Aplicaciones Web

AJAX

David Cabrero Souto

Grupo MADS (http://www.grupomads.org/)
Universidade da Coruña

Introducción

- Asynchronous Javascript and XML.
- Tecnología conocida.
- Buzz: Gmail, Web2.0, ...
- Santillana del Mar:
  - Asynchronous, pero también síncrono.
  - XML, pero también JSON.

Introducción

- Asynchronous Javascript and XML.
- Tecnología conocida.
- Buzz: Gmail, Web2.0, ...
- Santillana del Mar:
  - Asynchronous, pero también síncrono.
  - XML, pero también JSON.

Introducción

- Asynchronous Javascript and XML.
- Tecnología conocida.
- Buzz: Gmail, Web2.0, ...
- Santillana del Mar:
  - Asynchronous, pero también síncrono.
  - XML, pero también JSON.

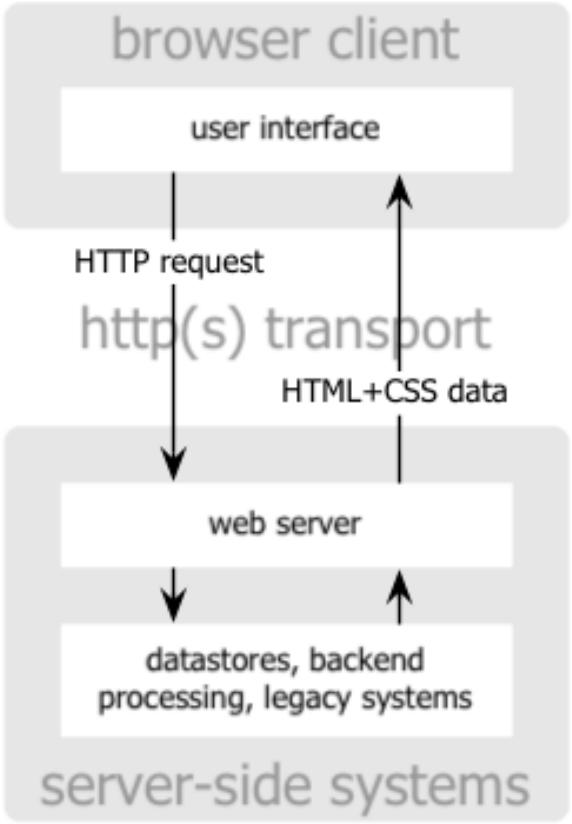
Introducción

- Asynchronous Javascript and XML.
- Tecnología conocida.
- Buzz: Gmail, Web2.0, ...
- Santillana del Mar:
  - Asynchronous, pero también síncrono.
  - XML, pero también JSON.

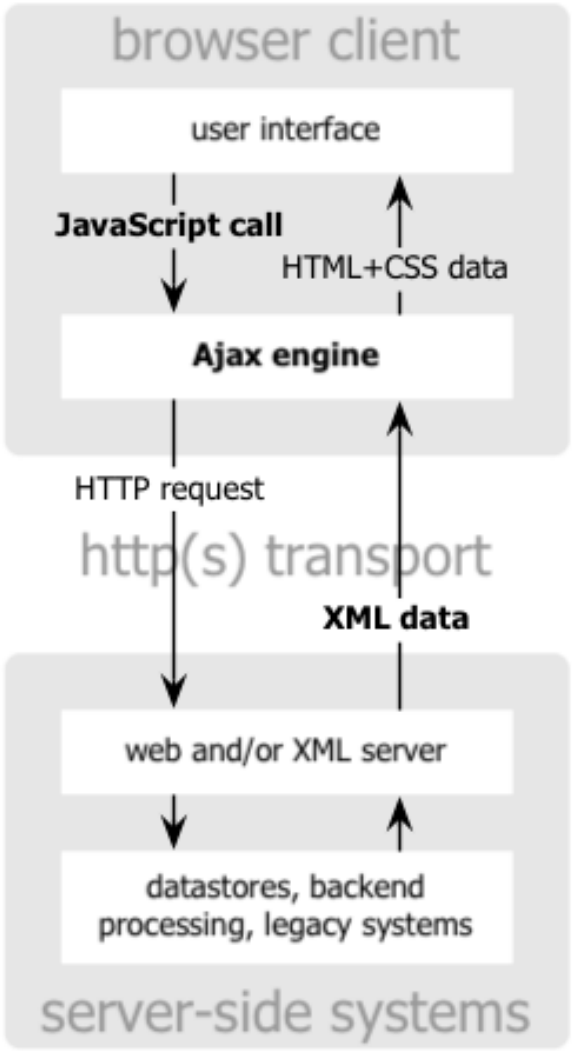
Introducción

- Asynchronous Javascript and XML.
- Tecnología conocida.
- Buzz: Gmail, Web2.0, ...
- Santillana del Mar:
  - Asynchronous, pero también síncrono.
  - XML, pero también JSON.

AJAX vs modelo tradicional



classic



Ajax

Core

- El objeto `XMLHttpRequest` incluye la funcionalidad necesaria.
- No forma parte de ningún estándar
- Realiza peticiones *GET* o *POST*.
- Asíncrono (habitual) o síncrono.
- Al recibir la respuesta invoca el *callback* correspondiente.

Core

- El objeto `XMLHttpRequest` incluye la funcionalidad necesaria.
- No forma parte de ningún estándar
- Realiza peticiones *GET* o *POST*.
- Asíncrono (habitual) o síncrono.
- Al recibir la respuesta invoca el *callback* correspondiente.

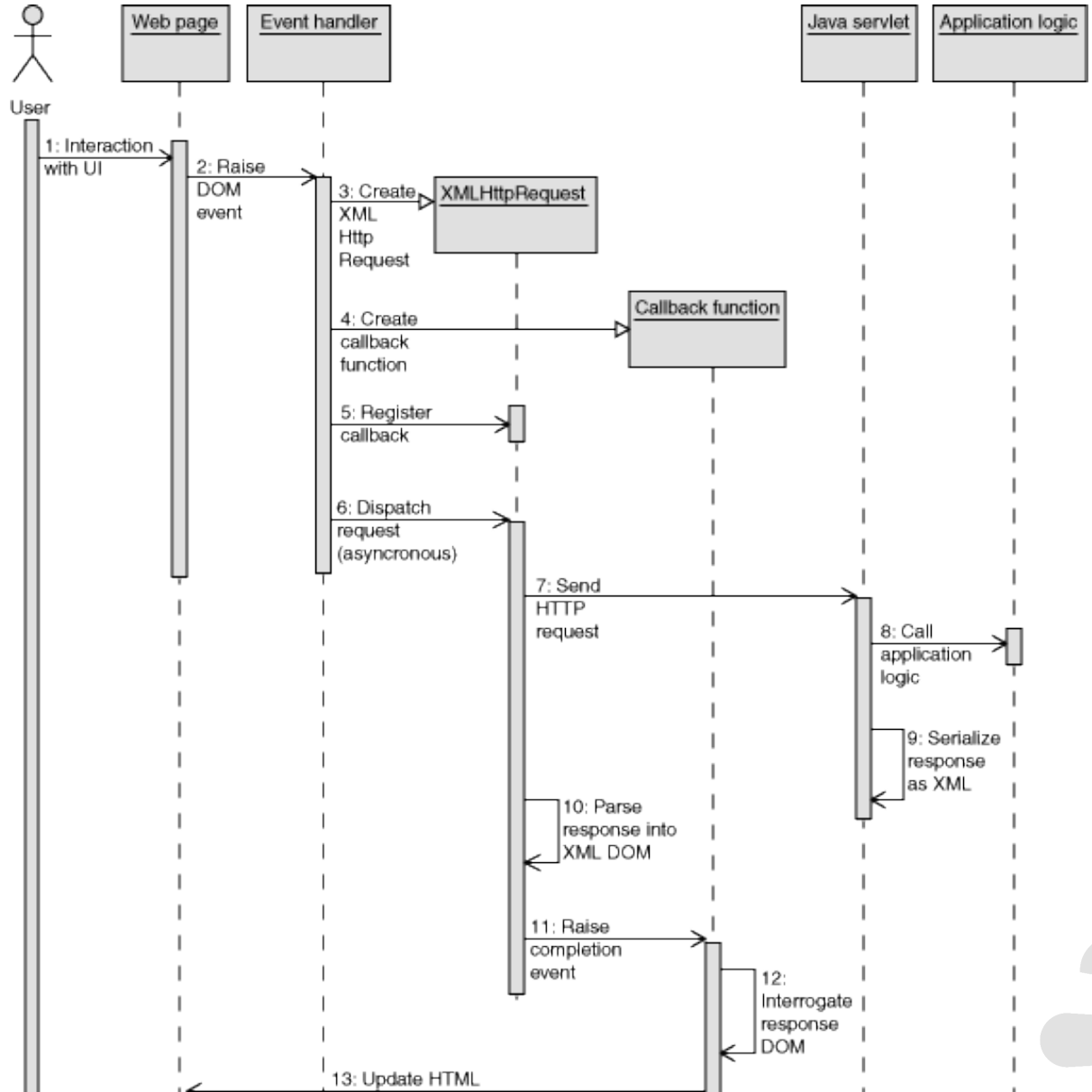
Core

- El objeto `XMLHttpRequest` incluye la funcionalidad necesaria.
- No forma parte de ningún estándar (¿Todavía?)
- Realiza peticiones *GET* o *POST*.
- Asíncrono (habitual) o síncrono.
- Al recibir la respuesta invoca el *callback* correspondiente.

Core

- El objeto `XMLHttpRequest` incluye la funcionalidad necesaria.
- No forma parte de ningún estándar (¿Todavía?)
- Realiza peticiones *GET* o *POST*.
- Asíncrono (habitual) o síncrono.
- Al recibir la respuesta invoca el *callback* correspondiente.

AJAX roundtrip



Cross-browser XMLHttpRequest

- Existen diferencias entre navegadores.

```
function newXMLHttpRequest() {
  var xmlreq = null;

  if (window.XMLHttpRequest) {
      xmlreq = new XMLHttpRequest();
  }
  else if (window.ActiveXObject) {
      try {
        xmlreq = new ActiveXObject("Msxml2.XMLHTTP");
      } catch (e1) {
        try {
          xmlreq = new ActiveXObject("Microsoft.XMLHTTP");
        }
        catch (e2) {
        }
      }
```

Navegadores sin XMLHttpRequest

- Imitar funcionalidad con frames.
- *iframe* en XHTML.
- Idea:
  - Los frames se cargan de forma asíncrona.
  - Usar un frame oculto.
  - Respuesta = contenido del frame.
  - El evento `onload` dispara el callback.

Navegadores sin XMLHttpRequest

- Imitar funcionalidad con frames.
- *iframe* en XHTML.
- Idea:
  - Los frames se cargan de forma asíncrona.
  - Usar un frame oculto.
  - Respuesta = contenido del frame.
  - El evento `onload` dispara el callback.

Petición XMLHttpRequest

- Ejemplo de petición.

```
var req = newXMLHttpRequest();
var handler = getReadyStateHandler(req, hacer_algo);
req.onreadystatechange = handler;

req.open("POST", "una_url", true);

req.setRequestHeader("Content-Type",
                     "application/x-www-form-urlencode

req.send("action=add&item=xxxxxx");
```

readystatechange

- El evento `onreadystatechange` se lanza cada vez que cambia el estado de la petición.

```
req.onreadystatechange = getReadyStateHandler(req, hacer_algo);
```

```
function getReadyStateHandler(req, responseHandler) {
  return function() {
    if (req.readyState == 4) {
      if (req.status == 200) {
        responseHandler(req.responseXML);
      }
      else {
        alert("HTTP error: "+req.status);
      }
    }
  }
}
```

Respuesta XML

- La respuesta es un documento XML.
- Usamos XML DOM.

```
var cart = resp.getElementsByTagName("cart")[0];
if (cart.getAttribute("generated") > lastCartUpdate) {
  // Actualizar el documeno XHTML
  ...
}
```

Respuesta XML

- La respuesta es un documento XML.
- Usamos XML DOM.

```
var cart = resp.getElementsByTagName("cart")[0];
if (cart.getAttribute("generated") > lastCartUpdate) {
  // Actualizar el documeno XHTML
  ...
}
```

Respuesta JSON (I)

- La respuesta también está disponible sin procesar (texto).

```
responseHandler(req.responseText);
```

- Podemos enviar la respuesta en formato libre.
- Podemos enviar al respuesta en formato JSON.
- JSON: JavaScript Object Notation.
- Sintaxis de JSON = sintaxis Javascript para declarar datos:

```
var grupo = {
    nombre: "The Beatles",
    miembros: [
      {
        nombre: "John",
        instrumentos: ["Vocals", "Guitar", "Piano"]
      },
      ...
```

Respuesta JSON (I)

- La respuesta también está disponible sin procesar (texto).

```
responseHandler(req.responseText);
```

- Podemos enviar la respuesta en formato libre.
- Podemos enviar al respuesta en formato JSON.
- JSON: JavaScript Object Notation.
- Sintaxis de JSON = sintaxis Javascript para declarar datos:

```
var grupo = {
    nombre: "The Beatles",
    miembros: [
      {
        nombre: "John",
        instrumentos: ["Vocals", "Guitar", "Piano"]
      },
      ...
```

Respuesta JSON (I)

- La respuesta también está disponible sin procesar (texto).

```
responseHandler(req.responseText);
```

- Podemos enviar la respuesta en formato libre.
- Podemos enviar al respuesta en formato JSON.
- JSON: JavaScript Object Notation.
- Sintaxis de JSON = sintaxis Javascript para declarar datos:

```
var grupo = {
    nombre: "The Beatles",
    miembros: [
      {
        nombre: "John",
        instrumentos: ["Vocals", "Guitar", "Piano"]
      },
      ...
```

Respuesta JSON (I)

- La respuesta también está disponible sin procesar (texto).

```
responseHandler(req.responseText);
```

- Podemos enviar la respuesta en formato libre.
- Podemos enviar al respuesta en formato JSON.
- JSON: JavaScript Object Notation.
- Sintaxis de JSON = sintaxis Javascript para declarar datos:

```
var grupo = {
    nombre: "The Beatles",
    miembros: [
      {
        nombre: "John",
        instrumentos: ["Vocals", "Guitar", "Piano"]
      },
      ...
```

Respuesta JSON (II)

- El servidor manda la respuesta en formato JSON.

```
{
  nombre: "The Beatles",
  miembros: [
    {
      nombre: "John",
      instrumentos: ["Vocals", "Guitar", "Piano"]
    },
    ...
```

- El cliente recupera la repuesta y la convierte en objetos Javascript.

```
var jsonExpression = "(" + req.responseText + ")";
var grupo = eval(jsonExpression);

var nombre = grupo.nombre;
...
```

Respuesta JSON (II)

- El servidor manda la respuesta en formato JSON.

```
{
  nombre: "The Beatles",
  miembros: [
    {
      nombre: "John",
      instrumentos: ["Vocals", "Guitar", "Piano"]
    },
    ...
```

- El cliente recupera la repuesta y la convierte en objetos Javascript.

```
var jsonExpression = "(" + req.responseText + ")";
var grupo = eval(jsonExpression);

var nombre = grupo.nombre;
...
```