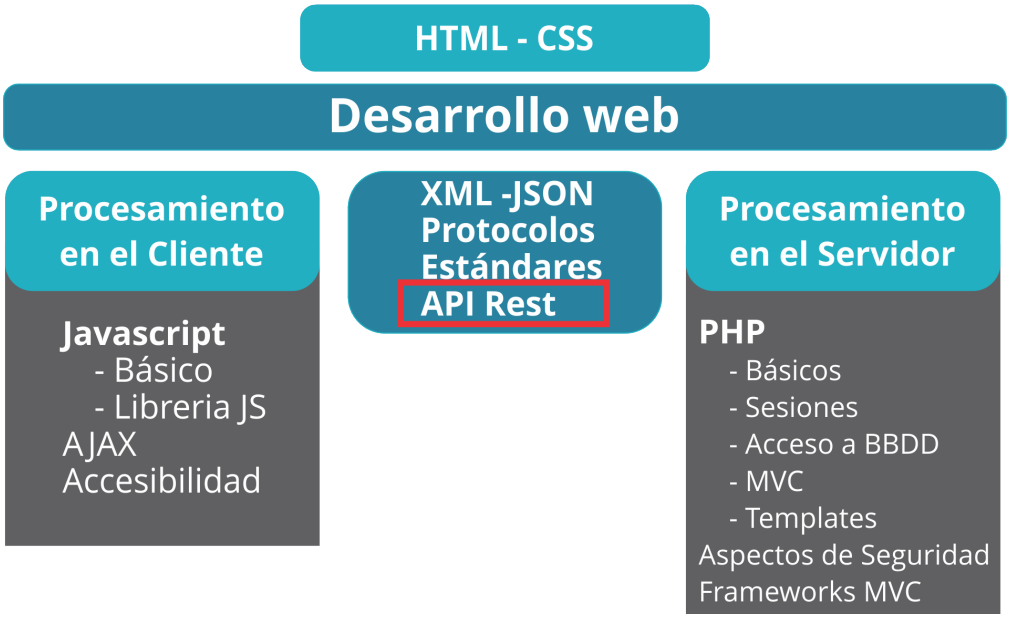
title: Clase 6 Proyecto 2016
Author: Einar Lanfranco, Claudia Banchoff
description: Notaciones para descripción de datos
keywords: DOM + Librerías Javascript
css: proyecto.css

Proyecto de Software

Cursada 2016

Hoy seguimos con ...

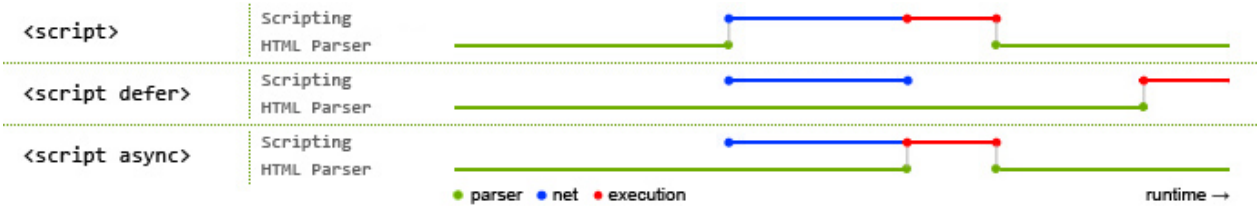


Temario

- **Repaso Clase Anterior**
    - Javascript
    - XML - JSON- YAML
- El Modelo de Objetos del Documento (DOM)
- JQuery
- AJAX

Repaso - Javascript

- Lenguaje interpretado.
- Usado tanto en el cliente como en el servidor, **aunque nosotros lo usamos en el cliente**.
- Atributos async y defer:



Sacado de: http://blognomak.blogspot.com.ar/2014/04/ejecucion-asincronica-de-javascript.html

---

Repaso - XML, JSON, YAML ....

- Surgen de la necesidad de contar con un mecanismo para describir información estructurada.
- **XML - eXtensible Markup Language**
    - Es un lenguaje de marcas.
    - Con una sintaxis estricta
    - Con posibilidad de definir la gramática: DTD - Schemas
    - Con especificaciones asociadas: XSL - Schemas - etc.
- **JSON - YAML**
    - Notaciones alternativas
    - Más ligeras
    - Populares
- **¿Cuál usamos?**
    - Depende...

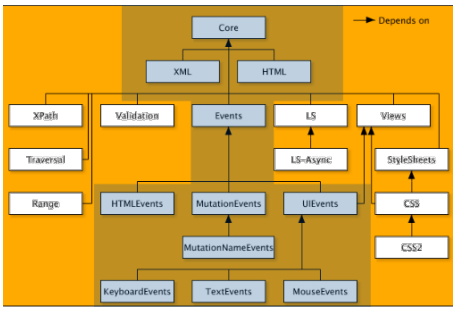
El Modelo de Objetos del Documento

DOM

- El Modelo de Objetos de Documento es una **API**, que permite acceder a los contenidos de un documento HTML/XML.
- Proporciona una **representación** estructurada, **orientada a objetos**, de los elementos individuales y el contenido del documento, con métodos para recuperar y fijar las propiedades de dichos objetos.
- Proporciona métodos para agregar y eliminar objetos al documento.

- También proporciona una **interfaz estándar** para trabajar con **eventos**.

Arquitectura DOM



Actualmente trabajando en DOM4

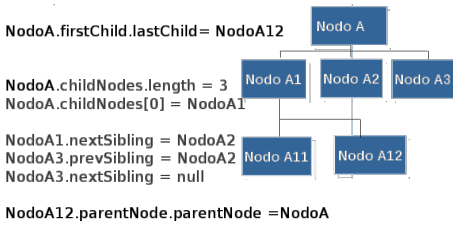
Interfaz Node

- El documento se ve como un **árbol de nodos**.
- **Cada nodo tiene sus propios métodos y propiedades, pero todos implementan la interfaz Node:**
    - Un conjunto común de métodos y propiedades relacionadas con la estructura de árbol del documento.
- **Algunos métodos implementados por esta interfaz:**
    - insertBefore()
    - appendChild()
    - removeChild()
    - cloneNode()
    - replaceChild()

Interfaz Node

- También proporciona varias propiedades que reflejan la estructura de árbol y permiten recorrerlo.
- **Algunas propiedades:**
    - firstChild
    - lastChild
    - childNodes
    - parentNode
    - nextSibling
    - prevSibling

Un ejemplo



En un documento HTML/XHTML

- El documento entero es un **nodo documento**.
- Cada elemento HTML es un **nodo elemento**.
- Los textos que aparecen en las páginas son **nodos de texto**.
- Los atributos de los elemento son **nodos atributos**.
- Los comentarios son **nodos comentarios**.

Cada nodo ...

- Posee propiedades tales como: **nodeName**, **nodeValue** y **attributes**.
- Los valores de estos atributos varían según el tipo de nodo.
- Ejemplo:

```
<img src="casita.png" alt=".."...>
```

- ¿nodeName?, ¿attributes?

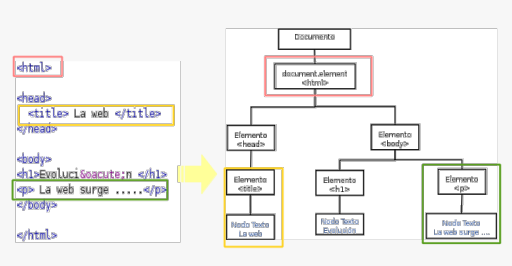
El documento

- La raíz del árbol es el **objeto document**.
- Este objeto implementa la **interfaz Document**.
- Esta interfaz proporciona métodos para acceder y crear otros nodos en el árbol del documento.
- **Algunos métodos son:**
    - getElementById()
    - getElementsByTagName()

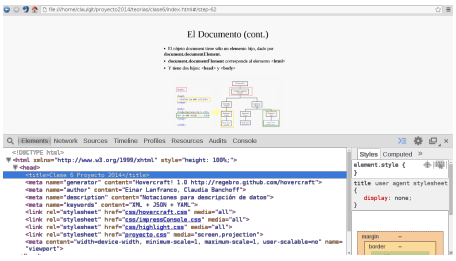
- createElement()
- createAttribute()
- createTextNode()

El documento (cont.)

- El objeto document tiene sólo un elemento hijo, dado por **document.documentElement**.
- **document.documentElement** corresponde al elemento **<html>**
- Y tiene dos hijos: **<head>** y **<body>**



DOM – Algunas herramientas



Recorriendo el árbol

Ejemplo: Dado el siguiente documento:

```
<html>
<head><title></title>
</head>
<body><h1>Algo.......</h1>
<p>bla bla bla </p>
</body></html>
```

- ¿document.documentElement.lastChild.firstChild.tagName?
- ¿Algún problema?

Accediendo a los nodos

- Todos los elementos del documento tienen un atributo que les permite identificarlos unívocamente: **id**
- Utilizando este atributo, luego podemos accederlo vía el método: **document.getElementById()**

```
function sumo(){
var x=parseInt(document.getElementById("n1").value);
var y=parseInt(document.getElementById("n2").value);
document.getElementById("result").value=x+y;
}
```

Accediendo a los nodos (cont.)

- Para recuperar todos los elementos de un mismo tipo, se puede usar el método: **document.getElementsByTagName()**

```
var x=document.getElementsByTagName("p");
for (var i=0;i<x.length;i++)
  {
  ..... // Estoy procesando todos los párrafos
  }
```

Tipos de Nodos

- **Nodos Elementos**: Corresponden a las etiquetas del documento. Pueden tener nodos hijos, que pueden ser otros elementos o nodos de texto.
- **Nodos de Texto**: Representan contenido, o simplemente caracteres. Tienen un nodo padre y, posiblemente, nodos del mismo nivel, pero no pueden tener nodos hijos.
- **Nodos atributos**: No están considerados una parte del árbol del documento. No tienen un nodo padre, ni hijos ni hermanos.

Modificando el árbol

- **Métodos para crear nodos:**
    - document.createElement()
    - document.createAttribute()
    - document.createTextNode()
- **Para insertarlo o eliminarlo del árbol:**

- appendChild()
- removeChild()

Modificando el árbol

Ejemplo: Quiero crear una lista en forma dinámica....

```
...
var lista=document.createElement("ul");
var item=document.createElement("li")

....
lista.appendChild(item);
....
document.documentElement.lastChild.appendChild(lista)
....
```

- Vemos ejemploListas

Nodos de texto

- Los nodos de texto no tienen un atributo ID.
- No se pueden acceder mediante los métodos getElementById() o getElementsByTagName().
- **Se acceden a través de su nodo padre.**
- Ejemplo:

```
.....
<p id="p1">Texto inicial ....</p>
.....
```

```
document.getElementById('p1').firstChild.nodeValue='Otro'
```

- Vemos ejemploTextos

Nodos de texto

- Contenido html en un nodo de texto

```
document.getElementById('p1').firstChild.nodeValue =
  "<ul><li>Uno</li><li>Dos</li></ul>"
```

¿Qué creen que pasa?

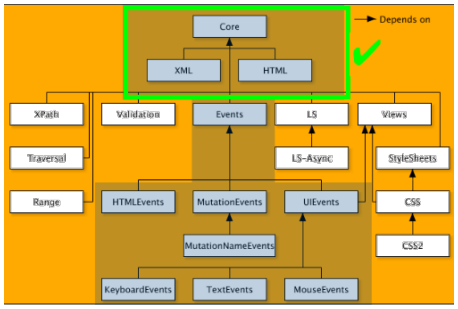
- Vemos ejemploTextosHTML

DOM y eventos

DOM también contempla...

- Un sistema de eventos genérico que permita registrar **manejadores** de eventos, describir el **flujo de eventos** a través de la estructura del árbol y proveer **información contextual** sobre cada evento.
- Un subconjunto común de los sistemas de eventos actuales.

Arquitectura DOM



Modelo de eventos

- Define y explica la propagación y registro de eventos.
- Define la Interfaz Event

Tipo de eventos

- Existe una lista de eventos definidos.
- Permite la implementación de múltiples módulos de eventos agregando los que sean necesarios.

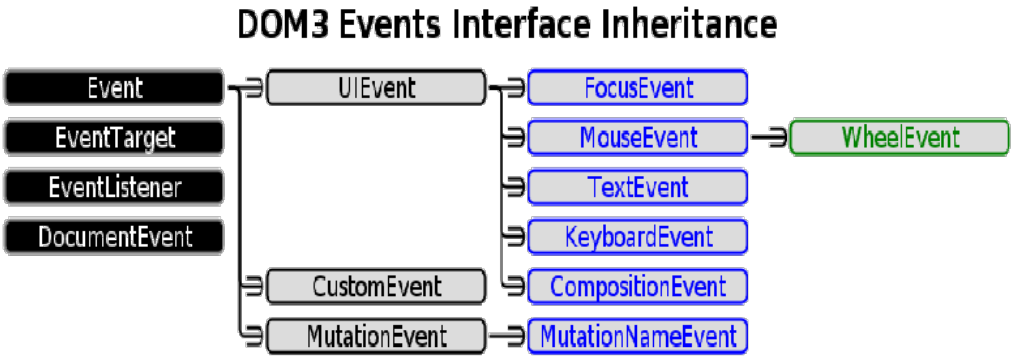


Figure 2: graphical representation of the DOM3 Events interface inheritance

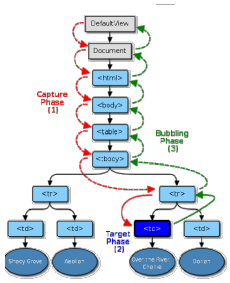
Sacado de: http://www.w3.org/TR/DOM-Level-3-Events/#event-interfaces

---

Flujo de eventos

Como los documentos HTML/XML son de naturaleza jerárquica, cuando un evento ocurre en un objeto en particular, dicho evento está ocurriendo en cualquier objeto(s) que contenga(n) a dicho objeto.

Flujo de eventos



- **1.capture phase**: El evento es enviado desde la raíz hasta el padre del nodo objetivo.
- **2.target phase**: el evento es enviado al nodo objetivo.
- **3.bubbling phase**:el evento es enviado desde el objetivo a la raíz.

La interfaz Event

- Contiene información específica sobre un evento.
- Proporciona propiedades que describen el evento y su estado actual.
- **Algunas propiedades...**
    - bubbles: si el evento burbujea o no.
    - cancelable: si el evento puede o no cancelarse.
    - target:El nodo que ha originado el evento.
    - type: tipo de evento (click, mouseover, etc)
- **Algunas métodos...**
    - preventDefault(): Para cancelar el evento.
    - stopPropagation(): Detiene la propagación.
- Ver http://www.w3.org/TR/DOM-Level-3-Events/#event-types-list

---

Manejadores de eventos

```
function manejador(evento) {
//
// "evento" se crea implícitamente y contiene la
// info sobre  el evento producido.
//
}
var elem=document.getElementById('p1')
elem.onmouseover = manejador;
```

---

class: destacado

¿Cómo asocio un manejador?

- En los **atributos** de las etiquetas HTML:

```
<input type="button" value="que aparezca la lista"
 onclick="agrego()">
```

- Usando la **propiedad** onclick del objeto

```
function agrego(evento) {....}
```

```
<input type="button" id="b1" value="que aparezca la lista">
```

```
document.getElementById('b1').onclick = agrego;
```

agrego es un objeto function. NO es una cadena de caracteres

Escuchadores de eventos

- Los objetos DOM también pueden ser registrados como escuchadores de eventos.
- Esta característica puede ser utilizada para asignar múltiples manejadores para un evento dado.
- **Los métodos básicos son:**
  - addEventListener
  - removeEventListener

Escuchadores de eventos (cont.)

```
node.addEventListener(eventType, function, useCapture);

node.removeEventListener(eventType, function, useCapture)
```

- **Donde:**
  - **eventType** es un nombre predefinido de evento como "mouseover" o "click"
  - **function** es el nombre de la función manejador y
  - **useCapture** es un booleano que especifica en qué fase del flujo de eventos el manejador debe ser llamado.

Escuchadores de eventos (cont.)

- Ejemplo:

```
<span id ="p1" style="background-color:yellow;" >
Este texto cambia de color..... </span>
.....
```

```
var elem=document.getElementById('p1')
elem.addEventListener("mouseover", f1, true):
elem.addEventListener("mouseout", f2, true):
.......

elem.removeEventListener("mouseover", f1, true);


elem.addEventListener("mouseover", f1, true):
elem.addEventListener("mouseover", f1, false):
```

AJAX

Asynchronous JAvascript + XML

AJAX

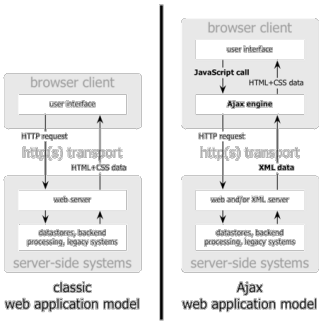
- NO es una tecnología, sino una combinación de varias tecnologías.
- **AJAX incluye:**
    - Presentación basada en estándares usando **HTML** y **CSS**;
    - Exhibición e interacción dinámicas usando **DOM**;
    - Intercambio y manipulación de datos usando **XML** y **XSLT**; (podemos usar otras notaciones también)
    - Recuperación de datos asincrónica usando **XMLHttpRequest**;
    - **JavaScript** como lenguaje de programación.

AJAX

- **Comenzó a ser popular a partir del año 2005, con Google Suggest.**
    - Hagamos una búsqueda

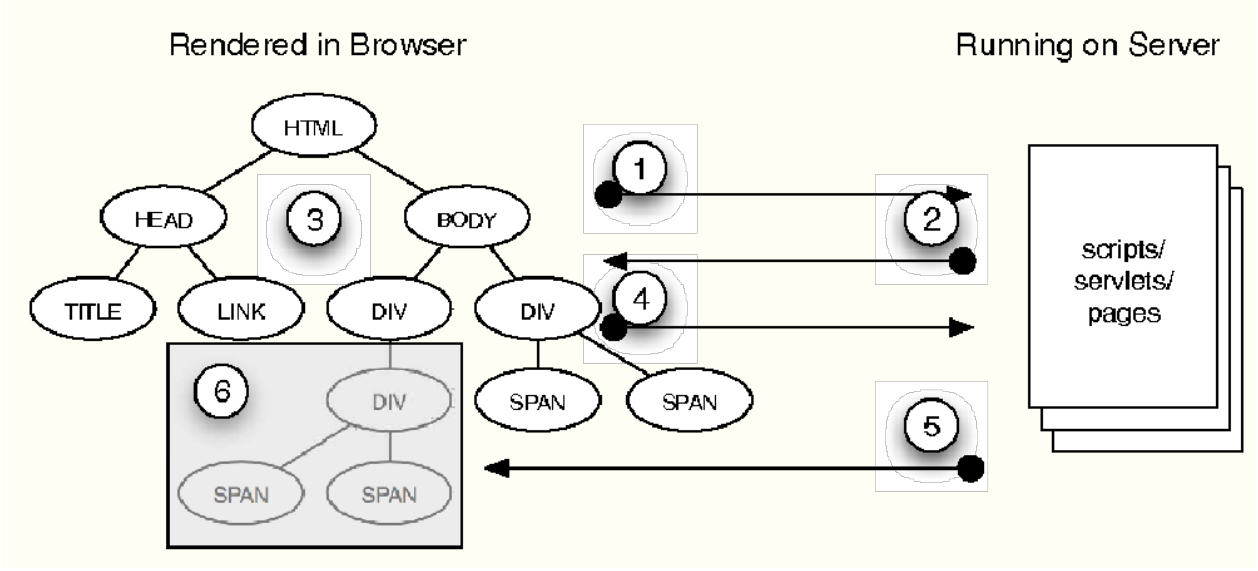
- El objetivo es crear interfaces de usuario más amigables, similares a las de las PCs de escritorio, sin afectar los tiempos y el esquema de navegación.
- **¡¡IMPORTANTE!!** El feedback al usuario.

Funcionamiento



De “Ajax: A New Approach to Web Applications”

Ciclo de vida



El Objeto XMLHttpRequest

- Es un objeto que permite realizar requerimientos HTTP al servidor web desde cualquier lenguaje de script client-side SIN recargar la página.
- Aún NO es estándar W3C XMLHttpRequest
- **Algunas propiedades...**
    - **onreadystatechange**: manejador de evento para un cambio de estado.

- **readyState: el estado del objeto:**
    - 0 = UNSENT
    - 1 = OPENED
    - 2 = HEADERS_RECEIVED
    - 3 = LOADING
    - 4 = DONE

El Objeto XMLHttpRequest (cont.)

- **Algunas propiedades (Cont.)...**
    - **responseText**: retorna la respuesta como texto.
    - **responseXML**: retorna la respuesta como XML que puede ser manipulado usando DOM.
- **Algunos métodos...**
    - **open("method", "URL", async, "uname", "pswd"): Especifica el método, URL y otros atributos opcionales del requerimiento:**
        - El método puede ser "GET", "POST", o "PUT"
        - La URL puede ser una URL completa o relativa
        - El parámetro **async** especifica si el requerimiento debe ser manejado en forma asincrónica o no (true o false)

Algunos ejemplos

- El ejemplo más simple.
- la X de AJAX es por XML: veamos ejemploXML

Librerías Javascript

Librerías Javascript

- Contienen soluciones ya implementadas para un dominio de aplicación, que el programador sólo debe usarlas.
- El objetivo es **simplificar el desarrollo**.
- **Hay muchas!**
    - Prototype: http://www.prototypejs.org/

    - YUI - Yahoo! UI Library: http://developer.yahoo.com/yui/
    - jQuery: http://jquery.com/
    - Mootools: http://mootools.net/
    - Dojo Toolkit: http://dojotoolkit.org/
- La mayoría, **open source**.

¿Por qué?

- Elementos comunes en las aplicaciones que deben ser implementados
- Controles
- AJAX
- **UI más amigables pero requieren implementación de las componentes.**
    - drag&drop
    - Sliders
    - Galerías de imágenes
    - Efectos

---

jQuery

- Una de las tantas ...
- Muy usada.
- Se debe incluir el archivo jquery.js (descargado de http://jquery.com/download/)
- Es código Javascript:

```
<script  src="ruta/jquery.js"> </script>
```

---

jQuery (cont.)

- **Nos provee formas de acceder a los elementos con atajos a la función DOM getElementById.**
    - Con DOM: **document.getElementById("p1")**
    - Con JQuery: **$("#p1")**
- **JQuery usa los selectores CSS para acceder a los elementos:**
    - **$("p"):** todos los elementos <p>.
    - **$("#elem"):** el elemento cuyo id="elem".
    - **$("p.intro"):** todos los elementos <p> con class="intro".

- **$(".intro"):** todos los elementos con class=”intro”
- **$("p#demo"):** todos los elementos <p> id="demo".
- **$(this):** el elemento actual
- **$("ul li:odd"):** Los <li> impares dentro de

Ejemplos con jQuery

- Ocultamos todo
- Ocultamos los párrafos
- Ocultamos algunos items
- Algunos efectos en imágenes
- Más efectos en los textos
- Usamos eventos
- Con formularios

Algunas consideraciones

- Función *ready()*: Nos asegura que el árbol DOM se cargue por completo.

```
$(document).ready(function() {
   // instrucciones
 });
```

- Para manejar eventos:

```
$('p').click(function() {
  alert($(this).text())
});
```

Ajax con jQuery

```
$.ajax({
  url: '/ruta/hasta/pagina.php',
  type: 'POST',
  async: true,
  data: 'parametro1=valor1&parametro2=valor2',
  success: procesaRespuesta,
  error: muestraError
});
```

- Veamos un ejemplo de ajax
- Hablamos de vinos con JSON? ¿AJAJ?