TEMA 4. PROGRAMACIÓN DE APLICACIONES EN EL CLIENTE

Curso 2019-2020

Tecnologías Web

Último cambio: 28/4/2020

Bibliografía

- R. Nixon, «PHP, MySQL, & Javascript», O'Reilly, 2009
- T. Wright, «Learning JavaScript», Addison-Wesley, 2013
- D. Flanagan, «JavaScript: The Definitive Guide», 5th Edition, O'Reilly, 2006
- A. Freeman «Pro JavaScript Web Apps», Apress, 2012

Contenido

- Aplicaciones web en el cliente
- Lenguaje JavaScript
  - DOM
  - PDO
  - Ejemplos de código
- AJAX

APLICACIONES WEB EN EL CLIENTE

Aplicación web en el cliente

- Ejecución de aplicaciones en el dispositivo cliente (navegador)
- Poca carga computacional
- Más interactividad y más rápida
  - Validación sintáctica/semántica de entradas
  - Información contextual
  - Efectos de animación
  - Gráficos interactivos
  - …

El navegador se está convirtiendo en la nueva plataforma nativa

Requisitos

- Lenguaje multiplataforma
- ¿Interpretado?
- Impone carga leve de traducción y ejecución
- Fácilmente integrable con el lenguaje de marcado (HTML)

LENGUAJE JAVASCRIPT

Lenguaje JavaScript

- Lenguaje de alto nivel, débilmente tipado, incrustable en HTML, con soporte para PDO, interpretado y ejecutado en el navegador
- Tiene acceso a todos los elementos del documento web
- Aparece en 1995 con el navegador Netscape
- Nombre oficial: EMACScript
- ¿Conexión con Java? Gestión de Java Applets
- Interacción con el servidor a través de AJAX y Node.js
- Expansión: desarrollo para otras aplicaciones

Modos de uso

1. En línea, asociado a eventos:

```
<a href="/about" onclick=
“alert(‘mensaje');” > About</a>
```

2. Incrustado: `<script> … </script>`

3. En ficheros externos:

```
<script type="text/javascript"
src="script.js"> </script>
```

El fichero no puede incluir <script>

Javascript para navegadores que no lo soportan

```
<SCRIPT>
    código javascript
</SCRIPT>
<NOSCRIPT>
Este navegador no comprende los scripts que se están ejecutando, debes
actualizar tu versión de navegador a una más reciente.
<br><br>
</NOSCRIPT>
```

Script sencillo

hola-js.html

```
<script type="text/javascript">
  window.alert("Hola mundo");
</script>
```

Ejemplos en http://betatun.ugr.es/~jmbs
http://betatun.ugr.es/~jmbs/jsX.html

Ubicaciones en el documento

- Cabecera:

  Procesado antes de recibir el cuerpo del documento; disponible en todo el documento

  Escribir metadatos
- Cuerpo
- Externo

Depuración de JavaScript

- Los mensajes no se muestran en la página
- Depende del navegador. No hay uniformidad
- Habitualmente existe una «Consola de errores» donde se muestran los mensajes

Elementos del lenguaje JavaScript

Tutorial JavaScript w3schools.com

- Sentencias; Comentarios
- Variables
- Operadores
- Tipos de datos; *arrays*
- Concatenadores
- Funciones
- Expresiones y control de flujo
- PDO
- Formularios

Sentencias

- El intérprete se activa para bloques delimitados por:

```
<script>  … </script>
```

- No es necesario que las sentencias terminen con «;», aunque sí es necesario para separar sentencias
- Sintaxis que recuerda a la de C/C++, Perl, …

Variables en JavaScript

- No hay verificación de tipos. Una variable es un espacio para almacenar datos

- Nombres formados por letras, números o $
- No tienen que comenzar por $
- Distingue entre Mayúsculas y minúsculas
- Se declaran con: `var`
- También se pueden definir como constantes: `const`

Operadores

- Aritméticos: +, -, *, /, %, ++, --
- De asignación: =, +=, -=, *=, /=, %=, .=
- Lógicos: &&, ||, !
- Comparación: ==, !=, <, >, <=, >=, ===, !==
- De bit: !,
- Concatenación de cadenas: +
- Casting: (int), (double), (string), (array), (object)

Tipos de datos

- Numéricos: enteros, coma flotante
- Lógicos: booleanos
- Cadenas de caracteres:
  - Literales: encerradas entre comillas simples: ‘
  - Interpretadas: encerradas entre comillas dobles: "

Arrays

- Tipo de dato estructurado (no necesariamente homogéneo)
- Se crean con `Array()` o simplemente `[]`
- Los componentes se acceden mediante índices, comenzando en 0. También los hay asociativos (diccionarios)
- Pueden ser multidimensionales:

```
matriz = Array(Array(1, 2, 3), Array(4, 5, 6))
```

- Crecen mediante el método `push`

Funciones

```
<script>
function suma(a, b)
{
     return a + b
}
</script>
```

Argumentos: nombre_funcion.arguments

Extensa disponibilidad de **bibliotecas** de funciones en JavaScript. Muchos *frameworks*

Ámbito de las variables

- Local: en el ámbito de la función en que se definen
- Global: definidas fuera de una función

Document Object Model (DOM)

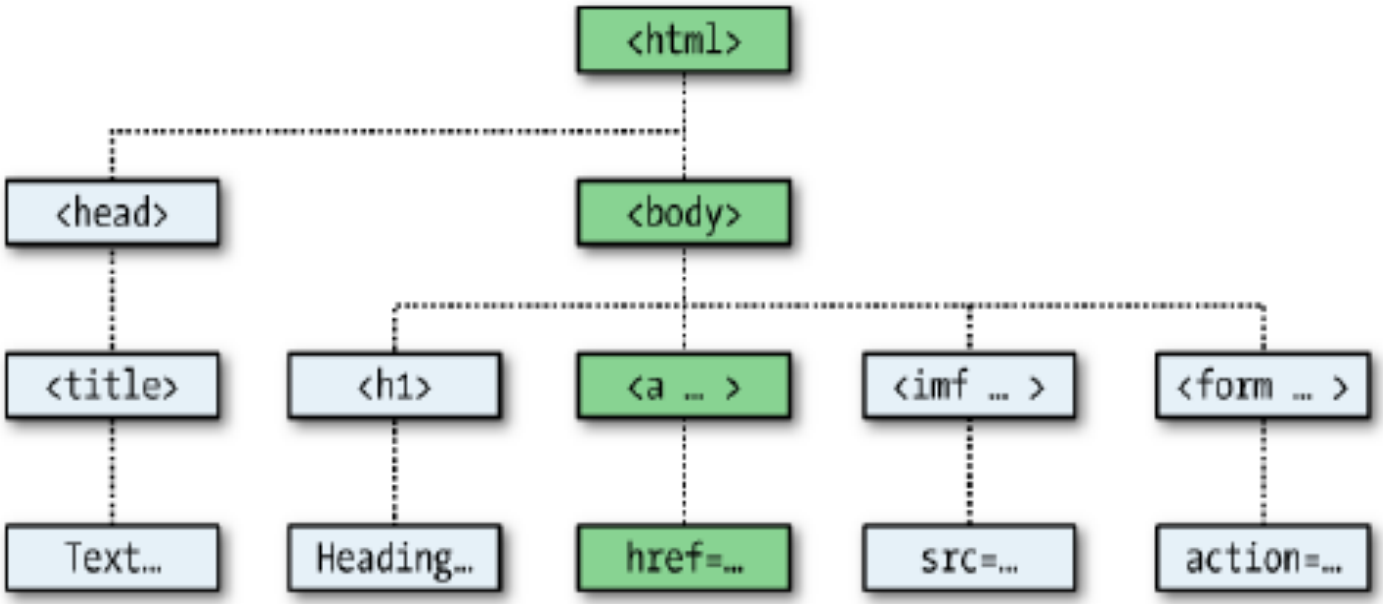
- JavaScript está diseñado entorno al Modelo de Objetos Documentales (DOM): Las partes de un documento HTML son objetos, cada uno con sus variables de instancia y métodos
- DOM: API para el procesamiento y modificación de documentos en HTML y XML
- Permite la creación, navegación, adición, modificación y eliminado de elementos y contenido
- Notación para acceso a miembros: «.»
- La relación entre los objetos es jerárquica:

  url = document.links.linkname.href
- Estándar W3C

jerarquia-js.html

```
<html>
<head>
      <title>Link Test</title>
</head>
<body>
      <a id="mylink"
         href="http://betatun.ugr.es">
            Click me</a><br />
      <script>
            url = document.links.mylink.href
            document.write("El URL es " + url)
      </script>
</body>
</html>
```

Jerarquía de objetos

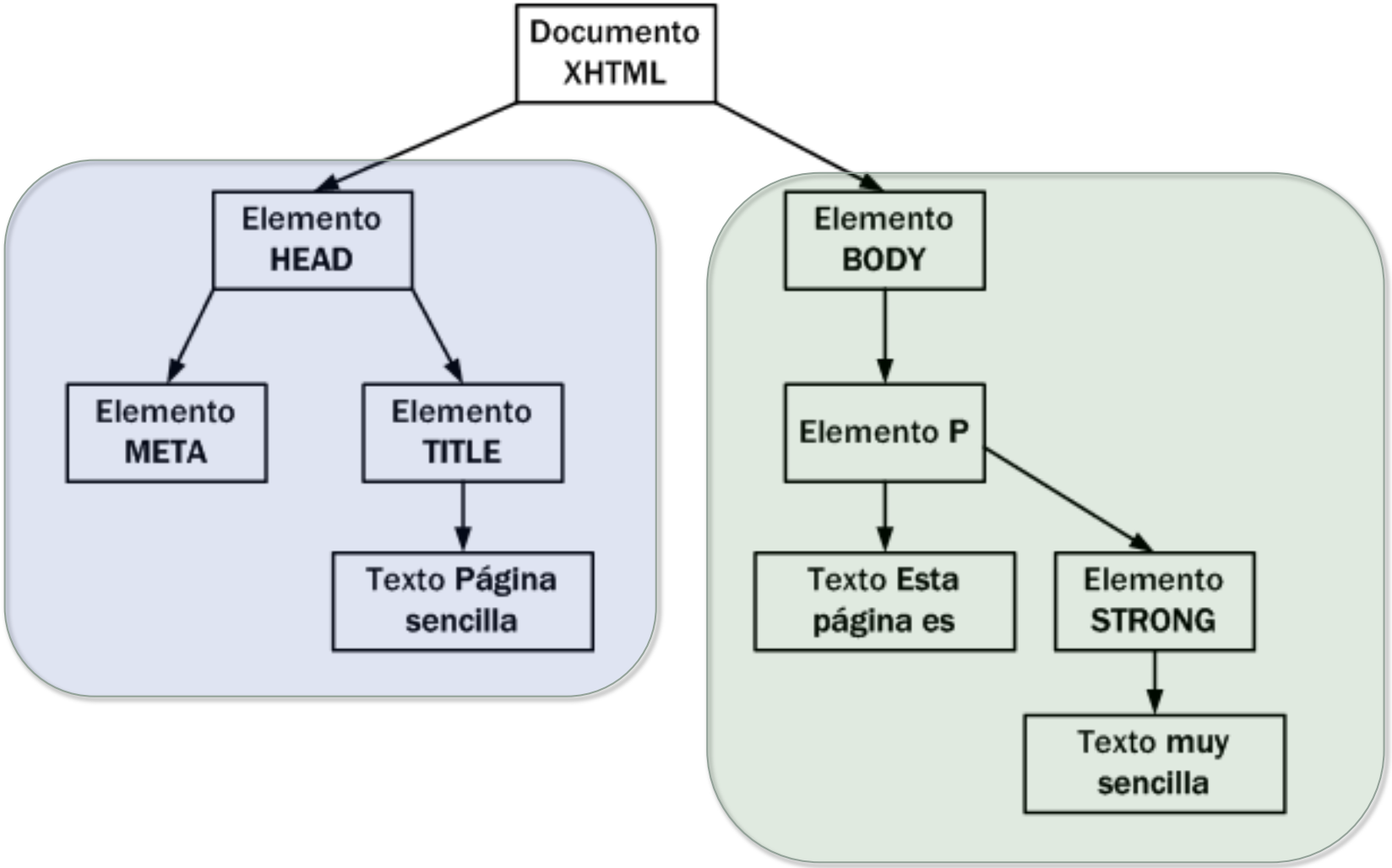


arbol-nodos.html

```
<!DOCTYPE html>
<head>
<meta http-equiv="Content-Type"
content="text/html; charset=utf-8" />
<title>Página sencilla</title>
</head>

<body>
<p>Esta página es <strong>muy
sencilla</strong></p>
</body>
</html>
```

Árbol (DOM) del arbol-nodos.html



DOM: Tipos de nodos

- Document
- Element
- Attr
- Text
- Comment
- Otros 7 tipos

HTML DOM Objects

Finding HTML Objects

The first HTML DOM Level 1 (1998), defined 11 HTML objects, object collections, and properties. These are still valid in HTML5.

Later, in HTML DOM Level 3, more objects, collections, and properties were added.

Property	Description	DOM
document.anchors	Returns all <a> elements that have a name attribute	1
document.applets	Returns all <applet> elements (Deprecated in HTML5)	1
document.baseURI	Returns the absolute base URI of the document	3
document.body	Returns the <body> element	1
document.cookie	Returns the document's cookie	1
document.doctype	Returns the document's doctype	3
document.documentElement	Returns the <html> element	3
document.documentMode	Returns the mode used by the browser	3
document.documentURI	Returns the URI of the document	3
document.domain	Returns the domain name of the document server	1

Arrays de objetos

- En realidad, cada objeto es un array:

```
url = document.links[0].href
```

- Acceso a todos los enlaces de un documento:

```
for (j=0 ; j < document.links.length ; ++j)
  document.write(document.links[j].href + '<br />')
```

Salidas de Javascript

- El lenguaje permite enviar salidas de distintas formas:

- Elemento HTML: innerHTML
  `document.getElementById(id)`
- Salida HTML:
  `document.write()`
- Ventana:
  `window.alert()`
- Consola del navegador:
  `console.log()`

Eventos en HTML

- Durante la interpretación de HTML o interacción del usuario se producen eventos, como:
  - Cambio de valor de un campo (onchange)
  - Se ha pulsado un botón (onclick)
  - Termina la descarga de una página web (onload)
- HTML permite asociar código para el procesamiento de eventos

Ejemplo: Temporizador

```
<!DOCTYPE html>
<html>
<body>

<p>Pulsa el botón para esperar 3 segundos; después sale
un mensaje</p>
<button onclick="myFunction()">Pruébalo</button>

<script>
function myFunction()
{
  setTimeout(function(){alert("Hola")},3000);
}
</script>

</body>
</html>
```

PDO en JavaScript

- El paradigma PDO está incluido en Javascript desde su creación, por tanto, es *natural*, no añadido
- Todos los elementos en Javascript derivan de esa concepción
- Ofrece dos sintaxis para la definición de clases:
  - Basada en prototipos
  - Basada en la construcción `class`

Objetos

```
var person = {
  firstName: "John",
  lastName : "Doe",
  id       : 5566,
  fullName : function() {
    return this.firstName + "_" +
       this.lastName;
  }
};

person.firstName
person["firstName"]
```

Los objetos son dinámicos

```
var person = new Object();
person.firstName = "John";
person.lastName = "Doe";
person.age = 50;
person.eyeColor = "blue";


delete person.age;


for (x in person) {
  txt += person[x];
}


var mystring = JSON.stringify(person)
```

Creación de clases con constructores

```
function User(forename, username, password)
{
  this.forename = forename
  this.username = username
  this.password = password
  this.showUser = function()
  {
    document.write("Forename: " + this.forename
+ "<br>")
    document.write("Username: " + this.username
+ "<br>")
    document.write("Password: " + this.password
+ "<br>")
  }
}
```

Creación de objetos

```
userM = new User("Wolfgang", "w.a.mozart",
"composer")

userW = new User()
userW.forename = "Wolfgang"
userW.username = "w.a.mozart"
userW.password = "composer"
```

Con sintaxis de “class”

```
class User {
  constructor(forename, username, password ) {
    this.forename = forename;
    this.username = username;
    this.password = passsword;
  }

showUser ()
 {
   document.write("Forename: " +
      this.forename + "<br />")
   document.write("Username: " +
      this.username + "<br />")
   document.write("Password: " +
      this.password + "<br />")
 }
}
```

Ampliando clases con prototipos

- Añade propiedades a una clase ya creada:
  - `this.showUser = function()`
  - `User.prototype.showUser = function()`
  - `User.prototype.greeting = “Hello”`
- Permite ampliar la funcionalidad de objetos predefinidos:
  - `string.prototype.trim = function()` …

Gestión de errores: excepciones

- Mecanismo básico de gestión de errores: excepciones
- Sintaxis y semántica similar a Java:
  - `try/catch/throw/finally`

```
try {
  Block of code to try
}
catch(err) {
  Block of code to handle errors
}
finally {
  Block of code to be executed regardless of
the try / catch result
}
```

Ejemplo

```
function lastElement(array) {
  if (array.length > 0)
    return array[array.length - 1];
  else
    throw "No puedo coger el último elemento de un" +
          "array vacío";
}

function lastElementPlusTen(array) {
  return lastElement(array) + 10;
}

try {
  print(lastElementPlusTen([]));
}
catch (error) {
  print("Algo ha ido mal: ", error);
}
```

Validación de formularios

- Se puede realizar la validación de datos de formularios con JavaScript

```
function validateForm() {
  var x =
document.forms["myForm"]["fname"].value;
  if (x == "") {
    alert("Name must be filled out");
    return false;
  }
}
<form name="myForm" action="/action_page.php"
onsubmit="return validateForm()"method="post">
Name: <input type="text" name="fname">
<input type="submit" value="Submit">
</form>
```

Manejo de cookies

- Creación
  ```
  document.cookie = "username=John Doe";
  ```
- Consulta
  ```
  var x = document.cookie;
  ```
  Devuelve **todas** las cookies en una cadena:
  ```
  cookie1=value; cookie2=value; cookie3=value;
  ```
  Hace falta procesar la cadena
- Modificación
- Borrado
  ```
  document.cookie = "username=; expires=Thu, 01 Jan
  1970 00:00:00 UTC; path=/;";
  ```

getCookie

```
function getCookie(cname) {
  var name = cname + "=";
  var ca = document.cookie.split(';');
  for(var i = 0; i < ca.length; i++) {
    var c = ca[i];
    while (c.charAt(0) == ' ') {
      c = c.substring(1);
    }
    if (c.indexOf(name) == 0) {
      return c.substring(name.length,
                         c.length);
    }
  }
  return "";
}
```

setCookie, checkCookie

```
function setCookie(cname, cvalue, exdays) {
  var d = new Date();
  d.setTime(d.getTime() +
            (exdays * 24 * 60 * 60 * 1000));
  var expires = "expires="+d.toUTCString();
  document.cookie = cname + "=" + cvalue + ";" +
                    expires + ";path=/";
}
function checkCookie() {
  var user = getCookie("username");
  if (user != "") {
    alert("Welcome again " + user);
  } else {
    user = prompt("Please enter your name:", "");
    if (user != "" && user != null) {
      setCookie("username", user, 365);
    }
  }
}
```

Browser Object Model (BOM)

- Jerarquía de objetos similar al DOM para los **componentes del navegador**.
- Elementos:
  - Screen
  - Location
  - History
  - Navigator
  - Cookies
- permite acceder y modificar las propiedades de las ventanas del propio navegador
- es posible redimensionar y mover la ventana del navegador, modificar el texto que se muestra en la barra de estado y realizar muchas otras manipulaciones no relacionadas con el contenido de la página HTML
- Poca estandarización

Principales bibliotecas en JS

https://www.javascripting.com

- Prototype: extensión de funcionalidad PDO
- React: Interfaz
- script.aculo.us: Extensión para interfaces de usuario
- MooTools
- jQuery: document manipulation, navegation and animation
- DOJO
- DS3
- Vue

Intercambio ligero de datos: JSON

- JavaScript Object Notation: formato de intercambio de datos que no requiere XML
- Es más ligero, más simple
- Más sencillo de usar y manejar
- Se puede usar conjuntamente con XML

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

Ejemplo de JSON: menú

```
{"menu": {
    "id": "file",
    "value": "File",
    "popup": {
      "menuitem": [
        {"value": "New", "onclick”:"CreateNewDoc()"},
        {"value": "Open", "onclick": "OpenDoc()"},
        {"value": "Close", "onclick": "CloseDoc()"}
      ]
    }
 }
}
```

Rizando el rizo … node.js

- Entorno de ejecución multiplataforma, de código abierto, para aplicaciones en servidores web y nativas, basado en JavaScript.
- Arquitectura orientada a eventos
- Asíncrono
- Motor V8 JS (Chrome)

WebAssembly (Wasm)

https://webassembly.org

- Lenguaje binario para una máquina virtual basada en pilas.
- *Bytecode* para lenguajes de alto nivel como C/C++ o Rust, para facilitar el desarrollo en el ámbito del servidor y del cliente.
- Propiedades
  - Tamaño reducido: transferencias rápidas
  - Rápido: tiempos cercanos al ensamblador
  - Seguro: entorno de ejecución controlado
  - Soporte para depuración
  - Estándares abiertos
  - Para la web y fuera de la web

WEBASSEMBLY

AJAX

Bibliografía

- D. Crane, E. Pascarello, D. James, «AJAX in Action», Manning, 2006
- A. Harris, «JavaScript & AJAX for Dummies», Wiley, 2010
- S. Jacobs, «Beginning XML with DOM and AJAX», Wrox Professional, Apress, 2006
- R. Aselson, N.T. Schutta, “Foundations of Ajax”, Apress, 2006.

Contenido

- Concepto
- Tecnologías componentes
- Principios
- Ejemplos de aplicaciones

AJAX

- **A**synchronous **J**avaScript **a**nd **X**ML

- Técnica de desarrollo web para crear **aplicaciones interactivas** de modo particular y gestionar **comunicación asíncrona** con el servidor (Rich Internet Applications)
- Multiplataforma (S.O., navegador)
- Integra diversas tecnologías de desarrollo web usándolas de formas creativas y nuevas
- Término creado por J.J. Garrett en 2005.

  "Técnicas para carga asíncrona de contenidos en una página existente sin requerir recarga completa"

Asíncrono

- Hace referencia a dos procesos entre los que no hay dependencia temporal. En el contexto web, significa que se pueden hacer simultáneamente. Por ejemplo, múltiples peticiones al servidor, independientes entre sí
- AJAX no implica obligatoriedad en la asincronía, pero suele ser así

XML (Datos)

- El formato de datos habitual de intercambio es XML, pero no es exclusivo.
- Otras opciones (Tema 5):
  - Ficheros de texto
  - HTML formateado
  - JSON

¿Por qué surge AJAX?

- Surge de la necesidad de empujar el desarrollo web a niveles mucho más lejanos de los disponibles
- Forzar el alcance de la tecnología mucho más allá de lo previsto en su creación
- Desarrollar aplicaciones web con la interactividad de las de escritorio
- Aplicación ilustrativa: carrito de la compra
  - Versión clásica: se envía una página para cada consulta
  - Versión AJAX: se envía sólo la información precisa, sin cambiar **toda** la página

AJAX en acción

La comunicación con el servidor se realiza en el trasfondo, obteniendo sólo los datos realmente necesarios, en lugar de páginas completas

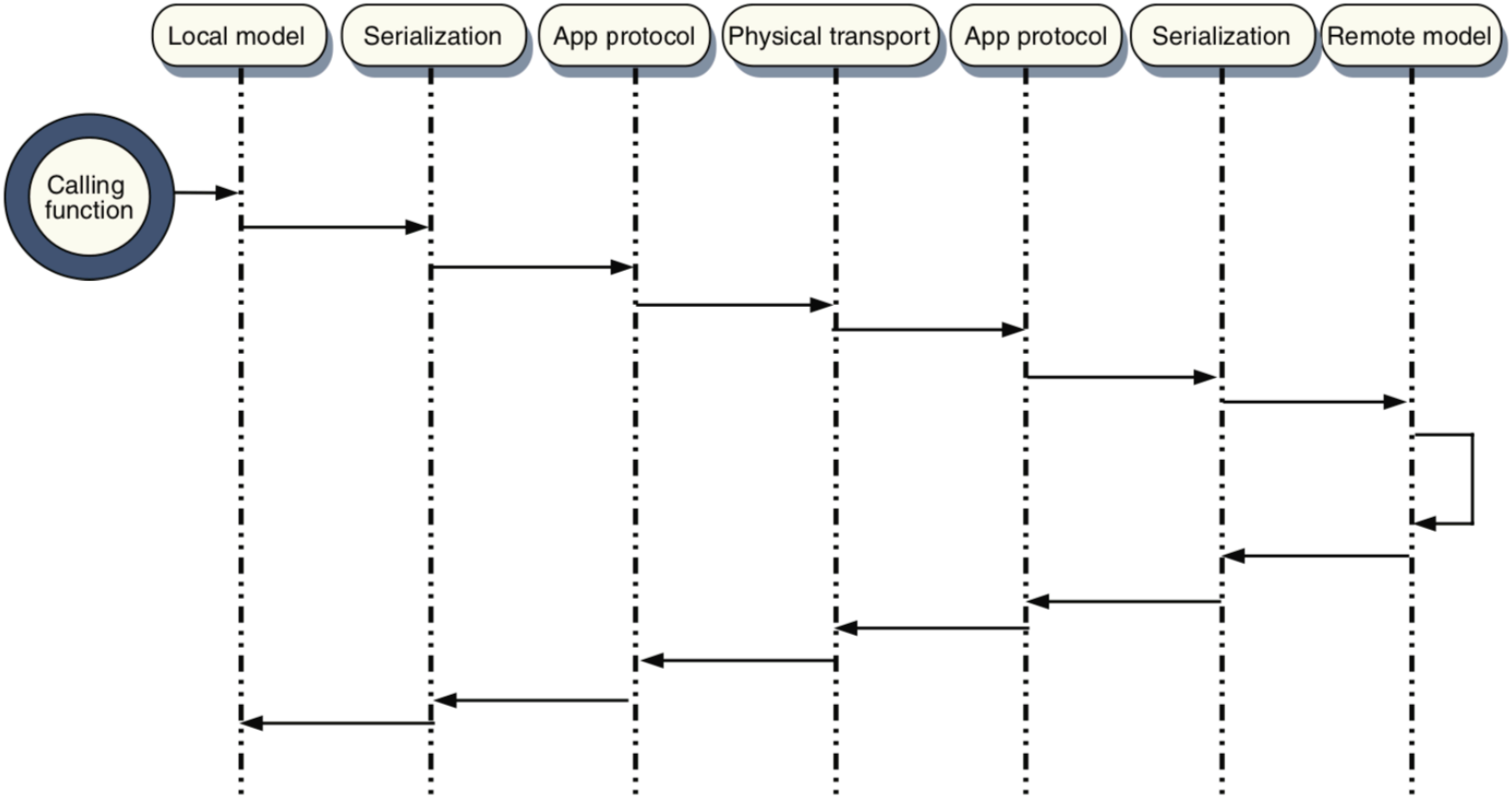
La información se muestra instantáneamente sin refrescos de página, ni esperas

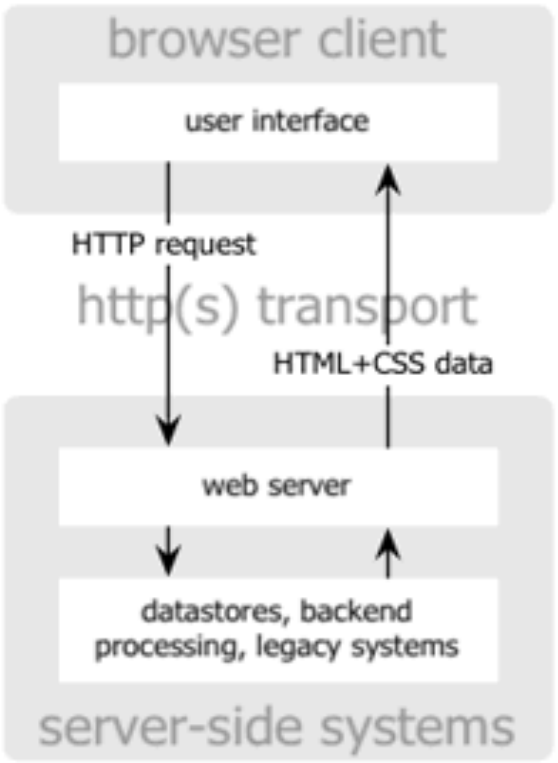
Páginas más dinámicas y reducción de tráfico (volumen de datos)

Probar www.google.com, Google Docs y Google maps

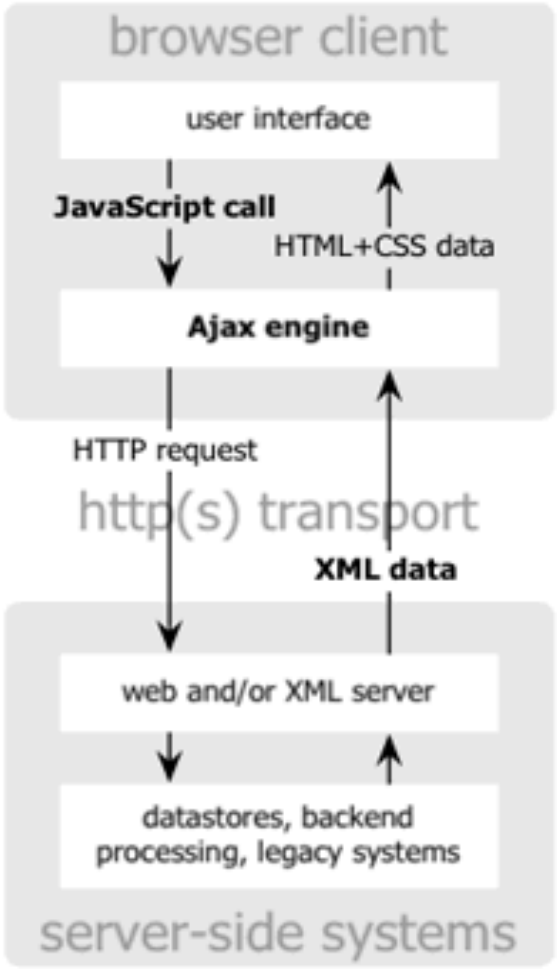
Búsquedas en vivo o “drag-and-drop”

Llamadas a procedimiento remoto (RPC)



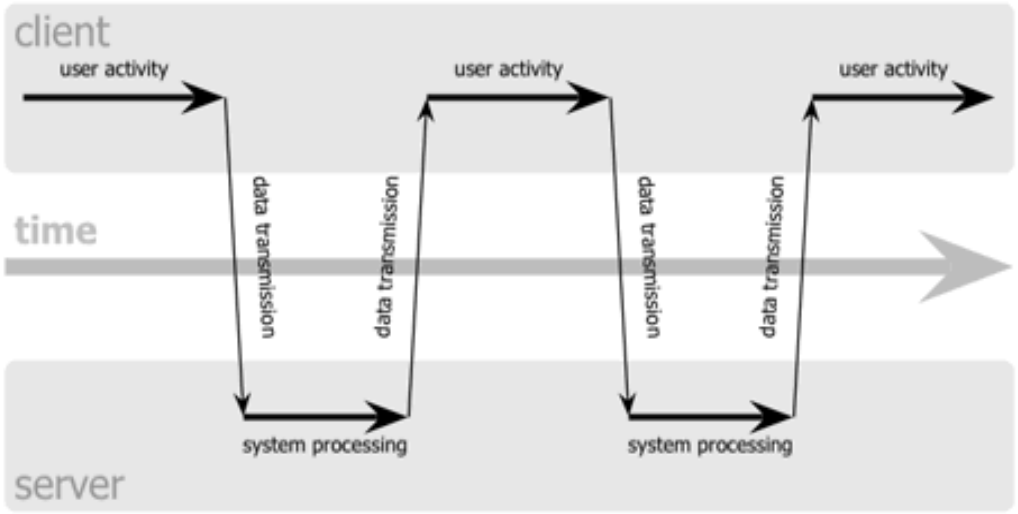


classic
web application model

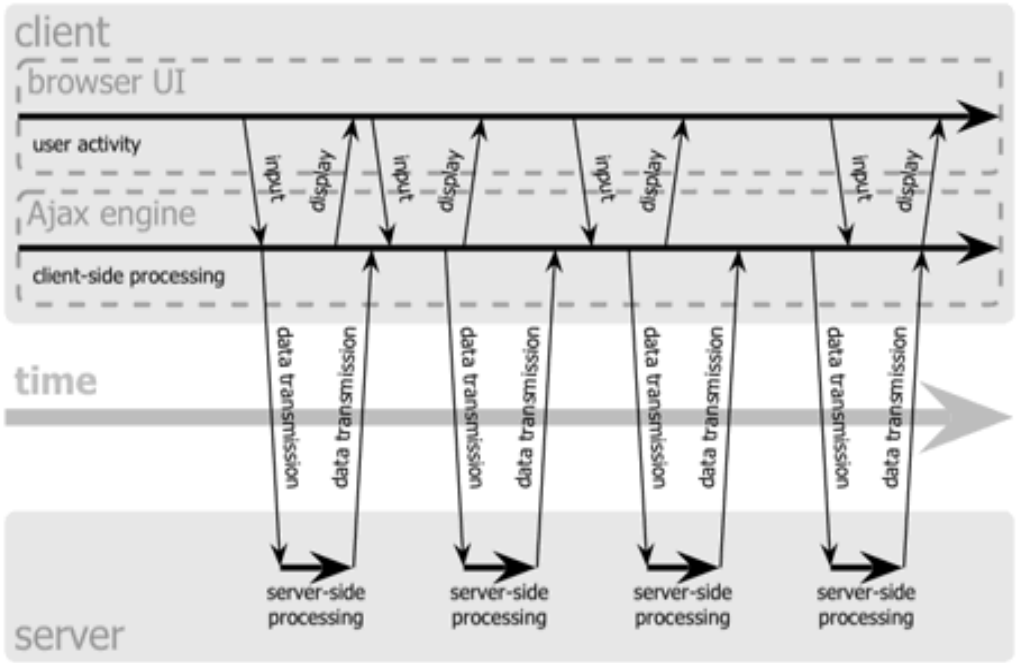


Ajax
web application model

classic web application model (synchronous)



Ajax web application model (asynchronous)

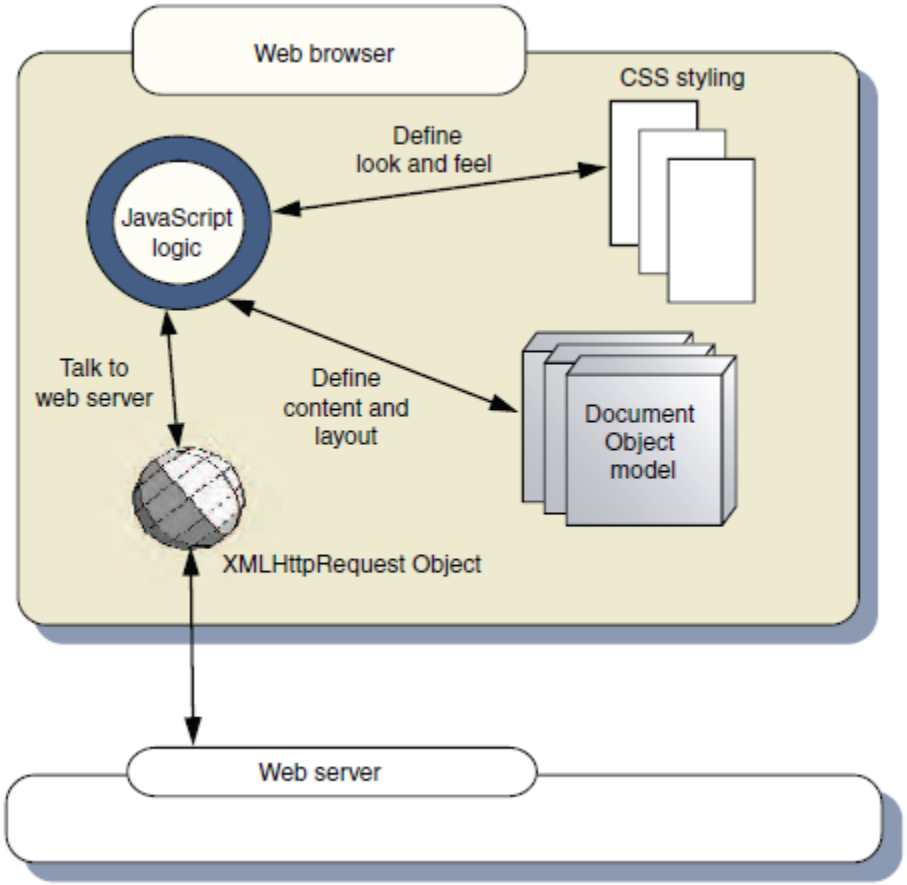


Tecnologías incluidas

- JavaScript
- DOM
- CSS
- XMLHttpRequest

XMLHttpRequest (XHR)

- Objeto que no pertenece al estándar, pero soportan la mayoría de (implementaciones de JavaScript en) los navegadores
- Es un objeto para implementar una forma efectiva de hacer peticiones a un servidor web sin tener que recargar una página. Permite hacer peticiones http desde JavaScript
- El objeto se controla con los métodos de la clase


Web browser
CSS styling
Define
look and feel
JavaScript
logic
Talk to
web server
Define
content and
layout
Document
Object
model
XMLHttpRequest Object
Web server

```
<!DOCTYPE html PUBLIC
“-//W3C//DTD XHTML 1.0 Strict//EN”
“http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd”>
<html lang = “EN” xml:lang = “EN” dir = “ltr”>
<head>
<meta http-equiv=“content-type” content=“text/xml; charset=utf-8” />
<title>asynch.html</title>
<script type = “text/javascript”>
//<![CDATA[
var request; //make request a global variable
function getAJAX(){
  request = new XMLHttpRequest();
  request.open(«GET», «beast.txt»);
  request.onreadystatechange = checkData;
  request.send(null);
}
function checkData(){
  if (request.readyState == 4) {
    // if state is finished
    if (request.status == 200) {
      // and if attempt was successful
     alert(request.responseText);
    }
  }
}
```

Ejemplo: Validacion de usuario (1)

```
<form>
  <p>
    Username: <input type="text”
      id="txtUserName" size="20”
      onblur="doCheck(this.value);"/>
    <span id="invalidMessage" class="invalid">
       </span>
  </p>
  <p>
    Password: <input type="text”
      id="txtPassword" size="20"/>
  </p>
</form>
```

Validación de usuario (2)

```
function doCheck(username) {
  if (username.length > 0) {
    document.getElementById("invalidMessage").innerHTML = "";
    if (window.XMLHttpRequest){
      xmlhttp=new XMLHttpRequest();
    }
    else if (window.ActiveXObject){
      try {
        xmlhttp = new ActiveXObject("Msxml2.XMLHTTP");
      } catch(e) {
      try {
        xmlhttp = new ActiveXObject("Microsoft.XMLHTTP");
      } catch(e) {
      xmlhttp = false;
      }
    }
  }
  if (xmlhttp){
    xmlhttp.onreadystatechange=checkNames;
    xmlhttp.open("GET", "usernames.xml", true);
    xmlhttp.send(null);
    }
   }
}
```

Clientes en JavaScript

- Un cliente en JavaScript combina datos, presentación y lógica de programa
- El aspecto se define con CSS
- La estructura de un documento es consultada y modificada mediante programación a través del DOM
  - La variable «document» es un enlace al nodo raíz
- Acceso asíncrono a los datos mediante XML

Principios definitorios de AJAX

1. El navegador alberga aplicaciones
2. El servidor provee datos
3. Interacción con el usuario fluida y continua
4. Aplicaciones AJAX

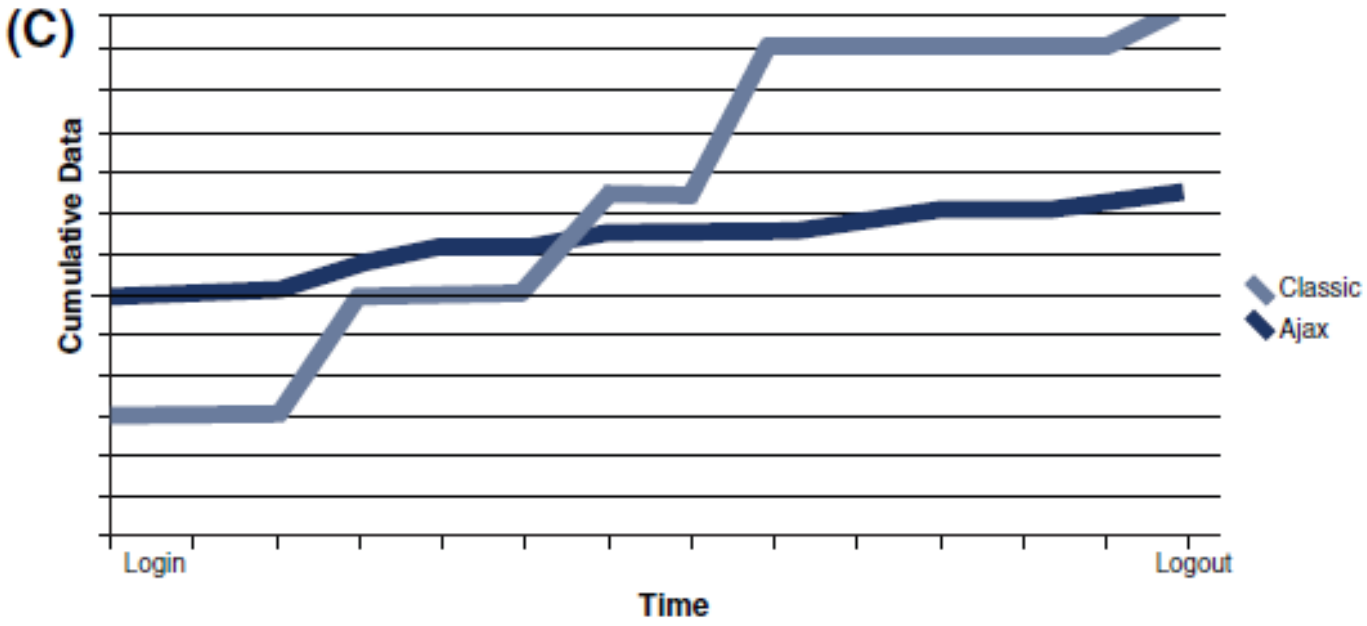
1. El navegador alberga aplicaciones

- En el modelo clásico de aplicación web, el navegador es como un terminal «tonto»
- En AJAX, parte de la lógica de la aplicación se traslada al navegador. Los documentos son más complejos
- El documento es persistente durante toda la sesión. Puede almacenar un estado

2. El servidor provee datos

- La interacción del usuario requiere del envío de pequeñas cantidades de datos
- El tráfico en aplicaciones AJAX tiene una carga fuerte inicial y después es muy reducida

Tráfico de red: AJAX vs clásica



3. Interacción del usuario fluida y continua

- Interacción en AWC: formularios e hiperenlaces
- «limbo» de interacción mientras se actualiza la página
- AJAX: interacción más fluida y continua: conectar funciones a eventos
- Conceptos de «drag-and-drop» acercan la experiencia web a la de aplicaciones de escritorio
- La comunicación no necesita una confirmación explícita

4. Aplicaciones AJAX

- Aplicación AJAX: aplicación informática en sentido estricto, no sólo pequeños scripts.
- Aplicación: modelado de datos, interacción con el usuario, procesamiento, comunicación con el servidor, generación de salida
- Aplicar buenas prácticas de desarrollo y programación: programación basada en patrones

Ejemplos de aplicaciones AJAX

- Google Apps:
  - Gmail
  - Google Maps
  - Google Suggest
  - www.flickr.com

Aplicaciones web clásicas vs. AJAX

- Aplicación web clásica: El flujo de trabajo se define por código en el servidor; el usuario va de una página a otra, con la recarga de páginas completas
- AJAX: El flujo de trabajo se define mediante software en el cliente y la comunicación el servidor se hace en el trasfondo mientras el usuario interacciona con el cliente
- La diferencia que imprime AJAX no es la tecnología sino el modelo de interacción que imprime a través del uso de las tecnologías

Inconvenientes de AJAX

- Sobrecarga de la red (descargas iniciales; mayor número de conexiones)
- Dificultad para identificar cambios
- Las «nuevas» páginas no se registran en el historial de navegación
- El contenido generado no es indizado por motores de búsqueda
- No es totalmente portable

Bibliotecas AJAX

- DOJO: widgets para la GUI
- MochiKit: Facilita la programación en JavaScript, permitiendo un estilo similar a Python
- Prototype: Soporte para AJAX y sus extensiones
- jQuery: Facilita el desarrollo de código en JavaScript

ACTIVIDADES COMPLEMENTARIAS

Actividades recomendadas

- Probar desarrollo con node.js (contenedor en betatun)
- Probar desarrollo y despliegue de aplicaciones con webassembly (contenedor en betatun)

Para profundizar

1. Estudiar las principales bibliotecas para JavaScript: React, jQuery, D3S
2. Desarrollo de las prácticas en node.js
3. Desarrollo de las prácticas en webassembly
4. Estudiar el estándar BOM