AJAX y jQuery

Adolfo Sanz De Diego

Junio 2013

Contents

1 Acerca de **7**

1.1 Autor 7

1.2 Licencia 7

2 JavaScript **9**

2.1 Historia 9

2.2 El lenguaje 9

3 Arrays **11**

3.1 ¿Qué son? 11

3.2 Propiedad y métodos 11

4 Objects **13**

4.1 ¿Qué son? 13

4.2 Propiedades 13

4.3 Tipos 13

5 Funciones **15**

5.1 ¿Qué son? 15

5.2 This 15

5.3 Arguments 15

5.4 Constructores y Prototipos 16

6 Herencia **17**

6.1 Introducción 17

6.2 Clásica 17

6.3 Prototipos 18

6.4 Funcional 18

7 DOM **21**

7.1 ¿Qué es DOM? 21

7.2 Tipos de nodos 21

7.3 Selección 21

7.4 Manipulación 22

8 JSON **23**

8.1 ¿Qué es JSON? 23

8.2 Parse 23

8.3 Ejemplo 23

9 AJAX **25**

9.1 ¿Qué es AJAX? 25

9.2 Tecnologías AJAX 25

9.3 ¿Qué es el XMLHttpRequest? 26

9.4 Ejemplo 26

10 REST **27**

10.1 ¿Qué es REST? 27

10.2 ¿Por qué REST? 27

10.3 Ejemplo API 27

10.4 Manejo de errores 28

11 jQuery **29**

11.1 ¿Qué es jQuery? 29

11.2 Selectores 29

11.3 Eventos 29

11.4 CSS 30
11.5 Efectos 30
11.6 HTML 30
11.7 Plugins 30
11.8 AJAX 30

12 jQuery UI **31**
12.1 ¿Qué es jQuery UI? 31
12.2 Componentes 31

13 jQuery Mobile **33**
13.1 ¿Qué es jQuery Mobile? 33
13.2 Ejemplos 33

Chapter 1

Acerca de

1.1 Autor

- **Adolfo Sanz De Diego**
  - Correo: asanzdiego@gmail.com
  - Twitter: @asanzdiego
  - Linkedin: http://www.linkedin.com/in/asanzdiego
  - Blog: http://asanzdiego.blogspot.com.es

1.2 Licencia

Chapter 2

JavaScript

2.1 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.)

2.2 El lenguaje

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

Chapter 3

Arrays

3.1 ¿Qué son?

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

```
a = [1, 2, 3, 4]
```

3.2 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

Chapter 4

Objects

4.1 ¿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"
};
```

4.2 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; // añadir
delete objeto.nuevaPropiedad; // eliminar
```

4.3 Tipos

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

Chapter 5

Funciones

5.1 ¿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**

5.2 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])
```

5.3 Arguments

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

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

5.4 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).

Chapter 6

Herencia

6.1 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.

6.2 Clásica

```
function A() {
    this.a = "Propiedad de A";
    this.unMetodo = function() {
        console.log(this.a);
    }
}

function B() {
```

```
    this.b = "Propiedad de B";
}

// B extends A
B.prototype = new A();

B.prototype.otroMetodo = function() {
    console.log(this.b);
};

var instancia = new B();
instancia.unMetodo();
instancia.otroMetodo();
```

6.3 Prototipos

```
var A = {
    a: "Propiedad de A",
    unMetodo: function() {
        console.log(this.a);
    }
}

// B extends A
var B = Object.create(A);
B.b = "Propiedad de B";

B.otroMetodo = function() {
    console.log(this.b);
};

B.unMetodo();
B.otroMetodo();
```

6.4 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);
```

Chapter 7

DOM

7.1 ¿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.

7.2 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.

7.3 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**.

7.4 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 sustituy
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**.

Chapter 8

JSON

8.1 ¿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()**.

8.2 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:
  - JSON in JavaScript - Explanation
  - JSON in JavaScript - Downloads

8.3 Ejemplo

```
{
    curso: "AJAX y jQuery",
```

```
    profesor: "Adolfo",
    participantes: [
        { nombre: "Isabel", edad: 35 },
        { nombre: "Alba", edad: 15 },
        { nombre: "Laura", edad: 10 }
    ]
}
```

Chapter 9

AJAX

9.1 ¿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**.

9.2 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.

9.3 ¿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.

9.4 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;
  }
}
```

Chapter 10

REST

10.1 ¿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)

10.2 ¿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**.

10.3 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

10.4 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

Chapter 11

jQuery

11.1 ¿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

11.2 Selectores

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

11.3 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.

11.4 CSS

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

11.5 Efectos

- **Ocultar, mostrar, desvanecer** elementos.
- También podemos hacer **animaciones** cambiando el CSS.
- Ver también CSS3 transitions jQuery plugin

11.6 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.

11.7 Plugins

- Existe un **gran catálogo** de plugins.
- Los plugins **se crean de una forma muy sencilla**.
- Ver Tutorial Oficial

11.8 AJAX

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

Chapter 12

jQuery UI

12.1 ¿Qué es jQuery UI?

- Son un conjunto de **componentes visuales**, con temas personalizables.
- Las clases CSS se pueden reutilizar. Ver The jQuery UI CSS Framework
- Las demos están muy bien, con muchos y muy buenos ejemplos jQuery UI Demos

12.2 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

Chapter 13

jQuery Mobile

13.1 ¿Qué es jQuery Mobile?

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

13.2 Ejemplos

- Toda la documentación del site está hecha con el propio framework.
- Ver jQuery Mobile
- Ver jQuery Mobile Documentation