AJAX y jQuery

Adolfo Sanz De Diego

Junio 2013

# Acerca de

## Autor

* **Adolfo Sanz De Diego**
  + Correo: [asanzdiego@gmail.com](mailto:asanzdiego@gmail.com)
  + Twitter: [@asanzdiego](http://twitter.com/asanzdiego)
  + Linkedin: <http://www.linkedin.com/in/asanzdiego>
  + Blog: <http://asanzdiego.blogspot.com.es>

## Licencia

* **Este obra está bajo una licencia:**
  + [Creative Commons Reconocimiento-CompartirIgual 3.0](http://creativecommons.org/licenses/by-sa/3.0/es/)
* **El código fuente de los programas están bajo una licencia:**
  + [GPL 3.0](http://www.viti.es/gnu/licenses/gpl.html)

# JavaScript

## Historia

* Lo crea **Brendan Eich en Netscape en 1995** para hacer páginas web dinámicas
* Aparece por primera vez en Netscape Navigator 2.0
* Cada día más usado (clientes web, videojuegos, windows 8, servidores web, etc.)

## El lenguaje

* Orientado a objetos
* Basado en prototipos
* Funcional
* Débilmente tipado
* Dinámico

# Arrays

## ¿Qué son?

* Lista de valores (de cualquier tipo) separados por comas.

a = [1, 2, 3, 4]

## Propiedad y métodos

* Propiedad **length**: longitud del array.
* Método **push**: añadir un elemento al final del array
* Método **pop**: sacar el último elemento del array
* Método **unshift**: añadir un elemento al principio del array
* Método **shift**: sacar el primer elemento del array

# Objects

## ¿Qué son?

* Colección de propiedades separados por comas.
* Una propiedad tiene un nombre y un valor separados por dos puntos.

var objeto = { nombre: "Adolfo", edad: "35"};

## Propiedades

* Podemos acceder directamente o como si fuese un contenedor:

objeto.nombre === objeto[nombre] // true

* Podemos crearlas y destruirlas en tiempo de ejecución

var objeto = {};objeto.nuevaPropiedad = 1; // añadirdelete objeto.nuevaPropiedad; // eliminar

## Tipos

* Todo son objetos excepto: strings, números, booleans, null o undefined
* Strings, números y booleans se comportan como objetos inmutables

# Funciones

## ¿Qué son?

* Son objetos con sus propiedades.
* Se pueden pasar como parámetros a otras funciones.
* Pueden guardarse en variables.
* Son mensajes cuyo receptor es **this**

## This

* Dos funciones permiten manipular el this: **call y apply** que en lo único que se diferencian es en la llamada.

fn.call(thisArg [, arg1 [, arg2 [...]]])

fn.apply(thisArg [, arglist])

## Arguments

* Es un objeto que contiene los parámetros de la función.

function echoArgs() { alert(arguments[0]); // 1}echoArgs(1, 2, 3, 4);

## Constructores y Prototipos

* Los constructores son funciones precedidas con **new** cuyo contexto es el objeto generado.
* Sólo se puede modificar el prototipo de objetos creados con un constructor.
* Modificar un prototipo afecta a todas las instancias anteriores (y futuras).

# Herencia

## Introducción

* En JavaScript no existen las clases tal cual las conocemos en otros lenguajes.
* La aproximación **clásica**:
  + Está más extendida.
  + Y es más fácil de entender.
* La aproximación por **prototipos**:
  + Es más "natural" al lenguaje.
  + Y es más eficiente en el uso de memoria.
* La aproximación **funcional**:
  + Encapsulado público/privado.

## Clásica

function A() { this.a = "Propiedad de A"; this.unMetodo = function() { console.log(this.a); }}function B() { this.b = "Propiedad de B";}// B extends AB.prototype = new A();B.prototype.otroMetodo = function() { console.log(this.b);};var instancia = new B();instancia.unMetodo();instancia.otroMetodo();

## Prototipos

var A = { a: "Propiedad de A", unMetodo: function() { console.log(this.a); }}// B extends Avar B = Object.create(A);B.b = "Propiedad de B";B.otroMetodo = function() { console.log(this.b);};B.unMetodo();B.otroMetodo();

## Funcional

function A() { var a = "Propiedad Privada de A"; var metodoPrivado = function(s) { return s.toUpperCase(); }; var self = {}; self.metodoPublico = function() { return "a="+metodoPrivado(a); }; return self;}function B() { // B extends A var self = A(); self.b = "Propiedad Pública de B"; return self;}var instancia = new B();console.log(instancia.metodoPublico());console.log("a="+instancia.a);console.log("b="+instancia.b);

# DOM

## ¿Qué es DOM?

* Acrónimo de **Document Object Model**
* Es un conjunto de utilidades específicamente diseñadas para **manipular documentos XML, y por extensión documentos XHTML y HTML**.
* DOM transforma internamente el archivo XML en una estructura más fácil de manejar formada por una jerarquía de nodos.

## Tipos de nodos

* Los más importantes son:
  + **Document**: representa el nodo raíz.
  + **Element**: representa el contenido definido por un par de etiquetas de apertura y cierre y puede tener tanto nodos hijos como atributos.
  + **Attr**: representa el atrributo de un elemento.
  + **Text**: almacena el contenido del texto que se encuentra entre una etiqueta de apertura y una de cierre.

## Selección

* JavaScript proporciona funciones para la sección de nodos:

var parrafos = document.getElementsByTagName("p");var parrafo0 = parrafos[0];var nodoSeleccionadoPorId = parrafo0.getElementsById("Id");

* Pero como veremos adelante, es mucho más cómodo utilizar **jQuery**.

## Manipulación

* JavaScript proporciona funciones para la manipulación de nodos:

var nuevoP = document.createElement("p");var texto = document.createTextNode("Este parrafo se ha creado dinámicamente y sustituye al parrafo original");nuevoP.appendChild(texto);var anteriorP = document.body.getElementsByTagName("p")[0];anteriorP.parentNode.replaceChild(nuevoP, anteriorP);

* Pero como veremos adelante, es mucho más cómodo utilizar **jQuery**.

# JSON

## ¿Qué es JSON?

* Acrónimo de **JavaScript Object Notation**.
* Es un subconjunto de la notación literal de objetos de JavaScript.
* Sirve como formato ligero para el intercambio de datos.
* **Su simplicidad ha generalizado su uso, especialmente como alternativa a XML en AJAX**.
* En JavaScript, un texto JSON se puede analizar fácilmente usando la **función eval()**.

## Parse

miObjeto = eval('(' + json\_datos + ')');

* Eval es muy rápido, pero como compila y ejecuta cualquier código JavaScript, las consideraciones de seguridad recomiendan no usarlo.
* Lo recomendable usar las librerías de [JSON.org](http://www.json.org/):
  + [JSON in JavaScript - Explanation](http://www.json.org/js.html)
  + [JSON in JavaScript - Downloads](https://github.com/douglascrockford/JSON-js)

## Ejemplo

{ curso: "AJAX y jQuery", profesor: "Adolfo", participantes: [ { nombre: "Isabel", edad: 35 }, { nombre: "Alba", edad: 15 }, { nombre: "Laura", edad: 10 } ]}

# AJAX

## ¿Qué es AJAX?

* Acrónimo de **Asynchronous JavaScript And XML**.
* Técnica para crear **aplicaciones web interactivas** o RIA (Rich Internet Applications).
* Estas aplicaciones se ejecutan en el cliente, es decir, en el navegador de los usuarios.
* Mientras se mantiene la comunicación asíncrona con el servidor en segundo plano.
* De esta forma es posible realizar **cambios sobre las páginas sin necesidad de recargarlas**.

## Tecnologías AJAX

* AJAX no es una tecnología en sí misma, en realidad, se trata de varias tecnologías independientes que se unen de formas nuevas y sorprendentes.
* Las tecnologías que forman AJAX son:
  + **XHTML y CSS**, como estándares de presentación.
  + **DOM**, para la manipulación dinámica de la presentación.
  + **XML, JSON y otros**, para la la manipulación de información.
  + **XMLHttpRequest**, para el intercambio asíncrono de información.
  + **JavaScript**, para unir todas las demás tecnologías.

## ¿Qué es el XMLHttpRequest?

* El intercambio de datos AJAX entre cliente y servidor se hace mediante el objeto XMLHttpRequest, disponible en los navegadores actuales.
* **No es necesario que el contenido esté formateado en XML**.
* Su manejo puede llegar a ser complejo, aunque librerías como **jQuery** facilitan enormemente su uso.

## Ejemplo

var http\_request = new XMLHttpRequest();var url = "http://example.net/jsondata.php"; // Esta URL debería devolver datos JSON // Descarga los datos JSON del servidor.http\_request.onreadystatechange = handle\_json;http\_request.open("GET", url, true);http\_request.send(null); function handle\_json() { if (http\_request.readyState == 4) { if (http\_request.status == 200) { var json\_data = http\_request.responseText; var the\_object = eval("(" + json\_data + ")"); } else { alert("Ocurrio un problema con la URL."); } http\_request = null; }}

# REST

## ¿Qué es REST?

* REST (Representational State Transfer) es una técnica de arquitectura software para sistemas hipermedia distribuidos como la World Wide Web.
* Es decir, **una URL (Uniform Resource Locator) representa un recurso al que se puede acceder o modificar mediante los métodos del protocolo HTTP (POST, GET, PUT, DELETE)**.
* Ver [Artículos de REST de Enrique Amodeo Rubio (@eamodeorubio)](https://eamodeorubio.wordpress.com/category/webservices/rest/)

## ¿Por qué REST?

* Es **más sencillo** (tanto la API como la implementación).
* Es **más rápido** (peticiones más lijeras que se puede cachear).
* Es **multiformato** (HTML, XML, JSON, etc.).
* Se complementa muy bien con **AJAX**.

## Ejemplo API

* **GET** a *http://myhost.com/person*
  + Devuelve todas las personas
* **POST** a *http://myhost.com/person*
  + Crear una nueva persona
* **GET** a *http://myhost.com/person/123*
  + Devuelve la persona con id=123
* **PUT** a *http://myhost.com/person/123*
  + Actualiza la persona con id=123
* **DELETE** a *http://myhost.com/person/123*
  + Borra la persona con id=123

## Manejo de errores

* **Se pueden utilizar los errores del protocolo HTTP**:
  + 200 OK Standard response for successful HTTP requests
  + 201 Created
  + 202 Accepted
  + 301 Moved Permanently
  + 400 Bad Request
  + 401 Unauthorised
  + 402 Payment Required
  + 403 Forbidden
  + 404 Not Found
  + 405 Method Not Allowed
  + 500 Internal Server Error
  + 501 Not Implemented

# jQuery

## ¿Qué es jQuery?

* Es una librería JavaScript que **simplifica el manejo del DOM** del HTML cliente.
* También **simplifica el manejo de peticiones AJAX** con el servidor.
* Funciona seleccionando uno o varios elementos y ejecutando una acción sobre ellos.
* "**Write less, do more.**"
* La mejor API que he visto [jQuery Quick API Reference](http://oscarotero.com/jquery/)

## Selectores

* Utiliza **los mismos que CSS**, y alguno más propio.

## Eventos

* jQuery está pensado para **recoger y/o lanzar** eventos.
* Estos eventos normalmente son **eventos de ratón, de teclado**.
* También maneja los **eventos de cambio de estado** de algún elemento del DOM.

## CSS

* Podemos cambiar tanto el **atributo style**, como las **clases** de un elemento.

## Efectos

* **Ocultar, mostrar, desvanecer** elementos.
* También podemos hacer **animaciones** cambiando el CSS.
* Ver también [CSS3 transitions jQuery plugin](http://ricostacruz.com/jquery.transit/)

## HTML

* Podemos **cambiar el DOM** del HTML:
  + añadiendo texto tanto al principio como al final de un elemento,
  + cambiando el texto de un elemento,
  + añadiendo un elemento tanto antes como después de un elemento,
  + eliminando elementos.

## Plugins

* Existe un **gran catálogo** de plugins.
* Los plugins **se crean de una forma muy sencilla**.
* Ver [Tutorial Oficial](http://docs.jquery.com/Tutorials:Getting_Started_with_jQuery)

## AJAX

* **Simplifica las peticiones AJAX**, pudiendo manejar su estado.

# jQuery UI

## ¿Qué es jQuery UI?

* Son un conjunto de **componentes visuales**, con [temas personalizables](http://jqueryui.com/themeroller/).
* Las clases CSS se pueden reutilizar. Ver [The jQuery UI CSS Framework](http://docs.jquery.com/UI/Theming/API)
* Las demos están muy bien, con muchos y muy buenos ejemplos [jQuery UI Demos](http://jqueryui.com/demos/)

## Componentes

* **Interactions**: draggable, droppable, resizable, selectable, sortable
* **Widgets**: accordion, autocomplete, button, datepicker, dialog, menu, progressbar, slider, spinner, tabs, tooltip
* **Efects**: blind, bounce, clip, drop, explode, fade, fold, highlight, pulsate, scale, shake, slide, transfer

# jQuery Mobile

## ¿Qué es jQuery Mobile?

* Son componentes pensados para **aplicaciones móviles**.
* Están pensados para el manejo de **eventos táctiles**.

## Ejemplos

* Toda la documentación del site está hecha con el propio framework.
* Ver [jQuery Mobile](http://jquerymobile.com/)
* Ver [jQuery Mobile Documentation](http://jquerymobile.com/demos/)