html>
<head>
<title></title>
</head>
<body><p>Este es un párrafo</p></body>
</html>
```

La siguiente expresión es equivalente al nombre "P"

document.documentElement.lastChild.firstChild.tagName

Tag HTML de la página

Tag BODY

1er. Elemento
del tag BODY

Nombre del 1er.
elemento del tag
BODY

Un acceso más simple puede conseguirse ante la posibilidad de nombrar los elementos de un documento HTML...