Universidad de las Ciencias Informáticas

Facultad X

Librerías de la programación en el Cliente (JavaScript). Incorporación de estas tecnologías al Sistema de Administración de Contenidos Drupal.

Trabajo de Diploma para optar por el título de Ingeniería en Ciencias Informáticas.

Autor(es): Ailen Moreira Padrón
Yoan Abreu Cabrera
Tutor: Ing. William Azcuy Morales

Ciudad de La Habana, Cuba

Junio, 2007

Declaración de Autoría

Declaramos ser autores de la presente tesis y reconocemos a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo.

Para que así conste firmo la presente a los ____ días del mes de _____ del año _____.

_____

Ailen Moreira Padrón

_____

Yoan Abreu Cabrera

_____

Ing. William Azcuy Morales

DATOS DE CONTACTO

Ing. William Azcuy Morales, Profesor de Programación Facultad X, Universidad de Ciencias Informáticas. Carretera a San Antonio de los Baños Km. 2 ½, Torrens, Boyeros, Ciudad de La Habana. Cuba. william@uci.cu. Ingeniero Informático, Instituto Superior Politécnico “José Antonio Echeverría” CUJAE, 2002.

Culminó sus estudios con índice académico de 5.16 puntos, obteniendo Título de Oro. Ha participado en un gran número de eventos nacionales e internacionales, obteniendo una Distinción Relevante en el Forum Nacional de Ciencia y Técnica y el sello Forjadores del Futuro en el año 2002.

En ese mismo año ingresa a la Universidad de las Ciencias Informática, siendo fundador de la misma. En la universidad ha participado en varios proyectos, entre ellos algunos para la República Bolivariana de Venezuela. Ha sido tutor de nueve Trabajos de Diploma.

Actualmente es Jefe de Despliegue del proyecto Intranet de PDVSA.

Dedicatoria

Ailen

A mi mamá por ser mi guía y mi mayor inspiración, te quiero con todo mi corazón.

A mis hermanas por darme la esperanza por la sonrisa.

A mi padre por su tenacidad.

A mi tío del alma, Andrés, por confiar en mí y darme su amor incondicional.

Yoan

A mi mamá por estar siempre a mi lado y apoyarme en todo momento, por todo el cariño que me ha brindado y por ser mi guía, un beso grande.

A mi papá por haberme querido tanto en todos estos años y por darme tan buenos consejos, eres mi ejemplo a seguir.

A mi hermano Iván, gracias por haberme apoyado tanto y haberme querido de esa manera, sin ti, nada de esto hubiese sido posible, te quiero con todo mi corazón.

Agradecimientos

A nuestro tutor William Azcuy, por su dedicación, su entrega aun cuando no estaba cerca de nosotros no nos abandonó, nos dió seguridad y depositó su confianza en nosotros, a usted por acompañarnos. Nuestros más profundos agradecimientos a todas aquellas personas que de una manera u otra contribuyeron en nuestra formación como profesionales durante la etapa de estudiante universitario, especialmente a nuestros profesores.

Ailen

A Dora y Erasmin, por su apoyo y su cariño.

A mi familia por estar ahí siempre.

A mis amigas Carmen, Marelis, Dainelys, Lisset, Yaumara y Liana por ser tan especiales.

A mis amigos Abel, Rubier, Yoan, Livan, Ionian por acompañarme y cuidarme.

A mis Compañeras del Apto por su alegría.

A Daciel por ser mi arcoiris en medio de la tormenta.

A Fernando y Josefina por su preocupación y soporte.

A Graciela por su paciencia y comprensión.

A Héctor y Plasencia que son mi ejemplo, que lideran sin perder la ternura.

Yoan

A Oscar, a Marlene gracias por estar ahí cuando los necesité.

A mis amigos Rubier, Geidy, Dayris, Ernesto, Annalie, Carmen Yadira, Ailen, Layla, Temis, Lisset, Ediel, Eikel gracias por todo el apoyo y soporte que me dieron durante estos años; sin Uds no hubiese sido igual. Gracias por su amistad.

A mi compañera de tesis Ailen, gracias por todo, por los buenos y malos momentos, al fin llegamos…

A toda mi familia donde quiera que esté, gracias por apoyarme siempre.

A mi profesor de Cálculo, Dr. Antonio Otero, donde quiera que esté, gracias por su amistad y por haberme ayudado en mi formación como profesional.

A mi profe de ingles Matilde, muchas gracias por todos estos años, por su preocupación y cariño.

A mis compañeros de aula, fue genial compartir estos años con Uds.

Resumen

El mundo actual está inmerso en un desarrollo acelerado de las TICS y para ello, parte de implementar Sistemas Operativos que lo sustenten. Hoy contamos con dos paradigmas, el Software propietario y el Software libre, nuestro país está optando por migrar hacia el Software libre, en esta misión nuestra universidad abanderada, dentro de los proyectos que se realizan está el de portales, una de las problemáticas que se presentan es la necesidad de documentar librerías de la programación en cliente en el lenguaje JavaScript y su integración al CMS Drupal. A darle solución a esta problemática va encaminado este trabajo investigativo para ello, se realizó una búsqueda exhaustiva de varias librerías con el fin de determinar cual será más factible. Aún cuando se tiene predefinida una herramienta a la que va encaminado el resultado final de la investigación, en este trabajo se argumenta otras que se utilizan en el mundo de la programación Web con el fin de que el usuario final las conozca así como sus ventajas y desventajas con respecto a Drupal. La librería que se propone como resultado de la investigación es JQuery, puesto que permite ampliar las extensiones de Drupal, mejorando así sus funcionalidades y es tan sencillo de integrar como poner el archivo jquery.js en las páginas donde necesitemos utilizar dicha librería.

Palabras Claves:
Librería, JavaScript, Herramientas, Drupal y JQuery.

Índice

DATOS DE CONTACTO II

INTRODUCCIÓN 1

1 CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA 5

1.1 INTRODUCCIÓN 5
1.2 HERRAMIENTAS UTILIZADAS EN LA PROGRAMACIÓN WEB 5
1.2.1 Herramienta de Autor 5
1.2.2 Inspector DOM 6
1.2.3 Sajax 6
1.2.4 Dojo toolkit 6
1.2.5 Macromedia Dreamweaver 7
1.3 SISTEMAS DE GESTIÓN DE CONTENIDOS 7
1.3.1 Drupal 8
1.3.2 Drupal por dentro 9
1.4 JAVASCRIPT 14
1.5 DEFINIENDO AJAX 16
1.5.1 Características de AJAX 16
1.5.2 Uso de Ajax 18
1.5.3 Ventajas de Ajax 19
1.5.4 Aplicaciones que utilizan Ajax 19
1.6 JSON 19
1.6.1 Utilidades de JSON 20
1.6.2 ¿Qué es la notación objeto en JavaScript? 20
1.6.3 Ventajas de JSON 21
1.7 LIBRERÍAS 22
1.7.1 Librería Xajax 23
1.7.2 Librería DOJO Toolkit 23
1.7.3 Librería XOAD 24

1.7.4 Librería Jmol.js 24
1.7.5 Librería Ajax Auto Suggest 24
1.7.6 Librería Interfaz de Usuario de Yahoo (YUI) 24
1.7.7 Librería Ext 25
1.7.8 Librería JQuery 25
1.8 DESCRIPCIÓN DE LAS LIBRERÍAS 25
1.8.1 Xajax 25
1.8.2 DOJO Toolkit 27
1.8.3 Ajax Auto Suggest 28
1.8.4 Librería Interfaz de Usuario de Yahoo (YUI) 30
1.8.5 JQuery 32
2 CAPÍTULO 2: ARGUMENTACIÓN DE LA PROPUESTA 35
2.1 INTRODUCCIÓN 35
2.2 ¿CÓMO OBTENERLA? 35
2.3 LO BÁSICO DE JQUERY 35
2.3.1 Hacer que algo suceda con un clic 36
2.3.2 Modificando la apariencia 37
2.4 EFECTOS ESPECIALES 37
2.4.1 Encadenamiento 37
2.5 EL NÚCLEO DE JQUERY 38
2.6 SELECTORES DE CSS 43
2.6.1 Soportados pero diferentes 44
2.6.2 No soportados 44
2.6.3 Selectores XPath 44
2.6.4 Selectores de ejes soportados 45
2.6.5 Predicados soportados 45
2.6.6 Predicados soportados con modificaciones 45
2.6.7 Selectores personalizados 46
2.6.8 Selectores de formularios 46
2.7 USANDO CSS Y XPATH AL MISMO TIEMPO 47

2.8 ATRIBUTOS 48
2.8.1 AddClass (class) 48
2.9 TRAVERSING 53
2.10 MANIPULATION 62
2.11 CSS 69
2.12 EFECTOS 72
2.12.1 Animaciones (Parámetros, velocidad, facilidad, callback) 72
2.12.2 fadeIn (speed, callback) 73
2.12.3 fadeout (speed, callback) 74
2.12.4 fadeTo (speed, opacity, callback) 74
2.12.5 hide () 75
2.12.6 hide (speed, callback) 75
2.12.7 show () 76
2.12.8 show (speed, callback) 76
2.12.9 slideDown (speed, callback) 77
2.12.10 slideToggle (speed, callback) 77
2.12.11 slideUp (speed, callback) 78
2.13 AJAX 79
2.13.1 Características de $.ajax () 79
2.14 PLUGGINS 89
2.15 DRUPAL Y JQUERY 90
2.15.1 Editor de contenidos. 90
2.15.2 Ajax. 92
2.15.3 Drupal 5. 95

CONCLUSIONES GENERALES 96

RECOMENDACIONES 97

BIBLIOGRAFÍA 98

GLOSARIO 101

INTRODUCCIÓN

Dado el creciente auge de las empresas cubanas en el desarrollo de nuevas tecnologías y la producción de software, se pretende ilustrar en este trabajo una alternativa de utilización de tecnologías y herramientas que sean más factibles para la creación de sistemas haciendo uso de software libre, con el objetivo de maximizar y economizar dicha producción.

Con esta investigación, se darán a conocer nuevas variantes de solución a los problemas que hoy presenta nuestro país llegado el momento de determinar qué herramienta o librería utilizar en el momento de realizar una aplicación Web dentro de una determinada plataforma; como ejemplo se puede mencionar una de ellas, Drupal, que es a la cual va enfocada este proyecto investigativo.

Drupal es un sistema de gestión de contenido (Content Management System, en inglés, abreviado CMS) que recoge muchas de las ventajas y suaviza, en gran medida, las limitaciones que antes se vislumbraban con la utilización del software propietario.

Debido a las actuales tendencias que existen en las empresas y organizaciones dedicadas a desarrollar portales Web más dinámicos y seguros, la empresa de software cubana y nuestra Universidad, como puntal del desarrollo de software, se han dado a la tarea de investigar las distintas plataformas que hoy en día se acercan más a las solicitudes del cliente. La utilización de librerías y herramientas de JavaScript orientadas del lado del cliente se han ganado su posición en esta rama de la Informática, por lo que se puede puntualizar que el **problema de investigación** a resolver es:

La falta de documentación de las librerías de la programación en cliente en JavaScript para su integración con el CMS Drupal.

A partir de esta problemática planteada se infiere que en la actualidad la utilización de herramientas y técnicas de programación para realizar Portales Web u otras aplicaciones es muy variable, y debido al creciente desarrollo y creación de nuevas herramientas, las empresas se ven obligadas a actualizar y realizar de manera más eficaz y rápida, los proyectos que se les asignan, por lo cual están forzadas a realizar una constante investigación sobre software, técnicas, herramientas, librerías y muchas otras aplicaciones y soluciones para no quedar rezagadas con respecto a la competencia que existe por parte de otras empresas que se dedican a su mismo perfil de producción. Tampoco se conoce cómo se debe dar una solución rápida y eficiente, puesto que el cliente que es el principal consumidor está en constante búsqueda de lo nuevo y más diverso. Visto esto y determinado el principal problema, debe argumentarse

que con la realización de esta investigación se pretende avanzar un poco más en este largo camino que se presenta continuamente ante nuestros ojos; que no es más que lograr la eficiencia, además de apoyar y ayudar en futuras ocasiones a distintas personas o entidades que se encuentren en procesos similares a este, y no encuentren una salida óptima.

Antecedentes:

Antes de formular alguna asociación o pensamiento que pudiese conducir a confusiones, se debe aclarar que aunque las librerías y herramientas de JavaScript se han utilizado desde hace algún tiempo y son algo conocidas, Drupal es bastante joven y se debe aclarar que es por dicha razón por la cual esta investigación está orientada a determinar cual es la librería que presenta las características que más se adecuan a este CMS, para la futura utilización que se le pretende dar a este software una vez concluida su etapa de elaboración.

Por lo que se puede deducir que **el aporte práctico** de este proyecto investigativo es orientar y encontrar la solución adecuada para la utilización de una determinada librería que se acerque a los servicios que pretende dar el software final, construido sobre una plataforma de Drupal, la cual va a utilizar tecnología que esté del lado del cliente y en lenguaje JavaScript.

En tal sentido se puede determinar que **el objeto de estudio** que se presenta es:

El perfeccionamiento y la optimización de la plataforma Drupal, mediante el uso de potentes librerías de JavaScript que se encuentren encaminadas a satisfacer las necesidades del cliente, en un ambiente previamente determinado.

Del Objeto de estudio analizado, se puede definir que **el Campo de Acción** es el que sigue:

Propuestas de librerías novedosas que nos ayuden a mejorar el desarrollo de los portales Web.

Una vez que se haya realizado una búsqueda informativa exhaustiva de la información disponible en la red, Internet, revistas, conferencias magistrales u otro tipo de banco de información se puede definir que **el objetivo general** de este proyecto investigativo es:

Realizar una propuesta de librería(s) de JavaScript que se encuentre(n) orientada(s) al cliente, y que sea(n) posible(s) de utilizar en Drupal con la finalidad de aumentar sus servicios y prestaciones en la creación de proyectos productivos.

Para complementar este Objetivo General se deben plantear los siguientes **objetivos específicos**:

Viabilizar una mejor comprensión de la información que se pretende utilizar en la plataforma, con la finalidad de que se le dé un uso adecuado y provechoso.

Documentar la(s) librería(s) de JavaScript que sea(n) capaz (ces) de satisfacer las necesidades el cliente tanto en el producto final, así como en las posibilidades de modificar dicho producto en caso que sea necesario.

Se plantearon las siguientes **preguntas científicas**:

¿Que características deben tener las librerías de JavaScript que cumplan con las necesidades del proyecto?

¿Cuales son las principales propiedades y ventajas del lenguaje JavaScript?

Con esta finalidad, se deben plantear un conjunto de tareas que permitan la puesta en práctica de sistemas que utilicen Drupal como plataforma de trabajo, y que a su vez, estas utilicen librerías adecuadas en código JavaScript y orientadas al cliente. Dichas **tareas** son:

Estudiar detalladamente que características se requiere(n) de la(s) librería(s) que se proponga(n) para que cumpla(n) con las exigencias del proyecto.

Estudiar el lenguaje de programación JavaScript en el cliente, sus ventajas y su impacto en la actualidad en la programación Web, así como herramientas de la programación Web y en especial Drupal.

Llevar a cabo una profunda investigación cuando se haya(n) seleccionado las librería(s) a utilizar.

El **método** utilizado en el desarrollo de la investigación es el **método teórico**, dentro del mismo el analítico- sintético, con el cual buscamos las particularidades del lenguaje de programación JavaScript, así como las características de sus librerías, algunas de las herramientas que lo utilizan y en particular Drupal. Con el resultado del procesamiento de esta información lograr la propuesta de la integración de una de las librerías al CMS Drupal.

Este trabajo investigativo estará conformado esencialmente por dos capítulos, cada uno conformado por Introducción, Desarrollo y Conclusiones. En el primero de estos, titulado Fundamentación teórica, se hará referencia al estado del arte del tema que se debe investigar, tanto a nivel internacional, nacional, como de la Universidad; en él se explicarán los principales conceptos y fundamentos de la investigación, las técnicas, metodologías y otras herramientas que se utilizan en la actualidad para dar solución a problemas del mismo carácter. Además, se hará una aproximación al lenguaje de programación JavaScript así como a algunas de sus librerías y herramientas más utilizadas en la Web. Se hará una descripción detallada de las librerías y herramientas mencionadas anteriormente a través de la cual se podrá identificar cuál es la que reúne las características necesarias para darle respuesta al objetivo principal de esta investigación que no es más que seleccionar una librería para argumentar una propuesta a utilizar en Drupal.

Esto servirá de preámbulo al capítulo dos, “Argumentación de la Propuesta” en el que se dará una explicación bien detallada de la librería escogida y se hablará de su integración con Drupal como paso final en el completamiento de los objetivos.

1 CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA

1.1 Introducción

En este capítulo se analizan aspectos de carácter teórico, que es sumamente necesario investigar puesto que es importante tener los argumentos para hacer una propuesta que cumpla con las necesidades planteadas y de una solución lo mas eficiente posible a nuestro problema.

En la actualidad el mundo se debate principalmente entre dos paradigmas de desarrollo, el software propietario y el software libre. Cuba en medio de un proceso de informatización de la sociedad opta por lograr la migración completa al segundo mencionado, que permite acceder al código fuente haciéndole los cambios que requiera el usuario, sean cuales quiera sus intenciones con el mismo, con la única condición de redistribuir el software libremente licenciado bajo algún tipo de licencia de software libre que beneficie a la comunidad. Una vez que un producto de software libre ha empezado a circular, rápidamente está disponible a un costo muy bajo o sin costo alguno. Al mismo tiempo, su utilidad no decrece. Esto significa que el software libre se puede caracterizar como un bien público en lugar de un bien privado. La mayoría del software libre se produce por equipos internacionales que cooperan a través de la libre asociación. Los equipos están típicamente compuestos por individuos con una amplia variedad de motivaciones, que permite ampliar las soluciones a problemas de la vida diaria, que serán solucionados por quienes los sufren. Esto abre una puerta a la teoría de la masividad del conocimiento, todo el que tenga acceso al código fuente con un poco de estudio puede satisfacer sus necesidades.

Seguidamente se abordan aspectos de carácter teórico relacionados con las tipicidades del lenguaje de programación JavaScript del lado del cliente, así como las características de las diferentes librerías utilizadas en la implementación de este lenguaje y herramientas para la programación Web.

1.2 Herramientas utilizadas en la programación Web

1.2.1 Herramienta de Autor

Las herramientas de autor o metamedios son aplicaciones que permiten un trabajo multimedia y constructivista para generar un entorno de aprendizaje dinámico, dentro de las funcionalidades que este tipo de herramientas presentan se puede destacar la posibilidad de crear actividades o pequeñas aplicaciones desde la misma herramienta. Las herramientas de autor proveen generalmente módulos

desde los cuáles se pueden organizar actividades o se puede interconectar pequeños componentes y se pueden adecuar a los objetivos, los conocimientos y habilidades que se busque desarrollar por parte del autor. Las herramientas de autor más básicas son aquellas que solamente permiten un conjunto limitado de acciones para que el usuario interactúe con el sistema, como por ejemplo, mapas Web sensibles en conjuntos de páginas HTML con JavaScript. Las herramientas más avanzadas incluyen lenguajes de programación. (WIKIMEDIA FOUNDATION 2006)

1.2.2 Inspector DOM

El inspector DOM es una de las herramientas de desarrollo Web incluidas en la Suite Mozilla así como en otros navegadores Web. Su principal propósito es inspeccionar el Document Object Model tree de los documentos basados en HTML y XML. Un nodo DOM puede seleccionarse desde la estructura, o cliqueando en el cromado del explorador. Las hojas de estilo de los documentos y los objetos JavaScript se pueden seleccionar desde el árbol. El elemento activo se ilumina con un borde rojo parpadeante, el cual es útil para depurar el código CSS. Además de inspeccionar, también es posible editar, aunque no permite un editor de texto enriquecido. (WIKIMEDIA FOUNDATION 2007c)

1.2.3 Sajax

Sajax (del inglés "Simple Ajax Toolkit", "Colección de herramientas simples de AJAX") es una herramienta de código abierto diseñada para ayudar a los sitios Web que usen AJAX framework (también conocido como XMLHTTPRequest). Permite al programador llamar a funciones PHP, Perl o Python desde su página Web por medio de JavaScript sin necesidad de forzar una actualización de la página en el navegador. (WIKIMEDIA FOUNDATION 2007e)

1.2.4 Dojo toolkit

Dojo es más que un conjunto de herramientas, es un Framework que contiene APIs y widgets (controles) para soportar el desarrollo de aplicaciones Web, abstracción de eventos, almacenamiento de APIs en el cliente, e interacción de APIs con AJAX. Dojo resuelve asuntos de usabilidad comunes como son la navegación y detección del browser, soportar cambios de URL en la barra de URLs para luego regresar a ellas (bookmarking), y la habilidad de degradar cuando AJAX/JavaScript no es completamente soportado en el cliente. Dojo es llamado “la navaja suiza del ejército de las bibliotecas del JavaScript”. Proporciona

una gama más amplia de opciones en una sola biblioteca y hace un trabajo muy bueno que apoya los nuevos y viejos Browsers. (WIKIMEDIA FOUNDATION 2007a)

1.2.5 Macromedia Dreamweaver

Macromedia Dreamweaver es un editor WYSIWYG de páginas Web creado por Macromedia (actualmente Adobe Systems). Es el programa de este tipo más utilizado en el sector del diseño y la programación Web, por sus funcionalidades, su integración con otras herramientas como Macromedia Flash y, recientemente, por su soporte de los estándares del World Wide Web Consortium. Tiene soporte tanto para edición de imágenes como para animación a través de su integración con otras herramientas.

La gran ventaja de este editor sobre otros es su gran poder de ampliación y personalización del mismo, puesto que este programa, sus rutinas (como la de insertar un hipervínculo, una imagen o añadir un comportamiento) están hechas en JavaScript-C, que no es más que un lenguaje cruzado, para la interpretación de objetos mediante Gecko-JavaScript invocado en C++, lo que permite la POO para el DOM de una aplicación, lo que le ofrece una gran flexibilidad en estas materias. Esto hace que los archivos del programa no sean instrucciones de C++ sino, rutinas de JavaScript que hace que sea un programa muy fluido, que todo ello hace, que programadores y editores Web hagan extensiones para su programa y lo ponga a su gusto. Dreamweaver permite al usuario utilizar la mayoría de los navegadores Web instalados en su ordenador para previsualizar las páginas Web. También dispone de herramientas de administración de sitios dirigidas a principiantes.

Dreamweaver oculta el código HTML de cara al usuario, desarrolladores Web critican esta propuesta ya que crean páginas HTML más largas de lo que solían ser al incluir mucho código inútil, lo cual va en detrimento de la ejecución de las páginas en el navegador Web, la aplicación facilita en exceso el diseño de las páginas mediante tablas.

1.3 Sistemas de Gestión de Contenidos

Un Sistema de gestión de contenidos permite la creación y administración de contenidos principalmente en páginas Web.

Consiste en una interfaz que controla una o varias bases de datos donde se aloja el contenido del sitio. El sistema permite manejar de manera independiente el contenido por una parte y el diseño por otra. Así, es posible manejar el contenido y darle en cualquier momento un diseño distinto al sitio sin tener que darle formato al contenido de nuevo, además de permitir la fácil y controlada publicación en el sitio a varios

editores. Un ejemplo clásico es el de editores que cargan el contenido al sistema y otro de nivel superior que permite que estos contenidos sean visibles a todo público.

1.3.1 Drupal

Drupal es un sistema de administración de contenidos para sitios Web. Permite publicar artículos, imágenes u otros archivos y servicios añadidos como foros, encuestas, votaciones, blogs y administración de usuarios y permisos. Drupal es un sistema dinámico: en lugar de almacenar sus contenidos en archivos estáticos en el sistema de ficheros del servidor de forma fija, el contenido textual de las páginas y otras configuraciones son almacenados en una base de datos y se editan utilizando un entorno Web incluido en el producto.

A pesar de que empezó como un pequeño BBS Drupal ha llegado a ser mucho más que sólo un portal de noticias gracias a su arquitectura flexible. Drupal se compone de una infraestructura base y un conjunto de módulos que ofrecen un amplio conjunto de funciones, incluyendo sistemas de comercio electrónico, galerías de fotos y administración de listas de correo electrónico. Es posible añadir módulos de terceros para modificar el comportamiento de Drupal u ofrecer nuevas funciones. (WIKIMEDIA FOUNDATION 2007b)

Drupal se usa, entre otros, en intranets de compañías, enseñanza en línea, comunidades de arte y administración de proyectos. Muchos piensan que la relevancia de Drupal en las comunidades de usuarios es lo que lo hace destacarse de la competencia, muy conocido por la calidad de su código y por la seguridad que brinda, es estable y de actualización continua, configuración sencilla, instalación ágil, importante cantidad de módulos y temas, excepcional documentación. A continuación algunos sitios que utilizan Drupal:

Returning Heroes Home (http://returningheroeshome.org). Un sitio dedicado a los esfuerzos de varias familias norteamericanas por el regreso de sus hijos y familiares de la guerra en el medio oriente.

University of Louisville, News & Forums (http://uofl.info). Sección informativa del sitio de la Universidad de Louisville.

Active Living Network (http://www.activeliving.org). Portal que promueve la vida saludable.

Alicante Rural (http://www.alicanterural.es). Web para el desarrollo del Turismo Rural en la provincia de Alicante, España.

NBA News (http://nbasketnews.com). Sitio informativo de la NBA.

Estos son solo algunos de los muchos sitios que se encuentran publicados en www.drupalsites.net y que nos dan una idea de la aceptación que ha tenido este CMS y la gran variedad de temas que abordan los sitios realizados con el mismo.

Drupal se distribuye bajo la licencia GNU GPL, y por lo tanto es software libre. La licencia GNU GPL posibilita la modificación y redistribución del software, pero únicamente bajo esa misma licencia. Y añade que si se reutiliza en un mismo programa código "A" licenciado bajo licencia GNU GPL y código "B" licenciado bajo otro tipo de licencia libre, el código final "C", independientemente de la cantidad y calidad de cada uno de los códigos "A" y "B", debe estar bajo la licencia GNU GPL.

En la práctica esto hace que las licencias de software libre se dividan en dos grandes grupos, aquellas que pueden ser mezcladas con código licenciado bajo GNU GPL (y que inevitablemente desaparecerán en el proceso, al ser el código resultante licenciado bajo GNU GPL) y las que no lo permiten al incluir mayores u otros requisitos que no contemplan ni admiten la GNU GPL y que por lo tanto no pueden ser enlazadas ni mezcladas con código gobernado por la licencia GNU GPL.

Hasta su versión 4 Drupal utilizó algunas librerías propias para integrar JavaScript, y recientemente Ajax. Las principales funciones se distribuían entre el archivo drupal.js y unos cuantos otros pero poco a poco empezaron a surgir conflictos en espacios cuando otros módulos usaban sus propias librerías, por lo tanto se hace necesario estandarizar en una sola librería JavaScript.

1.3.2 Drupal por dentro

El diseño de Drupal es especialmente idóneo para construir y gestionar comunidades en Internet. No obstante, su flexibilidad y adaptabilidad, así como la gran cantidad de módulos adicionales disponibles, hace que sea adecuado para realizar muchos tipos diferentes de sitio Web. El sitio principal de desarrollo y coordinación de Drupal es drupal.org, en el que participan activamente varios miles de usuarios de todo el mundo.

1.3.2.1 Características generales

Ayuda on-line: Un robusto sistema de ayuda online y páginas de ayuda para los módulos del 'núcleo', tanto para usuarios como para administradores.

Búsqueda: Todo el contenido en Drupal es totalmente indexado en tiempo real y se puede consultar en cualquier momento.

Código abierto: El código fuente de Drupal está libremente disponible bajo los términos de la licencia GNU/GPL. Al contrario que otros sistemas de 'blogs' o de gestión de contenido propietarios, es posible extender o adaptar Drupal según las necesidades.
Módulos: La comunidad de Drupal ha contribuido muchos módulos que proporcionan funcionalidades como 'página de categorías', autenticación mediante jabber, mensajes privados, bookmarks, etc.
Personalización: Un robusto entorno de personalización está implementado en el núcleo de Drupal. Tanto el contenido como la presentación pueden ser individualizados de acuerdo las preferencias definidas por el usuario.
URLs amigables: Drupal usa el mod_rewrite de Apache para crear URLs que son manejables por los usuarios y los motores de búsqueda.

1.3.2.2 Gestión de usuarios

Autenticación de usuarios: Los usuarios se pueden registrar e iniciar sesión de forma local o utilizando un sistema de autenticación externo como Jabber, Blogger, LiveJournal u otro sitio Drupal. Para su uso en una intranet, Drupal se puede integrar con un servidor LDAP.
Permisos basados en roles: Los administradores de Drupal no tienen que establecer permisos para cada usuario. En lugar de eso, pueden asignar permisos a un 'rol' y agrupar los usuarios por roles.

1.3.2.3 Gestión de contenido

Control de versiones: El sistema de control de versiones de Drupal permite seguir y auditar totalmente las sucesivas actualizaciones del contenido: qué se ha cambiado, la hora y la fecha, quién lo ha cambiado, y más. También permite mantener comentarios sobre los sucesivos cambios o deshacer los cambios recuperando una versión anterior.
Enlaces permanentes (Permalinks): Todo el contenido creado en Drupal tiene un enlace permanente asociado a él para que pueda ser enlazado externamente sin temor de que el enlace falle en el futuro.
Objetos de Contenido (Nodos): El contenido creado en Drupal es, funcionalmente, un objeto (Nodo). Esto permite un tratamiento uniforme de la información, como una misma cola de moderación para envíos de diferentes tipos, promocionar cualquiera de estos objetos a la página principal o permitir comentarios -o no- sobre cada objeto.

Plantillas (Templates): El sistema de temas de Drupal separa el contenido de la presentación permitiendo controlar o cambiar fácilmente el aspecto del sitio Web. Se pueden crear plantillas con HTML y/o con PHP.
Sindicación del contenido: Drupal exporta el contenido en formato RDF/RSS para ser utilizado por otros sitios Web. Esto permite que cualquiera con un 'Agregador de Noticias', tal como NetNewsWire o Radio UserLand visualice el contenido publicado en la Web desde el escritorio.

1.3.2.4 Blogging

Agregador de noticias: Drupal incluye un potente Agregador de Noticias para leer y publicar enlaces a noticias de otros sitios Web. Incorpora un sistema de caché en la base de datos, con temporización configurable.
Soporte de Blogger API: La API de Blogger permite que un sitio Drupal sea actualizado utilizando diversas herramientas, que pueden ser 'herramientas Web' o 'herramientas de escritorio' que proporcionen un entorno de edición más manejable.

1.3.2.5 Plataforma

Independencia de la base de datos: Aunque la mayor parte de las instalaciones de Drupal utilizan MySQL, existen otras opciones. Drupal incorpora una 'capa de abstracción de base de datos' que actualmente está implementada y mantenida para MySQL y PostgresSQL, aunque permite incorporar fácilmente soporte para otras bases de datos.
Multiplataforma: Drupal ha sido diseñado desde el principio para ser multi-plataforma. Puede funcionar con Apache o Microsoft IIS como servidor Web y en sistemas como Linux, BSD, Solaris, Windows y Mac OS X. Por otro lado, al estar implementado en PHP, es totalmente portable.
Múltiples idiomas y Localización: Drupal está pensado para una audiencia internacional y proporciona opciones para crear un portal multilingüe. Todo el texto puede ser fácilmente traducido utilizando una interfaz Web, importando traducciones existentes o integrando otras herramientas de traducción como GNU ettext.

1.3.2.6 Administración y Análisis

Administración vía Web: La administración y configuración del sistema se puede realizar enteramente con un navegador y no precisa de ningún software adicional.

Análisis, Seguimiento y Estadísticas: Drupal puede mostrar en las páginas Web de administración informes sobre referrals (enlaces entrantes), popularidad del contenido, o de cómo los usuarios navegan por el sitio.

Registros e Informes: Toda la actividad y los sucesos del sistema son capturados en un 'registro de eventos', que puede ser visualizado por un administrador.

1.3.2.7 Características de comunidad

Comentarios enlazados: Drupal proporciona un potente modelo de comentarios enlazados que posibilita seguir y participar fácilmente en la discusión sobre el comentario publicado. Los comentarios son jerárquicos, como en un grupo de noticias o un foro.

Encuestas: Drupal incluye un módulo que permite a los administradores y/o usuarios crear encuestas on-line totalmente configurables.

Foros de discusión: Drupal incorpora foros de discusión para crear sitios comunitarios vivos y dinámicos.

Libro Colaborativo: Esta característica es única de Drupal y permite crear un proyecto o "libro" a ser escrito y que otros usuarios contribuyan contenido. El contenido se organiza en páginas cómodamente navegables.

1.3.2.8 Rendimiento y escalabilidad

Control de congestión: Drupal incorpora un mecanismo de control de congestión que permite habilitar y deshabilitar determinados módulos o bloques dependiendo de la carga del servidor. Este mecanismo es totalmente configurable y ajustable.

Sistema de Caché: El mecanismo de caché elimina consultas a la base de datos incrementando el rendimiento y reduciendo la carga del servidor. (REYERO 2005)

1.3.2.9 Un blog para cada usuario Drupal

El módulo de blogs de Drupal les permite a todos los usuarios registrados mantener un weblog personal en el sitio. Los blogs son sitios Web fácilmente –y con frecuencia- actualizados con información escrita en un estilo informal y conversacional. Se organizan de forma inversa a la cronológica (es decir, la publicación más reciente va al principio) y archiva las publicaciones anteriores. Cada entrada individual tiene una URL permanente –o en otras palabras, estable- vinculando directamente a ese elemento. Los blogs usualmente tienen comentarios para cada entrada de tal manera que los usuarios puedan participar

en la discusión, y por lo general tienen fuentes RSS para ser publicadas en cualquier otro sitio o en un agregador de escritorio. Cada entrada usualmente contiene una idea, con un vínculo a la fuente principal sobre lo que se está discutiendo. Los blogs pueden ser escritos sobre cualquier tema, desde asuntos de su vida diaria hasta tecnología, política, deportes, productos de una compañía, entre otros.

Para tener un punto de referencia más práctico, los blogs pueden ser vistos como publicaciones del conocimiento personal, un lugar para que investigadores y entusiastas construyan y compartan información acerca de sus intereses. En sitios orientados a proyectos, es un espacio de trabajo para los miembros de un proyecto para publicar ideas que serán comentadas por otras personas del grupo. (JAIRO.SERRANO 2005a)

Cuando se habilita, el módulo comentarios de Drupal crea un grupo de discusión para cada nodo. Los usuarios pueden publicar sus comentarios para discutir sobre un tema del foro, publicaciones de diarios, artículos, páginas de libro, etc. Un administrador puede dar permisos de comentar a un grupo de usuarios, y los usuarios pueden (opcionalmente) editar el último comentario, asumiendo que nadie más ha comentado desde su publicación. Control del usuario sobre el despliegue de comentarios.

Adjunto a cada grupo de comentarios (comment board) hay un panel de control para personalizar la forma en que se muestran los comentarios. Los usuarios pueden controlar el orden cronológico de los comentarios (primero o último al comienzo) y el número de comentarios que aparecerá por cada página. Otras configuraciones adicionales son:

Lista hebrada: Despliega los comentarios agrupados de acuerdo a conversaciones y subconversaciones.

Lista plana: Despliega los comentarios en orden cronológico sin hilos.

Expandida: Despliega el título y el texto para cada comentario.

Colapsada: Despliega solo el título para cada comentario.

Cuando un usuario selecciona Guardar Configuración, los comentarios se muestran con la nueva configuración del usuario. Los administradores pueden establecer opciones por defecto para el panel de control, junto con los otros valores por defecto, en administrar » comentarios » configuración. (JAIRO.SERRANO 2005b)

Los comentarios se comportan como cualquier otro formulario de usuario en Drupal. Los filtros, iconos y código HTML que funcionan en otros nodos también funcionarán con los comentarios. Los administradores pueden controlar el acceso a varias funciones del módulo de comentarios a través de administrar » usuarios » configurar » permisos. En una nueva instalación de Drupal, todos los permisos de

los comentarios están deshabilitados por defecto. La selección de los permisos que se conceden a los roles (grupos de usuarios) los define el administrador del sitio. Estos son los permisos:

Acceder a los comentarios: Permite al usuario ver comentarios.

Administrar comentarios: Permite al usuario controlar la configuración, edición y eliminación de todos los comentarios.

Moderar comentarios: Permite al usuario organizar la publicación de comentarios (vea más acerca de moderación).

Publicar comentarios: Permite al usuario publicar comentarios en la cola de moderación del administrador.

Publicar comentarios sin aprobación: Permite al usuario publicar comentarios directamente sin pasar por la cola de moderación. (JAIRO.SERRANO 2005c)

1.4 JavaScript

JavaScript es un lenguaje de programación que ha permitido el gran desarrollo de la Web, ha sido el avance más significativo en el logro de páginas Web dinámicas y exactas en cuanto a posición y presentación de su contenido, es un lenguaje robusto y a la vez ligero, el cual a pesar de ser considerado por muchos como un lenguaje no orientado a objetos permite implementar varias de las características de este paradigma de programación, basado en prototipos, las nuevas clases se generan clonando las clases bases y extendiendo su funcionalidad, cualquier navegador puede interpretar el código JavaScript dentro de las páginas Web, permite la máxima interactividad entre el usuario y la página, además la verificación de los datos introducidos por el usuario antes de enviar el formulario al servidor, el manejo de applets y plugins dentro de múltiples marcos de HTML.

Este lenguaje puede desarrollarse en el cliente, se aplica en el navegador del cliente (JavaScript en el cliente) y además en el servidor (JavaScript en el servidor) el que nos ocupa en este trabajo es JavaScript en el cliente.

Este lenguaje de programación tiene algunas limitaciones:

- JavaScript por definición no es un lenguaje orientado a objetos y debido a la potencia de esta programación se ha conseguido un funcionamiento similar, intuitivo y potente.
- No se precompila.
- No es obligatorio declarar las variables.

- Verifica las referencias en tiempo de ejecución.
- No tiene protección del código, ya que se descarga en texto claro.
- El JavaScript es un lenguaje interpretado, o sea, que el sistema lo lee y traduce al mismo tiempo que lo va ejecutando, no confundir con el Java.

Con JavaScript se pueden lograr efectos realmente mágicos en una página, así como formularios, imágenes, textos, etc., tiene la ventaja de que la mayoría de los navegadores no tienen problemas para interpretarlo.

JavaScript es una forma de trasladar algunas tareas simples al lado del cliente.

Cuenta con un conjunto de nodos, que no es mas que todo lo que esta incluido dentro de una etiqueta de inicio y otra de final. Cada etiqueta se considera un nodo en si misma. Se distinguen los nodos de texto y los nodos de elementos. Estos nodos tienen arquitectura de objeto por tanto tienen un objeto prototipo. Cuando se tiene un nodo de tipo concreto se sabe que todos los nodos instancias heredan sus propiedades y métodos. No son objetos de JavaScript sino de HTML, pero pueden ser programados y manipulados desde JavaScript que es su ventaja.

En JavaScript se refiere siempre al objeto prototipo para crear las instancias, pero esto no hace al lenguaje ni más ni menos orientado a objetos que otros lenguajes. El concepto de clases en JavaScript estas dado por el prototipo, con la propiedad prototype de los objetos prototipos podemos añadir todas las propiedades que se quieran, no a la instancia sino al mismo prototipo. Permite codificar de una manera flexible, definir un objeto prototipo y si se necesita añadir alguna propiedad(es) lo único que se tiene que hacer es asignar el true. Se puede añadir además propiedades a los objetos prototipo predefinidos por el lenguaje.

JavaScript tiene una sintaxis que es en general parecida a la de C, excepto que no hay una función main sino que lo que no está dentro de una función se ejecutará mientras se cargue la página. Todos los métodos que JavaScript trae definidos son funciones desde el momento que se ejecutan. Se pueden crear funciones que ejecuten comandos que se le añadan. A partir de la versión 1.3 se puede anidar funciones que amplían notablemente las posibilidades y flexibilidad de la programación. También es necesario decir que no se pueden utilizar eficientemente el constructor o las funciones literales en los bucles (Función literal: Una función que se crea para ejecutar un código simple y por una sola vez). Tampoco es conveniente utilizar funciones anidadas en bucles. En JavaScript las funciones pueden manipularse dentro del mismo lenguaje, esto es una gran ventaja, ya que además de ser creada puede utilizarse como

dato, Por tanto puede asignarse a una variable, guardarse en los elementos de una matriz o en las propiedades de un objeto. Las funciones tienen argumentos que son tratados como objetos y forman una matriz. Esta matriz siempre tiene el nombre de argument(). El hecho que los argumentos formen una matriz permite que se puedan manipular, es decir, aumentar o reducir su número. Además se puede conocer el número de argumentos mediante la propiedad leght de las matrices.

JavaScript recibe información a través de eventos y propiedades de objetos, y la entrega mediante propiedades de objetos y métodos.

1.5 Definiendo AJAX

AJAX, en resumen, es el acrónimo para Asynchronous JavaScript + XML, no es una tecnología. Es realmente muchas tecnologías, cada una floreciendo por su propio mérito, uniéndose en poderosas nuevas formas.

1.5.1 Características de AJAX

AJAX incorpora:

- Presentación basada en estándares usando XHTML y CSS.
- Exhibición e interacción dinámicas usando el Document Object Model.
- Intercambio y manipulación de datos usando XML and XSLT.
- Recuperación de datos asincrónica usando XMLHTTPRequest.
- JavaScript poniendo todo junto.

Una aplicación AJAX elimina la naturaleza “arrancar-frenar- arrancar-frenar” de la interacción en la Web introduciendo un intermediario -un motor AJAX- entre el usuario y el servidor. Parecería que sumar una capa a la aplicación la haría menos reactiva, pero la verdad es lo contrario.

En vez de cargar un página Web, al inicio de la sesión, el navegador carga al motor AJAX (escrito en JavaScript y usualmente “sacado” en un frame oculto). Este motor es el responsable por renderizar la interfaz que el usuario ve y por comunicarse con el servidor en nombre del usuario.

El motor AJAX permite que la interacción del usuario con la aplicación suceda asincrónicamente (independientemente de la comunicación con el servidor). Así el usuario nunca estará mirando una ventana en blanco del navegador y un icono de reloj de arena esperando a que el servidor haga algo.

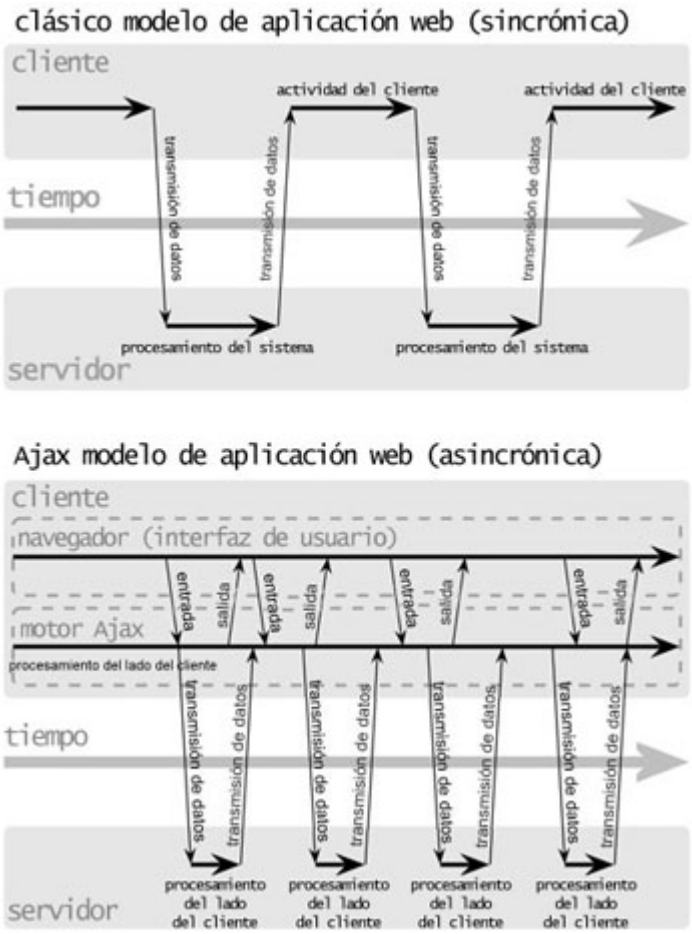

Figura 1. El patrón de interacción sincrónica de una aplicación Web tradicional (arriba) comparada con el patrón asincrónico de una aplicación AJAX (abajo).

Cada acción de un usuario que normalmente generaría un requerimiento HTTP toma la forma de un llamado JavaScript al motor AJAX en vez de ese requerimiento. Cualquier respuesta a una acción del usuario que no requiera un viaje de vuelta al servidor (como una simple validación de datos, edición de datos en memoria, incluso algo de navegación) es manejada por su cuenta.

Si el motor necesita algo del servidor para responder (sea enviando datos para procesar, cargar código adicional, o recuperando nuevos datos) hace esos pedidos asincrónicamente, usualmente usando XML, sin frenar la interacción del usuario con la aplicación.

El modelo Ajax provee un nivel intermedio llamado motor (engine) que no es más que un objeto o función de JavaScript, que será llamado cada vez que se necesite una respuesta del servidor. Lejos de lo que

hacen los modelos tradicionales que proveen un vínculo a otro recurso (como una página Web), cada vínculo hará una llamada al **ajax engine** quien ejecutará la petición. Cada petición se hará de manera asincrónica lo que significa que no tendrá que esperar que se haya completado para continuar. El servidor esta configurado para que lo que devuelve a esta petición pueda ser utilizado por **ajax engine** ,puede ser texto, XML o cualquier otro formato de dato que se pueda necesitar, el único requerimiento es que se pueda ser interpretado por **ajax engine** .Cuando el **ajax engine** recibe la respuesta se ejecuta el cambio en la interfaz del usuario basado en la información que se ha recibido, dado que este proceso involucra la transmisión de menos información que la forma tradicional del modelo de aplicación web, la actualización de la interfaz de usuario es mas rápida y el mismo puede realizar el trabajo mucho mas rápido.

1.5.2 Uso de Ajax

Dentro de los usos que no se pueden dejar de mencionar están:

- Validación de datos de formularios en tiempo real.
- Identificadores de usuario, nº de serie, códigos postales u otro código especial que necesite validación en el lado del servidor antes de ser enviado el formulario.
- Autocompletado.
- Direcciones de correo, nombres, ciudades.
- Operaciones de detalle.
- Obtener información más detallada de un producto.
- GUI avanzadas.
- Controles en árbol, menús, barras de progreso.
- Refrescador de datos.
- Notificaciones del servidor.
- Actualizar o eliminar registros.
- Expandir formularios web.
- Devolver peticiones simples de búsqueda.
- Editar árboles de categorías.

1.5.3 Ventajas de Ajax

Como principales ventajas de Ajax es importante que se mencione:

- Tráfico mínimo: Las aplicaciones Ajax deben enviar y recibir una pequeña cantidad de información desde y para el servidor, por lo que se minimiza el tráfico entre el cliente y el servidor, asegurándose que no reciba o envíe información innecesaria.
- Sin sorpresas: Introduce diferentes modelos de interacción con el usuario en comparación con el modelo tradicional.
- Convenciones establecidas: No gasta tiempo inventando nuevos modelos de interacción de usuario con los que no estarán además familiarizados.
- El usuario primero: diseñar las aplicaciones con el usuario en mente antes que nada.

1.5.4 Aplicaciones que utilizan Ajax

Diferentes proyectos demuestran que Ajax no es solo técnicamente importante, sino también prácticos para aplicaciones en el mundo real. Esta no es otra tecnología que solo trabaja en un laboratorio. Las aplicaciones Ajax pueden ser de cualquier tamaño, de lo más simple, funciones simples como Google Suggest a las muy complejas y sofisticadas como Google Maps. Otros ejemplos de aplicaciones que utilizan Ajax son Gmail, A9, Yahoo! News, and Bitflux Blog.

1.6 JSON

JSON, acrónimo de "**JavaScript Object Notation**", es un formato ligero para el intercambio de datos. Es un subconjunto de la notación literal de objetos de JavaScript pero no requiere el uso de XML. Se utiliza principalmente para transferir datos estructurados a través de una conexión de red, utilizando un proceso llamado serialización.

Es un formato de texto que es completamente independiente del lenguaje pero utiliza convenciones que son ampliamente conocidos por los programadores de la familia de lenguajes C, incluyendo C, C++, C#, Java, JavaScript, Perl, Python, y muchos otros. Estas propiedades hacen que JSON sea un lenguaje ideal para el intercambio de datos.

JSON está constituido por dos estructuras:

Una colección de pares de nombre/valor. En varios lenguajes, esto es conocido como un objeto, registro, estructura, diccionario, tabla hash, lista de claves o un arreglo asociativo.

Una lista ordenada de valores. En la mayoría de los lenguajes, esto se implementa como arreglos, vectores, listas o secuencias.

Estas son estructuras universales; virtualmente todos los lenguajes de programación las soportan de una forma u otra. Es razonable que un formato de intercambio de datos que es independiente del lenguaje de programación se base en estas estructuras.

1.6.1 Utilidades de JSON

Al hablar de JSON se tienen que listar sus utilidades:

- JSON es fácil.
- Si estás familiarizado con la escritura de clases en PHP o C++, entonces te vas a sentir muy a gusto escribiendo JavaScript en notación objeto.
- JSON no es más que escribir pares del tipo "nombre: valor" y asignarlos a un objeto.
- JSON es fácil de entender, porque si está bien escrito forma una estructura que se auto explica.
- JSON es muy rápido.
- JSON organiza el problema de la programación por procedimientos (PP). Puesto que tiene más de una función init.
- Al escribir en JSON lo más probable es que no ocasione conflictos con ningún otro script que sea llamado en la misma página.

1.6.2 ¿Qué es la notación objeto en JavaScript?

Como un simple ejemplo, la notación objeto puede ser expresada con el siguiente formato:

```
var obj =
{       // string
        a: 'valorEjemplo',
        // array
        b: [obj1, obj2, 'tres', 4, obj5],
```

```
    // funcion
    c: funcion () {var bum = this.a;}
}
```

En el ejemplo de arriba, `a`, `b` y `c` son **nombres**, y cada uno de ellos tiene su valor expresado inmediatamente después de dos puntos y seguido de una coma final (excepto el último nombre). Cada **par** es considerado un **miembro**.

1.6.2.1 Nombres y Valores

Los nombres pueden ser cualquier cosa excepto las palabras reservadas de JavaScript. Al igual que las variables, no pueden empezar con números. Tampoco pueden incluir caracteres no alfanuméricos, excepto el guión bajo o el signo pesos y no puede estar duplicado en el mismo objeto.

Los valores son expresados como *string*, *número*, *objeto*, *array*, *booleano* o *null*. Hay más reglas de sintaxis para casos específicos que se pueden encontrar en el sitio de Douglas (complicado y en inglés).

1.6.3 Ventajas de JSON

Parte de la razón por la que JSON es fácil es porque es *fácil de leer* tanto para humanos como para máquinas (piensen en el parsing). No existe más el problema de escribir función tras función que no tengan correlación entre sí o pertenezcan a grupos totalmente distintos.

"Programar con objetos permite al desarrollador separar funciones en subconjuntos que facilitarán la organización del código y evitará que los scripts se molesten unos a otros."

Y aunque mucha de la difusión que está teniendo JSON ahora mismo es porque funciona como un formato liviano de intercambio de datos que múltiples lenguajes pueden entender de manera fácil, simplemente tiene sentido usarlo en un nivel estructural, aun si no estás intercambiando ningún tipo de información.

La simplicidad de JSON ha dado lugar a la generalización de su uso, especialmente como alternativa a XML en AJAX. Una de las supuestas ventajas de **JSON** sobre XML como formato de intercambio de datos en este contexto es que es mucho más sencillo escribir un analizador semántico de JSON, este se emplea habitualmente en entornos donde el tamaño del flujo de datos entre cliente y servidor es de vital importancia (de aquí su uso por Yahoo, Google, etc., que atienden a millones de usuarios).

Cada vez hay más soporte de JSON mediante el uso de paquetes escritos por terceras partes. La lista de lenguajes soportados incluye ActionScript, C, C#, ColdFusion, Common Lisp, E, Java, JavaScript, ML, Objective CAML, Perl, PHP, Python, Rebol, Ruby, y Lua.
En Diciembre de 2005 Yahoo! comenzó a dar soporte opcional de JSON en algunos de sus servicios web.
El beneficio de JSON, entonces, no es que sea más pequeño a la hora de transmitir, sino que representa mejor la estructura de los datos y requiere menos codificación y procesamiento.

1.7 Librerías

En todos los lenguajes de programación existen librerías de funciones que sirven para hacer cosas diversas y muy repetitivas a la hora de programar. Las librerías de los lenguajes de programación ahorran la tarea de escribir las funciones comunes que por lo general pueden necesitar los programadores. Un lenguaje de programación bien desarrollado tendrá una buena cantidad de ellas. En ocasiones es más complicado conocer bien todas las librerías que aprender a programar en el lenguaje.
JavaScript contiene una buena cantidad de funciones en sus librerías. Como se trata de un lenguaje que trabaja con objetos muchas de las librerías se implementan a través de objetos.
Uno de los inconvenientes más comunes a la hora de diseñar una interfaz de aplicación Web es que una vez que la página se ha descargado en el cliente, la conexión con el servidor se corta. Cualquier intento de dinamismo en la interfaz por parte del cliente requiere una comunicación con el servidor para la recarga (proceso que tiende a convertir la aplicación poco elegante y lenta).
En el modelo tradicional de aplicaciones Web el usuario envía una petición al servidor requiriendo una página, la cual es construida y enviada al navegador. Esta página incluye un formulario HTML para capturar datos del usuario. Una vez que el usuario reenvía los datos al servidor, la siguiente página será generada y enviada dependiendo del valor de dichos datos, y así el proceso continúa. Supongamos una aplicación de escritorio para el registro de un número de serie. Según convenga se lo puede plantear de diversas formas, una vez hemos terminado de rellenar los correspondientes cuadro de textos con los caracteres del código, podríamos hacer aparecer un “Tilde” verde a la derecha indicando que hemos introducido un código válido. Tan pronto como se introduce el código, la aplicación puede comprobar su validez y responder.
En contraste con el ejemplo anterior pero esta vez orientado en una interfaz Web. Por supuesto, todos los cuadros de textos donde introducir el código serán idénticos, pero al rellenarlos, el usuario tendrá que

enviar esos datos al servidor para que éste valide el código. Una nueva página será entonces cargada informando del éxito o fracaso de la operación, y en caso de fallo, el usuario tendrá que volver atrás e intentarlo de nuevo cuantas veces sea necesario. Una solución a estos problemas se presenta con el objeto XMLHTTPRequest. Este objeto, ahora disponible como objeto nativo tanto en Mozilla como también en otros navegadores existentes, permite a JavaScript realizar peticiones al servidor remoto sin la necesidad de recargar la página. En esencia, pueden realizarse peticiones y recibir respuestas HTTP completamente en segundo plano y sin que el usuario experimente ninguna interrupción visual.

Hasta el momento se tienen los argumentos que sirven de introducción para comenzar a nombrar librerías del lenguaje de programación JavaScript, como punto de partida se tiene AJAX que no es ninguna tecnología, ni lenguaje de programación, es una técnica de desarrollo Web que combina varias tecnologías consiguiendo una navegación más ágil y rápida, más dinámica, que utiliza JavaScript como nexo de unión, dentro de AJAX tenemos la siguientes librería:

1.7.1 Librería Xajax

Xajax es una biblioteca de código abierto para PHP que permite crear de manera fácil y simple aplicaciones Web basadas en AJAX usando además HTML, CSS, y JavaScript. Las aplicaciones desarrolladas con Xajax pueden comunicarse asincrónicamente con funciones que se encuentran del lado del servidor y así actualizar el contenido de una página sin tener que recargarla nuevamente.

Xajax es compatible con Firefox, Mozilla, Internet Explorer y Safari, puede ser usado para actualizar Styles, CSS Classes, botones de selección, checkbox y botones de radios o cualquier otro atributo de un elemento. Cada función registrada para ser accesible a través de Xajax puede tener distintos tipos de petición. (WIKIMEDIA FOUNDATION 2007f)

1.7.2 Librería DOJO Toolkit

La librería DOJO Toolkit es una biblioteca JavaScript de código abierto, que proporciona un API para el control y manipulación de historial, permite la manipulación de URL y marcadores/favoritos, el trabajo con Widgets e incluye el ordenamiento de tablas, validación de formularios, menús y barras de menús, Google y Yahoo! Maps. (MOLINA 2006)

1.7.3 Librería XOAD

Es una biblioteca orientada a objetos basada en PHP, conocida anteriormente como NAJAX la misma permite la documentación de las clases y tutoriales sencillos. Emplea JSON y objetos PHP para la comunicación. Además soporta eventos del lado del cliente y del servidor. (MOLINA 2006)

1.7.4 Librería Jmol.js

Jmol.js es una biblioteca de JavaScript que puede usarse para simplificar el desarrollo de páginas Web que incluyan la mini aplicación JmolApplet.

Con esta librería se automatiza la generación de muchas etiquetas HTML, haciendo así el código más simple, más fácil de entender y menos propenso a errores.

La librería Jmol.js debería funcionar correctamente con la mayoría de los navegadores Web recientes (de este siglo). Además, en los sistemas Windows (Win32) generalmente funciona bien con la serie de navegadores Netscape 4.* (que pertenece al siglo pasado).

Hay algunas situaciones en las que se podría optar por no utilizar la biblioteca Jmol.js. Entre ellas, se puede querer construir una aplicación avanzada en el servidor Web, que genere HTML desde el servidor, tal como PHP, JSP, ASP, CGI, etc. Si éste es el caso, es lo adecuado que se genere el código HTML directamente en el servidor, en cuyo caso no se usará la biblioteca Jmol.js.

1.7.5 Librería Ajax Auto Suggest

Esta librería permite utilizar el teclado para seleccionar un valor. Los datos para esta lista se llaman desde un archivo XML que puede ser generado por PHP o cualquier otro lenguaje. (UNIJIMPE 2007)

1.7.6 Librería Interfaz de Usuario de Yahoo (YUI)

Esta librería cuenta con una serie de características, utilidades y controles desarrollados en JavaScript, para la construcción de aplicaciones Web interactivas y opulentas, utilizando técnicas como DOM scripting, DHTML y AJAX, también incluye diversos recursos basados en CSS. Todos los componentes de YUI son libres para el uso de todos los que accedan a ellos, esta librería esta patentada con la licencia BSD de software libre. (FIREBOY 2007)

1.7.7 Librería Ext

Ext es neutral al lenguaje que se use en el servidor. Siempre que el resultado se envíe a la página en el formato adecuado, Ext no se preocupará de lo que pase en el servidor. Hay docenas de widgets a escoger en Ext, incluyendo composiciones automáticas de páginas, pestañas, menús, barras de herramientas, diálogos, vistas en árbol. Proporciona un selector de nodos DOM extremadamente poderoso llamado DomQuery (DomQuery puede usarse como una librería independiente, pero en el contexto de Ext se usará para seleccionar elementos para poder interactuar con ellos a través de la interfaz Element. Elements contiene muchos de los métodos y propiedades de DOM que se necesitará, proporcionando una interfaz conveniente, unificada y multinavegador).

1.7.8 Librería JQuery

JQuery es un nuevo tipo de librería de JavaScript, rápida y concisa que simplifica el trabajo con documentos HTML. No es un extenso e inflado framework que prometa lo mejor en AJAX, ni un conjunto de innecesarias y complicadas mejoras: JQuery ha sido diseñado para cambiar tu forma de escribir JavaScript. (TEAM 2007)

Después de haber analizado las herramientas y librerías expuestas anteriormente se puede llegar a la conclusión de que a nivel internacional existen diferentes vertientes y criterios que apuestan por una en específico, dado el nivel de seguridad y usabilidad de la misma, en correspondencia además con el problema al que le da solución, aunque presentan características similares están en una constante competencia por lograr la satisfacción del cliente, en general debido a su nivel de complejidad y a la utilidad que pueda aportar en todo momento, uno de los aspectos fundamentales en este medio de las TICS.

1.8 Descripción de las librerías

1.8.1 Xajax

Las aplicaciones Web con Ajax funcionan porque hay un código JavaScript ejecutándose en el cliente que hace peticiones al servidor, Xajax nos permite automatizar la tarea de creación de estas funciones en JavaScript, indicando la información necesaria en el código PHP que se ejecutará inicialmente. Con Xajax generamos el JavaScript que nos permitirá usar Ajax en nuestra aplicación Web. O lo que es lo mismo: nos permitirá generar el JavaScript que se encargará luego de pedir al propio PHP qué información

necesita. Xajax permite escribir funciones en PHP que pueden ser accedidas cuando ya la página ha estado enviada al navegador, cuando el usuario ha disparado un evento o la función PHP ha sido llamada desde JavaScript. Estas funciones PHP modifican el contenido o el aspecto de la página, como lo podría hacer JavaScript, pero se hacen desde PHP, lo que abre un abanico de posibilidades: PHP puede acceder a bases de datos, al sistema de archivos sin recargar la página.

El sistema único de respuesta de XML de Xajax del lenguaje JavaScript, maneja automáticamente los datos devueltos de sus funciones y pone al día su contenido o el estado según las instrucciones que devuelve de sus funciones de PHP. Dado que Xajax hace este trabajo no es necesario que se tenga que escribir las funciones de callback de JavaScript.

Xajax es orientado a objeto mantiene estrechas relaciones entre el código y los datos, guarda el código del Xajax a parte del otro código. Porque es orientado objeto, se puede agregar cualquier funcionalidad de encargo al Xajax extendiendo la clase del xajaxResponse y usando el método del addScript ().

Además de poner al día valores del elemento y el innerHTML, el Xajax se puede utilizar para actualizar estilos, clases del css, la selección del checkbox y del botón de radio, o casi cualquier otra cualidad del elemento.

Xajax soporta temporalmente arreglos, arreglos asociativos, simples y multidimensionales de JavaScript a PHP, como parámetros a sus funciones del Xajax. Si se pasa un objeto de JavaScript en una función del Xajax, la función de PHP recibirá un arreglo asociativo que representa las características del objeto.

Xajax proporciona el proceso asincrónico fácil del Form. Usando xajax.getFormValues () el método del JavaScript, se puede enviar un arreglo que representa los valores en una forma como parámetro a una función asincrónica del Xajax:

xajax_processForm (xajax.getFormValues ('formId ');

Incluso trabaja con nombres complejos de entrada como "checkbox [] [ ]" y "name [first]" para producir arreglos multidimensionales y asociativos, apenas como si hubiera enviado la forma y utilizado el arreglo del PHP $_get

Usando Xajax puede enviar dinámicamente JavaScript adicional a la aplicación para que corra en respuesta a un requerimiento tan bien como actualizar dinámicamente atributos del elemento.

Xajax compara automáticamente los datos devueltos de las funciones de PHP con los datos que están ya en el atributo del elemento que se ha marcado para el cambio. La cualidad se actualiza solamente con los nuevos datos si cambia realmente cuál está ya en el atributo.

Cada función registrada para ser accesible a través de Xajax tiene diversos tipos de peticiones.
Si no se especifica ninguna petición URI, Xajax intenta de autodetectar el URL del script. El algoritmo del autodetección de Xajax es bastante sofisticado, en la mayoría de los servidores, trabajará bajo protocolo seguro de https:// así como http:// y con los puertos desconocidos.
Xajax codifica todas sus peticiones y respuestas en UTF-8 de modo que pueda apoyar una gama más amplia de caracteres de idiomas, se han probado con éxito varios caracteres incluyendo español, ruso, árabe, y hebreo.
Casi todo el código JavaScript generado por Xajax es incluido en la aplicación Web a través de JavaScript externo dinámico. Cuando se vea la fuente de la aplicación en el browser, el margen no será alterado por definiciones de la función de JavaScript.
Xajax se puede utilizar con el sistema de plantillas del Smarty creando una variable en Smarty que contiene el JavaScript de Xajax:
$smarty->assign ('xajax_javascript', $xajax->getjavascript ());
Entonces se puede utilizar
{$xajax_javascript}
En el encabezamiento de la plantilla para utilizar Xajax en el sitio.

1.8.2 DOJO Toolkit

DOJO es un toolkit open source DHTML (Dynamic HTML) escrito en JavaScript, su idea es la de abstraer al desarrollador de las complejidades del DHTML y de las discrepancias existentes entre navegadores, que hacen que el código JavaScript a utilizar sea diferente. Algunas de las características que tiene son:

- Maneja incompatibilidades entre navegadores.
- Soporte AJAX.
- Oculta el manejo del XMLHTTPRequest.
- Soporte de backward, forward y soporte de bookmarks.
- Sistema de eventos orientado a aspectos.
- Los componentes de DOJO están modelizados con Widgets. Y a estos widgets les podemos asociar eventos.

Los eventos son controlados por la librería dojo.event. Para asociar un evento a un botón se debe utilizar el método dojo.event.conect (widget, evento, función). Este método recibe tres parámetros:

- **widget**: es el widget al que queremos asociar el evento.
- **evento**: evento que queremos controlar.
- **función:** función JavaScript asociada al código del evento. (LINEADECODIGO 2007)

1.8.3 Ajax Auto Suggest

Esta librería es muy sencilla de implementar, lo único que debemos hacer es crear el PHP que haga la consulta a la base de datos y genere el XML en el formato deseado, la otra parte será controlada por esta librería. Además un punto importante es que estéticamente esta librería esta muy bien lograda pues el suggest es muy intuitivo y con colores muy sobrios.

Entre las principales características de esta librería tenemos:

- Uso del teclado para seleccionar un dato.
- Mensaje opcional cuando no hay resultados.
- Auto-completar cuando se presiona la tecla ENTER.
- Soporte a funciones Callback.
- Diseño excelente.
- Funcionamiento.

Para implementar Ajax Auto Suggest, el primer paso es descargar la librería, luego de extraer los archivos, creas un archivo HTML donde se debe insertar un llamado a la librería.

<script type="text/javascript" src="bsn.AutoSuggest_c_2.0.js"**>**

</script>

Luego se crea un formulario en el cual se inserta un campo de texto que servirá para hacer el suggest que se desee.

<form method="get" action=""**>**

Persona**<br** /**>**

<input type="text" id="criterio" value="" **/>
**

<input type="submit" value="Enviar" **/>**

</form>

El siguiente paso es convertir este campo de texto normal en un campo de texto AutoSuggest, esto se hace insertando el siguiente código JavaScript antes del tag.

<script type="text/javascript"**>**

```
var options = {
    script:"test.php?",
    varname:"input"
};
var as_xml = new AutoSuggest('criterio', options);
```

</script>

Donde se tiene que: **script.php** es el script que devolverá el XML con los datos para el suggest, **varname** es el nombre de la variable que se le pasará por método GET al script antes definido. Por ejemplo si se escribe *Al* en el campo se pedirán los datos de: *test.php?input=Al*.

Para finalizar el XML que devuelve el PHP debe tener un determinado formato para poder ser interpretado con nuestro Ajax Suggest.

<results>

<rs id="1" info=""**>**Alejandro**</rs>**

<rs id="2" info=""**>**Alberto**</rs>**

<rs id="3" info=""**>**Alma**</rs>**

</results>

(UNIJIMPE 2007)

1.8.4 Librería Interfaz de Usuario de Yahoo (YUI)

Algunos controles de la YUI contienen sus correspondientes ficheros en CSS que determinan la apariencia de este componente, para ello es necesario incluir por tanto el fichero con el código CSS, esto permite además modificar el código CSS para adaptarlo a las especificidades de la página.
Los componentes de la YUI están dentro de tres grupos: Utilidades, Controles UI y Recursos CSS.

1.8.4.1 Utilidades

Animación: Permite la rápida personalización e implementación de las animaciones que incluye tamaño, claridad, color, posición, así como otras características visuales. La sintaxis utilizada para configurar las animaciones hace que la utilidad Animación de la YUI sea fácil de implementar incluso cuando se van a realizar animaciones complicadas.
Administrador de conexión: Esta utilidad permite instanciar objetos en las páginas HTTP a través de una interfaz simplificada, así como administrar el XMLHttpRequest (Es un API -conjunto de especificaciones de comunicación entre componentes software-, que puede invocarse desde JavaScript, que se usa para transferir y manipular datos XML hacia y desde el navegador Web, estableciéndose un canal de conexión independiente entre el lado del cliente de la página Web y el servidor. Los datos devueltos por la llamada a XMLHttpRequest serán, por lo general, obtenidos desde bases de datos en el servidor. La llamada puede devolver datos en XML o en cualquier otro formato textual.)
Datasource: Provee una interfaz para acceder a los datos de un arreglo, configurable además para interactuar con diferentes tipos de datos.
Evento: Este sofisticado administrador de clases brinda una manera fácil y segura de acceder a los eventos en el explorador, además incluye un objeto de evento de cliente, un mecanismo para la publicación y suscripción de los momentos más interesantes en el flujo de la aplicación.

Objeto global de Yahoo: Aquí es donde reside todo el código de la librería YUI, tiene que incluirse en todas las páginas que van a utilizar la YUI y tiene que aparecer antes que cualquier otro componente, también incluye un número de métodos que se utilizan en toda la librería.

1.8.4.2 Controles YUI

Los controles de la librería YUI proveen elementos de diseño altamente interactivos para las páginas Web. Estos elementos son creados y administrados completamente en el lado del cliente y no requiere que la página se actualice.

YUI Controls incluye:

AutoComplete: Permite una completa interacción del usuario que incluye la entrada de textos, el control provee una lista de peticiones y funcionalidades type-ahead en una variedad de formatos de datos soportados en el servidor con XMLHttpRequest.

Button Control (Beta): El Button Control provee checkbox, radio button, submit y un menu-button, elementos estos que tienen un impacto visual mayor y son más potentes desde el punto de vista de la programación que los buscadores construidos a partir de widgets.

Calendar: El Calendar Control es un control gráfico, dinámico utilizado para la selección de datos.

Container: La familia de contenedores de controles soporta una variedad de patrones DHTML que incluye Tooltip, Panel, Dialog, SimpleDialog. Provee además una plataforma de implementación adicional, administrada por los patrones DHTML.

Logger: El logger de YUI provee una manera rápida y sencilla de escribir los mensajes de los logs en consola, el FireBug extensión de Firefox o la consola Safari JavaScript. El debuggueador construido en los componentes de la YUI está integrado con un Logger que muestra mensajes para implementaciones del debuggueador.

Menu: Aplicar estilo de menú requiere solo unas pocas líneas de código con el Menu Control. El menú puede ser generado completamente en JavaScript o puede ser organizado en el principio de la semántica de listas.

Slider: Este control provee un elemento genérico slider que permite al usuario escoger sin un rango finito de valores en uno o dos opciones.

TabView: TabView permite transformar limpiamente la semántica subrayada en un rico control tabbed-navigation.

TreeView: El control TreeView produce un árbol de contenido cuyos nodos que pueden expandirse y contraerse por la interacción del usuario (y por script, en caso de que sea necesario). Estos nodos pueden contener vínculos o propiedades y pueden ser cargados dinámicamente. El despliegue de los elementos del nodo puede ser administrado con CSS para crear un archivo visible que haga una lista de tareas u otro tratamiento visual.

1.8.4.3 Recursos CSS de la YUI

CSS Grids: CSS Grids ofrece casi 200 capas predefinidas y un número ilimitado de características permutacionales que trabajan a través todo el buscador.

CSS Fonts: El paquete de fuentes provee familias de fuentes y tamaños de renderizado estándares para varios buscadores.

CSS Reset: Utiliza las declaraciones CSS para remover márgenes y estandarizar el renderizado del buscador en un conjunto de elementos, neutralizando la construcción de hojas de estilos en el buscador.

1.8.4.4 Patrones de diseño de YUI

Es una sofisticada y envolvente guía de diseño para la construcción de páginas Web y aplicaciones. Estos patrones de diseño no requieren los componentes de la YUI, aunque utilizando esta librería YUI es más fácil la implementación de los patrones.

Estos controles proveen un diseño visual de elementos altamente interactivo para la página Web, estos elementos son creados y administrados enteramente del lado del cliente y no requiere que la página sea actualizada.

A pesar de todas estas facilidades y funcionalidades YUI se vuelve extremadamente pesada en su uso pues la mayoría de sus clases se encuentran interrelacionadas y esto obliga a enlazar grandes cantidades de código de la librería para dar solución incluso a problemas muy sencillos.

1.8.5 JQuery

Esta librería utiliza un interesante concepto para hacer el código corto y simple, tiene manejadores de eventos, como los métodos bind y unbind sirven para asignar y desasignar eventos. Otro tema que JQuery resuelve con facilidad es el de los efectos, añade dinamismo visual a la presentación del sitio, como son añadirle funcionalidad, tanto al código como al resto de los elementos. Además se le añade

plugins que le da versatilidad. Tiene un núcleo muy pequeño de solo 15KB, además es muy estable. Está bajo la licencia GPL, de Software Libre.

JQuery por otro lado tiene cosas de serie, como es el poder acceder a elementos DOM con la sintaxis propia de CSS, ha puesto en su Web una lista de plugins y demos muy útiles.

Normalmente, cuando se trabaja con JavaScript el código no corre hasta tanto no se hayan cargado las imágenes, incluyendo los banners, para solucionar este problema JQuery ha implementado un procedimiento que se puede utilizar, conocido como ready event, este código chequea el documento y espera a que este listo para manipularlo. También permite dibujar las tablas tal cual si fueran cebras, añadir y quitar clases, asignar reglas de estilo y leer valores computados, otro método sumamente útil, de nombre Hover, es el que acepta como parámetros dos funciones, la primera de ellas se ejecutará cuando el mouse se posicione sobre el elemento, la segunda, cuando salga de él. Dichos métodos al igual que la mayoría de los métodos de JQuery son encadenables.

Otra de las características es que un usuario de JQuery puede definir su browser, en la medida que se avanza en el proyecto, formulando preguntas, y el funcionamiento de la lógica de ellas contra su código fuente.

Alternativamente, el usuario puede elegir de una variedad de browsers desarrollados de antemano, y lo utiliza tal y como está o lo modifica para satisfacer sus necesidades específicas. El usuario puede especificar cómo los resultados de la pregunta se organizan usando un interfaz simple del redactor de las variables.

El lenguaje de interrogación de JQuery: Es un lenguaje de interrogación de la lógica (Prolog) basado en TyRuBa (El motor de inferencia de la lógica sobre el cual se basa JQuery). TyRuBa es un lenguaje de programación de la lógica puesto en ejecución en JavaScript.

Ve los sistemas de nodos del HTML como objetos para pasar mensajes. Más en vez de tener una clase separada que agregue el texto después de un nodo del HTML, el JQuery agrega la funcionalidad sobre el objeto del JQuery, que es devuelto por la función $(que se utiliza para seleccionar elementos). JQuery pasa mensajes a los objetos, puede poner funcionalidad adicional del selector en ejecución en el método $, usa los selectores del CSS y los operadores de XPath.

Los métodos de JQuery se requieren para colocar automáticamente todos los elementos de DOM en el código, y aplican el método deseado. Se elimina la iteración en el código (en la mayoría de los casos), esta es una de las ventajas prácticas, cotidianas y superiores de usar JQuery. Dada la madures que ha

adquirido esta librería, es más fácil construir plugins a partir de una estructura ya existente, permitiendo así que se elimine prácticamente toda la iteración molesta.

Las facilidades de codificación con esta librería, su sencillez, su eficiencia, las facilidades que brinda para la integración con otras librerías, unido a las opiniones de la comunidad de Drupal sobre que librería debe usarse nos llevo a seleccionarla como el mejor de los candidatos para la integración con el CMS Drupal.

En este capítulo se han abordado la mayoría de las librerías conocidas y más utilizadas, que de una manera u otra utilizan JavaScript para realizar sus funcionalidades más rápidamente y con mayor eficacia. Se ha intentado darle a conocer al lector las características principales y fundamentales de cada una de ellas, para que de un cierto modo pueda este seleccionar cual de ellas se adecua más a sus necesidades y conozca sus ventajas y desventajas con el fin de darle una utilización más eficiente y productiva.

También se hizo una breve descripción de dos herramientas relevantes (Quanta y Drupal) con el objetivo de hacerle ver al lector como funcionan internamente, aunque se debe aclarar que para la elaboración de este trabajo lo más importante es enfocarse en las características y ventajas que ofrece Drupal, por ser uno de los CMS más dinámicos, fácil de utilizar y administrar a la hora de administrar un Portal o Sitio Web.

En Cuba se utilizan muchas de estas librería y herramientas en la creación de portales Web, de carácter informativo, educativo y empresarial, permitiendo de forma bastante acelerada que todas las esferas del gobierno estén a la altura del desarrollo del mundo así como la incursión de la nación en el ciberespacio.

La Universidad de las Ciencias Informáticas, pionera de la producción de software, se abre camino en la Web, recurriendo a gestores de contenido que posibilitan no solo la exportación e importación de los productos sino que hacen más llevadera la vida universitaria, motivando tanto el estudio como los momentos de esparcimiento, mejorando así la calidad de vida en la comunidad universitaria lo que se refleja directamente en los resultados que obtiene el país y la institución.

2 CAPÍTULO 2: ARGUMENTACIÓN DE LA PROPUESTA

2.1 Introducción

Como se explica en el sitio Web jQuery.com esta librería no es un extenso e inflado framework que prometa lo mejor en AJAX, ni un conjunto de innecesarias y complicadas mejoras: JQuery ha sido diseñado para cambiar la forma de escribir JavaScript.

La función principal de JQuery está designada por el indicador $ y través de la misma podemos acceder a casi todas las funcionalidades de la librería.

Veamos un ejemplo sencillo antes de entrar a describir la JQuery en profundidad:

$('tbody tr: even').addClass ('alt');

Dicho código selecciona todos los tbody de las tablas que tengamos en nuestra página y añade a las filas impares (even) la clase alt. Si esta clase respondiera a un código CSS similar a tr.alt {background: #ddd;}, entonces las filas pares tendrían un color diferente que permitiría destacarlas y así facilitar la lectura de las tablas.

En resumen: JQuery sirve para seleccionar elementos del DOM y manipularlos a nuestro antojo.

2.2 ¿Cómo obtenerla?

Obtener la librería es muy sencillo, el núcleo se encuentra disponible en Internet a través del sitio jQuery.com en la sección de descargas, es muy ligero, apenas unos 25 Kb en su última versión (1.1.2).

Después se pueden ir añadiendo las extensiones que se deseen en dependencia de las necesidades como se podrá ver más adelante.

La idea de que las extensiones sean independientes entre sí facilita su ligereza, esto le brinda a JQuery una ventaja sobre otras librerías potentes como YUI. JQuery es desarrollada por John Resig bajo las licencias MIT y GPL.

2.3 Lo básico de JQuery

Para comenzar a usar las funcionalidades del núcleo de JQuery la acción a realizar es tan sencilla como añadir una etiqueta script en el encabezado (HEAD) de la página donde se especifique el camino de nuestra librería:

<script type="text/javascript" src="camino/a/jquery.js"> </script>

Y a partir de ese momento dentro de las etiquetas scripts se tendrán disponibles las utilidades de dicha librería.

Hasta el momento si se quería ejecutar una función cuando se completara la descarga de la página se hacía de la siguiente forma:

```
window.onload = function () {...}
```

Pero esto trae el problema de que el “script” no se ejecuta hasta que las imágenes y animaciones que pueda haber en nuestra página no concluyen su descarga, en ocasiones ese es el objetivo, pero la mayoría de las veces solo se está esperando la descarga del código HTML de la página para que los “scripts” entren a funcionar, para esto JQuery nos brinda la siguiente solución.

```
$(document).ready (function () {
   // Nuestro código va aquí
 });
```

En este caso nuestro código solo se ejecutará cuando el documento este listo pero sin esperas innecesarias.

2.3.1 Hacer que algo suceda con un clic

Si tenemos una página con una etiqueta “a” y si se quiere capturar el evento clic sobre la misma sería tan sencillo como:

```
$("a").click (function () {
   alert ("Gracias por visitarnos");
});
```

Claro en este caso además de mostrarse el mensaje se ejecutaría la acción por defecto asociada al vínculo, si se quiere evitar que esto suceda entonces el ejemplo quedaría de la siguiente forma:

```
$("a").click (function () {
   alert ("Gracias por visitarnos");
   return false;
 });
```

Más adelante se ampliará el trabajo con eventos de JQuery.

2.3.2 Modificando la apariencia

Continuando con la etiqueta “a” supongamos que en la página se tiene definido el siguiente estilo.

```
<style type="text/css">
a.test {font-weight: bold ;}
</style>
```

Si se desea aplicar de forma dinámica este estilo a la etiqueta se usará la funcionalidad addClass.

```
$("a").addClass ("test");
```

Convirtiendo el vínculo a negritas, en la sección de trabajo con estilos se verá la amplia gama de funciones de JQuery para el trabajo con la apariencia.

2.4 Efectos especiales

En muchas ocasiones el trabajo con interfaces de usuario requiere de efectos que son difíciles de lograr programando desde cero, más aún teniendo en cuenta las diferencias existentes entre los navegadores más populares.

JQuery brinda opciones sencillas y fáciles de usar para algunos de estos efectos.

```
$("a").click (function () {
  $(this).hide ("slow");
  Return false;
 });
```

Esto ocultaría lentamente el vínculo al hacer clic sobre él.

2.4.1 Encadenamiento

JQuery usa un concepto interesante para hacer su código simple y fácil de entender. Para aquellos relacionados con la programación orientada a objetos esto es algo natural ya que cada método devuelve un objeto JQuery y así se puede ir encadenando el código sin necesidad de usar variables auxiliares.

```
$("a").addClass ("test").show ().html ("foo");
```

Cada uno de estos métodos (addClass, show y html) devuelve una instancia de la clase JQuery y esto es justamente lo que permite ejecutar varias acciones sobre un mismo elemento de forma encadenada.

Los encadenamientos en ocasiones pueden resultar complejos no obstante son bastante claros en su lectura, si se tuvieran los siguientes vínculos en nuestro sitio:

```
<a href="http://intranet.uci.cu/" class="clickme">Te daré un mensaje </a>
<a href="http://inter-nos.uci.cu/" class="hideme">Has clic para ocultarme </a>
<a href="http://localhost/link/">Soy un vínculo normal </a>
```

Se Podría modificar fácilmente la estructura y el comportamiento con la siguiente cadena JQuery:

```
$("a")
  .filter (".clickme")
    .click (function () {
      alert ("Estás saliendo del sitio");
    })
  .end ()
  .filter (".hideme")
    .click (function () {
      $(this).hide ();
      Return false;
    })
  .end ();
```

En este caso al hacer clic sobre el primer vínculo se mostrará el mensaje "Estás saliendo del sitio" y abrirá la página principal de la intranet, el segundo vínculo se ocultará y el tercer link no ha sido modificado.

2.5 El núcleo de JQuery

Después de esta introducción a JQuery veamos cuales son las funcionalidades que posee el núcleo de la librería.

$(html)

Crea elementos del DOM en caliente a partir de una cadena de HTML.

Devuelve: JQuery

Parámetro:

html (String): Una cadena HTML para crear en caliente.

Ejemplo:

Crear un div (con contenido) y añadirlo al cuerpo de la página.

$(" Hola ").appendTo ("body")

$(elems)

Selecciona uno o varios elementos del DOM especificados como parámetros.

Devuelve: JQuery

Parámetro:

elems (Elemento|Array<Elemento>): Elemento del DOM para ser encapsulado como objeto JQuery.

Ejemplos:

Cambiar el color de fondo de la página a negro.

$(document.body).css (“background", "black”);

Ocultar los elementos de un formulario.

$(myForm.elements).hide ()

$(fn)

Esta es una abreviatura de $(document).ready ().

Devuelve: JQuery

Parámetro:

fn (Function): La función a ejecutar cuando el DOM se haya descargado completamente y este listo para la ejecución de scripts.

Ejemplo:

Ejecutar una función cuando el documento esté listo.

```
$(function () {
   alert (“El documento está listo”);
 });
```

$(expr, context)

Esta función recibe una cadena que consiste en un selector CSS o XPath que machea en un contexto determinado.

El núcleo de JQuery gira en torno a esta función. Si el contexto no es especificado entonces se toma como que el mismo es el documento HTML completo.

Devuelve: JQuery

Parámetros:

expr (String): Una expresión a buscar.

context (Elemento|jQuery): (opcional) Un elemento del DOM o un objeto JQuery que sirva como contexto para la búsqueda.

Ejemplos:

Encontrar todos los elementos “p” que son hijos de un elemento “div”

```
$("div > p")
```

El contexto sería todo el documento, si en el mismo tenemos el código HTML uno dos tres el resultado sería el objeto JQuery [two]

Buscar todos los elementos “input” de tipo “radio” en el primer formulario del documento.

```
$("input: radio", document.forms [0])
```

$.fn.extend (prop)

Extiende el objeto JQuery, permitiendo añadirle nuevas funcionalidades, incluyendo métodos de extensiones.

Devuelve: Objeto

Parámetro:

prop (Object): La propiedad que va a ser añadida al objeto JQuery.

Ejemplo:

Añadir dos funciones para trabajar con la propiedad “checked” de los elementos que lo permitan.

```
jQuery.fn.extend ({
  check: function () {
    return this.each (function () {this.checked = true ;});
  },
  uncheck: function () {
    return this.each (function () {this.checked = false ;});
  }
});
$("input [@type=checkbox]").check ();
$("input [@type=radio]").uncheck ();
```

$.noConflict ()

Esta función permite conocer si no existen conflictos con la variable $ que pueda haber sido definida en otras librerías que se estén usando en la misma página.

Devuelve: undefined

each (fn)

Ejecuta una función para cada elemento de una búsqueda anterior, adicionalmente cuando la función se ejecuta recibe como parámetro un entero (que comienza en 0) con el índice del elemento en cuestión.

Devuelve: JQuery

Parámetro:

fn (Function): Función a ejecutar.

Ejemplo:

Modificar el atributo src en todas las imágenes del documento.

```
$("img").each (function (i) {
   this.src = "test" + i + ".jpg";
 });
```

Teniendo un código HTML con dos imágenes <img/><img/> el resultado sería

eq (pos)

Selecciona el dentro que una lista de elementos aquel que tenga la posición especificada.

Devuelve: JQuery

Parámetro:

pos (Number): El índice del elemento al que queremos limitar la búsqueda.

Ejemplo:

Obtener el segundo elemento “p” del documento.

```
$("p").eq (1)
```

get ()

Obtiene los elementos del DOM a partir de los objetos JQuery. Útil si se desea trabajar con el DOM en lugar de con instancias de JQuery.

Devuelve: Array<Elemento>

Ejemplo:

Obtener todas las imágenes del documento como un arreglo de objetos del DOM.

```
$("img").get ();
```

get (pos)

Similar al anterior pero devuelve un solo elemento del DOM especificado por el índice que recibe como parámetro.

Devuelve: Elemento

Parámetro:

pos (Number): Índice del elemento que se desea obtener.

Ejemplo:

Filtrar todas las imágenes del documento y obtener solo la primera como un objeto del DOM.

$("img").get (0);

gt (pos)

Reduce la colección obtenida en una búsqueda a aquellos elementos que se encuentren después de la posición pasada como parámetro.

Devuelve: JQuery

Parámetro:

pos (Number): Índice del elemento después del cual se desea obtener la lista.

Ejemplo:

Obtener todos los elementos “p” del documento a excepción del primero.

$("p").gt (0)

index (subject)

Devuelve el índice que tiene en un búsqueda el elemento que se le pasa como parámetro, si el mismo no se encuentra en la colección resultante de la búsqueda entonces se devuelve -1.

Devuelve: Number

Parámetro:

subject (Elemento): Elemento del que se desea conocer el índice.

Ejemplo:

Devolver el índice del elemento con id foobar.

$("*").index ($('#foobar') [0])

length

Esta propiedad devuelve número de elementos resultantes de una búsqueda.

Devuelve: Number

lt (pos)

Reduce la colección obtenida en una búsqueda a aquellos elementos que se encuentren antes de la posición pasada como parámetro.

Devuelve: JQuery

Parámetro:

pos (Number): Índice del elemento antes del cual se desea obtener la lista.

Ejemplo:

Obtener todos los elementos “p” del documento a excepción del último.

$("p").lt ($("p").length-1)

size ()

Esta función devuelve el mismo valor de la propiedad length, se incluye por compatibilidad con versiones anteriores.

Devuelve: Number.

2.6 Selectores de CSS

JQuery hace un uso extensivo de los selectores de CSS basándose en las definiciones dadas por el W3C para CSS 1, CSS 2 y CSS 3.

Aquí podemos ver una lista de los selectores soportados por JQuery.

*: Cualquier elemento.

E: Un elemento de tipo E.

E: nth-child(n): Un elemento E que sea el hijo n-simo de su padre.

E: first-child: Un elemento E que sea el primer hijo de su padre.

E: last-child: Un elemento E que sea el último hijo de su padre.

E: only-child: Un elemento que sea único hijo.

E: empty: Un elemento E que no tenga hijos (incluyendo nodos de tipo text).

E: enabled: Un elemento E de interfaz que no esté deshabilitado.

E: disabled: Un elemento E de interfaz que esté deshabilitado.

E: checked: Un elemento E de interfaz que esté chequeado.

E: selected: Un elemento E de interfaz que esté seleccionado.

E.myclass: Un elemento E cuya clase sea “myclass”.

E#myid: Un elemento E con ID igual a “myid”.
E: not(s): Un elemento E que no machee con el selector simple “s”.
E F: Un elemento F descendiente de un elemento E.
E > F: Un elemento F hijo de un elemento E.
E + F: Un elemento F inmediatamente precedido de un elemento E.
E ~ F: Un elemento F precedido de un elemento E.
E, F, G: Todos los elementos E, F y G.

2.6.1 Soportados pero diferentes

JQuery soporta todos los selectores de atributo pero colocando @ delante del atributo en cuestión.
E [@foo]: Un elemento E con el atributo “foo” definido.
E@foo=bar]: Un elemento E con el atributo “foo” exactamente igual a “bar”.
E [@foo^=bar]: Un elemento E cuyo atributo “foo” tiene un valor que comienza con “bar”.
E [@foo$=bar]: Un elemento E cuyo atributo “foo” tiene un valor que termina con “bar”.
E [@foo*=bar]: Un elemento E cuyo atributo “foo” tiene un valor que tiene la subcadena “bar”.

2.6.2 No soportados

JQuery solo soporta selectores que se refieran a verdaderos elementos del DOM, el resto es ignorado.
E:link, E:visited, E:active, E:hover, E:focus, E:target, E::first-line, E::first-letter, E::selection, E::before y E::after no son selectores válidos para JQuery.
En adición existen otros selectores que por ser considerados poco útiles tampoco se incluyen dentro del soporte de JQuery.

2.6.3 Selectores XPath

Otro de los lenguajes de selectores que soporta JQuery, además de CSS 1-3, es XPath. La librería soporta expresiones básicas de XPath, a continuación relacionamos algunos ejemplos:

2.6.3.1 Caminos de localización.

Caminos absolutos:

```
$("/html/body//p")
$("/*/body//p")
```

$("//p/../div")

Caminos relativos:

$("a", this)

$("p/a", this)

2.6.4 Selectores de ejes soportados

Elemento descendiente de un elemento descendiente.

$("//div//p")

Elemento hijo de un elemento descendiente.

$("//div/p")

El elemento precedente del hermano tiene un elemento antes de él, en el mismo eje.

$("//div ~ form")

El padre selecciona el elemento del padre del elemento.

$("//div/../p")

2.6.5 Predicados soportados

[@foo]: Tiene un atributo “foo” definido.

$("//input[@checked]")

[@foo='test']: Tiene un atrbuto “foo” igual a “test”.

$("//a[@ref='nofollow']")

[Nodelist]: El elemento contiene una lista de nodos, por ejemplo:

$("//div[p]")

$("//div [p/a]")

2.6.6 Predicados soportados con modificaciones

[last ()] o [position ()=last ()] se cambia por: last

$("p:last")

[0] o [position ()=0] se cambia por: eq (0) o: first

$("p:first")

$("p:eq(0)")

[position () < 5] se cambia por: lt (5)

$("p:lt(5)")
[position () > 2] se cambia por: gt (2)
$("p: gt (2)")

2.6.7 Selectores personalizados

JQuery incluye algunas expresiones que no se encuentran disponibles en CSS o XPath, pero que pueden ser de gran utilidad a los desarrolladores.
even: Selecciona los elementos que ocupan posiciones pares dentro de la colección resultante de una búsqueda.
odd: Selecciona los elementos que ocupan posiciones impares dentro de la colección resultante de una búsqueda.
eq (0) y nth (0): Selecciona el n-simo elemento dentro de la colección resultante de una búsqueda.
gt (N): Selecciona los elementos con índice mayor que N dentro de la colección resultante de una búsqueda.
lt (N): Selecciona los elementos con índice menor que N dentro de la colección resultante de una búsqueda.
first: Equivalente a eq(0)
last: Selecciona el último elemento de la conexión.
parent: Selecciona todos los elementos que tienen hijos (incluyendo nodos de tipo text).
contains ('test'): Selecciona todos los elementos que contienen el texto “test”.
visible: Selecciona todos los elementos visibles.
hidden: Selecciona todos los elementos ocultos.

Algunos ejemplos:

$("p: first").css ("fontWeight","bold");
$("div: hidden").show ();
$("div: contains ('test')").hide ();

2.6.8 Selectores de formularios

Existen algunos selectores especiales para formularios:
input: Selecciona todos los elementos del formulario (input, select, textarea, button).
text: Selecciona todos los campos de tipo texto (type="text").

password: Selecciona todos los campos de tipo password (type="password").
radio: Selecciona todos los campos de tipo radio (type="radio").
checkbox: Selecciona todos los campos de tipo checkbox (type="checkbox").
submit: Selecciona todos los botones submit (type="submit").
image: Selecciona todas las imágenes del formulario (type="image").
reset: Selecciona todos los botones reset (type="reset").
button: Selecciona todos los botones de otro tipo (type="button").
Es recomendable usar en estos casos un contexto:
$('#myForm: radio') ó $('input: radio', myForm)
Esto selecciona todos los campos de tipo radio dentro del formulario myForm. El uso de radio tiene el mismo efecto de [@type=radio], pero es ligeramente más rápido, esto puede ser decisivo en formularios de gran tamaño.

2.7 Usando CSS y XPath al mismo tiempo

Este pudiera ser un punto de confusión para algunos como usar CSS y XPath al mismo tiempo. JQuery implementa algunas concesiones que permiten que esto sea posible, aunque siempre es bueno tener en cuenta las ventajas de cada lenguaje. He aquí algunos ejemplos:
Ocultar todos los párrafos que tengan un link:
$("p[a]").hide ();
Mostrar el primer párrafo de la página:
$("p: eq (0)").show ();
Ocultar todos los div que se estén mostrando:
$("div: visible").hide ();
Obtener todos los elementos en una lista, a partir de una lista sin orden:
$("ul/li")
/* válido también: $("ul > li") */
Obtener todos los párrafos, de clase “foo”, que tengan un link:
$("p.foo[a]");
Obtener el elemento de una lista que tiene un vínculo con el texto Registrar:
$("li [a: contains ('Registrar')]");

Obtener el valor de los campos imput con nombre “bar”:
$("input [@name=bar]").val ();
Obtener todos los radiobuttons que estén chequeados:
$("input [@type=radio] [@checked]");

2.8 Atributos

Existen otra serie de métodos para el trabajo con atributos y propiedades de los elementos, tanto para su lectura como para su modificación, los relacionamos a continuación.

2.8.1 AddClass (class)

Añade la clase especificada a los elementos resultantes de una búsqueda.
Devuelve: JQuery
Parámetro:
class (String): Una o más clases CSS.
Ejemplo:
$("p").addClass ("selected")
HTML de la página:
Hola
Resultado:
[Hola]
attr (name)
Accede a la propiedad pasada como parámetro para el primer elemento de la colección resultante de una búsqueda.
Devuelve: Object
Parámetro:
name (String): Nombre de la propiedad deseada.
Ejemplo:
Devolver el atributo src de la primera imagen del documento.
$("").attr ("src");
HTML de la página:

Resultado:

test.jpg

attr (properties)

Modifica las propiedades pasadas como parámetros de la forma Llave: Valor. Es la mejor forma de modificar varias propiedades al mismo tiempo.

Devuelve: JQuery

Parámetro:

properties (Mapa): Lista de propiedades y sus valores.

Ejemplo:

Modificar los atributos src y alt de todas las imágenes.

$("img").attr ({src: "test.jpg", alt: "Test Image”});

HTML de la página:

</>

Resultado:

attr (key, value)

En este caso modificaríamos una sola propiedad con el valor que pasemos como parámetro.

Devuelve: JQuery

Parámetros:

key (String): El nombre de la propiedad a modificar.

value (Object): El valor que le dará a dicha propiedad.

Ejemplo:

Modificar el atributo src de todas las imágenes.

$("img").attr ("src","test.jpg");

HTML de la página:

</>

Resultado:

attr (key, fn)

Se modifica una propiedad con el valor devuelto por una función que se pase como parámetro.

Devuelve: JQuery

Parámetros:

key (String): El nombre de la propiedad a modificar.

fn (Funtion): Función que devolverá el valor a dar a la propiedad.

Ejemplo:

Modificar el atributo title a partir del atributo src de todas las imágenes.

$("img").attr ("title", function () {return this.src});

HTML de la página:

< src="test.jpg"/>

Resultado:

html ()

Devuelve el contenido HTML del primero de los elementos resultantes de una búsqueda.

Devuelve: String

Ejemplo:

$("div").html ();

HTML de la página:

<input/>

Resultado:

<input/>

html (val)

Modifica el contenido HTML del primero de los elementos resultantes de una búsqueda.

Devuelve: JQuery

Parámetro:

val (String): Contenido HTML que se desea colocar.

Ejemplo:

$("div").html ("new stuff");

HTML de la página:

<input/>

Resultado:

new stuff

removeAttr (name)

Elimina un atributo de los elementos resultantes de una búsqueda.

Devuelve: JQuery

Parámetro:

name (String): Nombre del atributo que desea eliminarse.

Ejemplo:

$("input").removeAttr ("disabled")

HTML de la página:

<input disabled="true"/>

Resultado:

<input/>

removeClass (class)

Elimina la(s) clase(s) especificada(s) de los elementos resultantes de una búsqueda.

Devuelve: JQuery

Parámetro:

class (String): Una o más clases CSS.

Ejemplo:

$("p").removeClass ("selected")

HTML de la página:

Hola

Resultado:

[Hola]

text ()

Devuelve los textos concatenados de todos los elementos resultantes de una búsqueda.

Devuelve: String

Ejemplo:

Devolver el texto de todos los párrafos del documento.

$("p").text ();

HTML de la página:

Test Paragraph. Paragraph

Resultado:

Test Paragraph.Paragraph

text (val)

Modifica los textos de todos los elementos resultantes de una búsqueda. Similar a html (val).

Devuelve: String

Parámetro:

val (String): Texto a colocar dentro de los elementos a modificar.

Ejemplo:

$("p").text ("Un texto nuevo.")

HTML de la página:

Párrafo de prueba

Resultado:

Un texto nuevo.

toggleClass (class)

Añade la clase especificada si no está presente y la elimina si ya existe en los elementos resultantes de una búsqueda.

Devuelve: JQuery

Parámetro:

class (String): Una o más clases CSS.

Ejemplo:

$("p").toggleClass ("selected")

HTML de la página:

Hello Hello Again

Resultado:

[Hello , Hello Again]

val ()

Devuelve el valor del atributo value del primero de los elementos de la colección resultante de una búsqueda.

Devuelve: JQuery

Ejemplo:

$("input").val ();

HTML de la página:

<input type="text" value="some text"/>

Resultado:

"some text"

val (val)

Modifica el atributo value de los elementos de la colección resultante de una búsqueda.

Devuelve: JQuery

Parámetro:

val (Object): Valor que se desea modificar.

Ejemplo:

$("input").val ("test");

HTML de la página:

<input type="text" value="some text"/>

Resultado:

<input type="text" value="test"/>

2.9 Traversing

Add (expr)

Adiciona elementos que son comparados con una expresión dada, al conjunto de elementos seleccionados.

Devuelve: JQuery

Parámetro:

expr (String): Una expresión que corrobora que elementos serán adicionados.

Ejemplo:

Compare el resultado dado con $ (‘p ’), en este caso p seria [Hola], usar add (), comparar los elementos con $ (‘span’), estos serán simplemente adicionados en la lista de objetos JQuery.

$("p").add ("span")

Antes:

(html) Hola Hola Otra Vez (/html)

Resultado:

Se encontraron dos elementos.

add (html)

Adiciona elementos creados rápido al conjunto de elementos seleccionados.

Devuelve: JQuery

Parámetro:

html (string): Una expresión HTML que se crea rápido.

Ejemplo:

$("p").add (" Otra Vez ")

Antes:

Hola

Resultado:

[Hola , Otra Vez]

add (elements)

Adiciona uno a más elementos al conjunto.

Devuelve: JQuery

Parámetro:

elements (Element|Array<Element>): Uno a más elementos para adicionar.

Ejemplo:

$("p").add (document.getElementById ("a"))

Antes:

Hola Hola Otra Vez

Resultado:

[Hola , Hola Otra Vez]

children (expr)

Obtener un set de elementos que contenga todos lo hijos únicos de cada elemento similar del set. Este puede ser filtrado con una expresión, con la cual se obtendrán solamente los elementos que coincidan al ser comparados con la expresión dada.

Devuelve: JQuery

Parámetro:

expr (String): (opcional) una expresión para filtrar los elementos del hijo.

Ejemplo:

Encuentre a todos los hijos de cada div.

```
$("div").children ()
```

Antes:

```
<p>Hola</p><div><span>Hola Otra Vez</span></div><p>Y Otra Vez</p>
```

Resultado:

```
[<span>Hola Otra Vez</span>]
```

Ejemplo:

Encuentre a todos los hijos con una clase "seleccionada" de cada div.

```
$("div").children (".selected")
```

Antes:

```
<div><span>Hola</span><p class="selected">Hola Otra Vez</p><p>Y Otra Vez</p></div>
```

Resultado:

```
[<p class="selected">Hola Otra Vez</p>]
```

contains (str)

Filtra en el sistema de elementos a los que contengan el texto especificado.

Devuelve: JQuery

Parámetro:

str (String): La secuencia que será contenida dentro del texto de un elemento.

Ejemplo:

```
$("p").contains ("test")
```

Antes:

```
<p>this is just a test. </p><p>So is this</p>
```

Resultado:

```
[<p>this is just a test. </p>]
```

end ()

Invierte la operación ' destructiva ' más reciente, cambiando el conjunto de elementos a su estado anterior (justo antes de la operación destructiva). Si no había operación destructiva antes, se devuelve un sistema vacío. Una operación ' destructiva ' es cualquier operación que cambie el conjunto de elementos del JQuery. Estas funciones son: add, children, clone, filter, find, not, next, parent, parents, prev and siblings.

Devuelve: JQuery

Ejemplo:

Selecciona todos los párrafos, encuentra elementos span dentro de éstos, e invierte la selección de nuevo a los párrafos.

```
$("p").find ("span").end ();
```

Antes:

```
<p><span>Hola</span>, how are you? </p>
```

Resultado:

```
[<p>...</p>]
```

filter (expression)

Quita los elementos de un conjunto de elementos que no coincidan con la expresión dada. Este método se utiliza para estrechar los resultados de la búsqueda. Proporciona una lista de elementos a la que se le aplican los filtros múltiples inmediatamente.

Devuelve: JQuery

Parámetro:

expression (String): Expresión (s) a buscar con:

Ejemplo:

Selecciona todos los párrafos y quita éstos sin una clase "seleccionada".

```
$("p").filter (".selected")
```

Antes:

```
<p class="selected">Hola</p><p>How are you? </p>
```

Resultado:

```
[<p class="selected">Hola</p>]
```

Ejemplo:

Selecciona todos los párrafos y quita éstos sin la clase "seleccionada" que es la primera.

```
$("p").filter (".selected: first")
```

Antes:

```
<p>Hola</p><p>Hola Otra Vez</p><p class="selected">Y Otra Vez</p>
```

Resultado:

```
[<p>Hola</p>, <p class="selected">Y Otra Vez</p>]
```

filter (filter)

Elimina del conjunto de elementos aquellos que no pasan por el filtro especificado, lo que permite estrechar aun más el resultado.

Devuelve: JQuery

Parámetro:

filter (Function): Una función a utilizar para la filtración.

Ejemplo:

Quite todos los elementos que tengan un elemento del hijo de “ol”.

```
$("p").filter (function (index) {
  return $("ol", this).length == 0;
})
```

Antes:

```
<p><ol><li>Hola</li></ol></p><p>Cómo estas?</p>
```

Resultado:

```
[<p>Cómo estas?</p>]
```

find(expr)

Busca todos los elementos que coincidan con la expresión dada, este método es una manera ideal de buscar elementos adicionales los cuales serán procesados. Toda la búsqueda se realiza utilizando expresiones de JQuery, la expresión se puede escribir usando sintaxis del selector del CSS 1-3, o XPath básico.

Devuelve: JQuery

Parámetro:

expr (String): Una expresión a buscar.

Ejemplo:

Comenzar con todos los párrafos y buscadores que desciendan de los elementos span, tales como $("p span")

```
$("p").find ("span");
```

Antes:

```
<p><span>Hola</span>, como estas?</p>
```

Resultado:

```
[<span>Hola</span>]
```

is (expr)

Comprueba la selección actual contra una expresión y retorna verdadero, si por lo menos un elemento de la selección es encontrado en la expresión dada. Retorna falso, si no hay ajustes del elemento o la expresión inválidos. El filter (String) se utiliza internamente, por lo tanto todas las reglas que se aplican allí se aplican aquí también.

Devuelve: Booleano.

Parámetro:

expr (String): La expresión con la cual filtraremos.

Ejemplo:

```
$("input[@type='checkbox']").parent().is("form")
```

Antes:

```
<form><input type="checkbox" /></form>
```

Resultado:

Verdadero

Ejemplo:

```
$("input[@type='checkbox']").parent().is("form")
```

Antes:

```
<form><p><input type="checkbox" /></p></form>
```

Resultado:

Falso

not(expr)

Elimina los elementos que coinciden con la expresión dada. Este método se utiliza para quitar uno o más elementos de un objeto JQuery.

Devuelve: JQuery

Parámetro:

expr (String): Una expresión con la cual se puede quitar los elementos con los que coincide.

Ejemplo:

Quita el elemento con la identificación "seleccionada" del conjunto de todos los párrafos.

```
$("p").not ("#selected")
```

Antes:

```
<p>Hola</p><p id="selected">Hola Otra Vez</p>
```

Resultado:

```
[<p>Hola</p>]
```

not (elems)

Quita cualquier elemento dentro del conjunto de elementos. Este método se utiliza para quitar uno o más elementos de un objeto del JQuery, la expresión no puede utilizar una referencia al nombre del elemento.

Devuelve: JQuery

Parámetro:

elems (JQuery): Un sistema de elementos a quitar del conjunto de elementos.

Ejemplo:

Quita todos los elementos que coinciden "div p.selected" del total de todos los párrafos.

```
$("p").not ($("div p.selected"))
```

Antes:

```
<div><p>Hola</p><p class="selected">Hola Otra Vez</p></div>
```

Resultado:

```
[<p>Hola</p>]
```

parent(expr)

Devuelve un grupo de elementos que contienen a los padres únicos del conjunto de elementos. Se puede utilizar una expresión opcional para filtrar el conjunto de elementos del padre.

Devuelve: JQuery

Parámetro:

expr (String): Se puede utilizar una expresión con la cual se va a filtrar los padres.

Ejemplo:

Encuentre el elemento del padre de cada párrafo.

```
$("p").parent()
```

Antes:

```
<div><p>Hola</p><p>Hola</p></div>
```

Resultado:

```
[ <div><p>Hola</p><p>Hola</p></div> ]
```

Ejemplo:

Encuentre el elemento del padre de cada párrafo con una clase "seleccionada".

```
$("p").parent(".selected")
```

Antes:

```
<div><p>Hola</p></div><div class="selected"><p>Hola Otra Vez</p></div>
```

Resultado:

```
[<div class="selected"><p>Hola Otra Vez</p></div>]
```

parents (expr)

Retorna un grupo de elementos que contiene a los únicos antepasados del conjunto de elementos (a excepción del elemento de la raíz). Los elementos se pueden filtrar con una expresión opcional.

Devuelve: JQuery

Parámetro:

expr (String): Una expresión para filtrar a los antepasados es opcional.

Ejemplo:

Encuentre todos los elementos del padre.

```
$("span").parents ()
```

Antes:

```
<html><body><div><p><span>Hola</span></p><span>Hola Otra Vez</span></div></body></html>
```

Resultado:

```
[<body>...</body>, <div>...</div>, <p><span>Hola</span></p>]
```

Ejemplo:

Encuentre todos los elementos del padre que sean un párrafo.

```
$("span").parents("p")
```

Antes:

<html><body> Hola Hola Otra Vez </body></html>

Resultado:

[Hola]

prev (expr)

Retorna un grupo de elementos que contienen a los hermanos anteriores únicos de cada uno del conjunto de elementos. Se puede utilizar una expresión opcional para filtrar el conjunto de elementos Solamente retorna el hermano inmediatamente anterior, no todos los hermanos anteriores.

Devuelve: JQuery

Parámetro:

expr (String): Una expresión para filtrar los elementos anteriores.

Ejemplo:

Encuentre el hermano anterior de cada párrafo.

$("p").prev()

Antes:

Hola Hola Otra Vez Y Otra Vez

Resultado:

[Hola Otra Vez]

Ejemplo:

Encuentre el hermano anterior de cada párrafo que tenga una clase "seleccionada".

$("p").prev(".selected")

Antes:

Hola Hola Otra Vez Y Otra Vez

Resultado:

[Hola]

siblings (expr)

Retorna un grupo de elementos que contienen a todos los hermanos únicos de cada uno del conjunto de elementos. Se puede filtrar con las expresiones opcionales.

Devuelve: JQuery

Parámetro:

expr (String): Una expresión para filtrar los elementos del hermano.

Ejemplo:

Encuentre los hermanos de cada div.

```
$("div").siblings()
```

Antes:

```
<p>Hola</p><div><span>Hola Otra Vez</span></div><p>Y Otra Vez</p>
```

Resultado:

```
[ <p>Hola</p>, <p>Y Otra Vez</p> ]
```

Ejemplo:

Encuentre a todos los hermanos con una clase "seleccionada" de cada div.

```
$("div").siblings(".selected")
```

Antes:

```
<div><span>Hola</span></div><p class="selected">Hola Otra Vez</p><p>Y Otra Vez</p>
```

Resultado:

```
[<p class="selected">Hola Otra Vez</p>
```

2.10 Manipulation

after (content)

Inserte el contenido después de cada uno de los elementos del conjunto.

Devuelve: JQuery

Parámetro:

content (<Content>): Contenido a insertar después de cada blanco.

Ejemplo:

Inserta algún HTML después de todos los párrafos.

```
$("p").after ("<b>Hola</b>");
```

Antes:

```
<p>Quisiera decir: </p>
```

Resultado:

```
<p>Quisiera decir: </p><b>Hola</b>
```

Ejemplo:

Inserta un elemento después de todos los párrafos.

```
$("p").after( $("#foo")[0] );
```

Antes:

```
<b id="foo">Hola</b><p>Quisiera decir: </p>
```

Resultado:

```
<p>Quisiera decir: </p><b id="foo">Hola</b>
```

Ejemplo:

Inserta un objeto del JQuery (similar a un arreglo de elementos DOM) después de todos los párrafos.

```
$("p").after( $("b") );
```

Antes:

```
<b>Hola</b><p>Quisiera decir: </p>
```

Resultado:

```
<p>Quisiera decir: </p><b>Hola</b>
```

append(content)

Añade el contenido dentro de cada elemento que coincide con la expresión. Esta operación es similar a hacer un appendChild a todos los elementos especificados, agregándolos en el documento.

Devuelve: JQuery

Parámetro:

content (<Content>): Contenido a añadir al blanco.

Ejemplo:

Añade algún HTML a todos los párrafos.

```
$("p").append ("<b>Hola</b>");
```

Antes:

```
<p>Quisiera decir: </p>
```

Después:

```
<p>Quisiera decir: <b>Hola</b></p>
```

Ejemplo:

Añade un elemento a todos los párrafos.

```
$("p").append( $("#foo")[0] );
```

Antes:

Quisiera decir: Hola

Después:

Quisiera decir: Hola

Ejemplo:

Añade un objeto del JQuery (similar a un conjunto de elementos de DOM) a todos los párrafos.

$("p").append($("b"));

Antes:

Quisiera decir: Hola

Después:

Quisiera decir: Hola

appendTo(content)

Añade todos los elementos especificados a otros, que pertenecen al conjunto de elementos. Esta operación es, esencialmente, el revés de hacer un $(a).append(b) regular, o sea lejos de añadir B a A , se está añadiendo A a B.

Devuelve: JQuery

Parámetro:

content (<Content>): Contenido a añadir al elemento seleccionado.

Ejemplo:

Añade donde quiera que este el elemento "foo" todos los párrafos.

$("p").appendTo("#foo");

Antes:

Quisiera decir:

Resultado:

Quisiera decir:

before(content)

Inserte el contenido antes de cada uno de los elementos del conjunto.

Devuelve: JQuery

Parámetro:

content (<Content>): Contenido a insertar antes de cada blanco.

Ejemplo:

Inserta algún HTML antes de todos los párrafos.

```
$("p").before ("<b>Hola</b>");
```

Antes:

```
<p>Quisiera decir: </p>
```

Resultado:

```
<b>Hola</b><p>Quisiera decir: </p>
```

Ejemplo:

Inserta un elemento antes de todos los párrafos.

```
$("p").before( $("#foo")[0] );
```

Antes:

```
<p>Quisiera decir: </p><b id="foo">Hola</b>
```

Resultado:

```
<b id="foo">Hola</b><p>Quisiera decir: </p>
```

Ejemplo:

Inserta un objeto del JQuery (similar a un conjunto de elementos de DOM) antes de todos los párrafos.

```
$("p").before( $("b") );
```

Antes:

```
<p>Quisiera decir: </p><b>Hola</b>
```

Resultado:

```
<b>Hola</b><p>Quisiera decir: </p>
```

clone(deep)

Reproduce los elementos del DOM que coinciden, hace una selección de los clones. Esto es útil para mover las copias de los elementos a otra localización en el DOM.

Devuelve: JQuery

Parámetro:

deep (Boolean): Resulta falso si no se desea reproducir todos los nodos del descendiente, además del elemento sí mismo.

Ejemplo:

Reproduce todos los elementos de b (y selecciona el que se reproduce) e insértelos a todos los párrafos.

```
$("b").clone().prependTo("p");
```

Antes:

Hola , como estas?

Resultado:

Hola Hola, como estas?

empty()

Quita todos los nodos hijos del conjunto de elementos.

Devuelve: String

Ejemplo:

$("p").empty()

Antes:

Hola, Person and person

Resultado:

[ ]

insertAfter(content)

Inserte todos los elementos especificados detrás de los otros del conjunto de elementos. Esta operación es, esencialmente, el revés de hacer un $(a).after(b) regular, en eso en vez de insertar B después de A, usted está insertando A después de B.

Devuelve: JQuery

Parámetro:

content (<Content>): Contenido para insertar el elemento seleccionado después.

Ejemplo:

Same as $("#foo").after ("p")

$("p").insertAfter("#foo");

Antes:

Quisiera decir: Hola

Resultado:

Hola Quisiera decir:

insertBefore (content)

Inserte todos los elementos dados antes de los especificados del conjunto de elementos. Esta operación es, esencialmente, el revés de hacer un $(a).before(b) regular, en eso en vez de insertar B antes de A, usted está insertando A antes de B.

Devuelve: JQuery

Parámetro:

content (<Content>): Contenido para insertar el elemento seleccionado antes.

Ejemplo:

Same as $("#foo").before ("p")

```
$("p").insertBefore("#foo");
```

Antes:

```
<div id="foo">Hola</div><p>Quisiera decir: </p>
```

Resultado:

```
<p>Quisiera decir: </p><div id="foo">Hola</div>
```

prepend(content)

Inserta el contenido en el interior de cada elemento que coincida. Esta operación es la mejor manera de insertar elementos, al principio de todos los elementos emparejados.

Devuelve: JQuery

Parámetro:

content (<Content>): Contenido a insertar.

Ejemplo:

Inserte algún HTML a todos los párrafos.

```
$("p").prepend ("<b>Hola</b>");
```

Antes:

```
<p>Quisiera decir: </p>
```

Resultado:

```
<p><b>Hola</b>Quisiera decir: </p>
```

Ejemplo:

Inserte un elemento a todos los párrafos.

```
$("p").prepend( $(".foo")[0] );
```

Antes:

```
<p>Quisiera decir: </p>
<p>Quisiera decir: </p>
<b class="foo">Hola</b>
<b class="foo">Adios</b>
```

Resultado:

```
<p><b class="foo">Hola</b>Quisiera decir: </p>
<p><b class="foo">Hola</b>Quisiera decir: </p>
```

(observe [0], de que indica que inserte solamente el primer elemento de $(".foo"))

Ejemplo:

Inserte un objeto del JQuery (similar a un conjunto de elementos de DOM) a todos los párrafos.

```
$("p").prepend( $("b") );
```

Antes:

```
<p>Quisiera decir: </p><b>Hola</b>
```

Resultado:

```
<p><b>Hola</b>Quisiera decir: </p>
```

prependTo(content)

Inserte todos los elementos a otros específicos en el conjunto de elementos. Esta operación es, esencialmente, el revés de hacer un $(a).prepend(b) regular, en eso en vez de insertar B a A, se inserta A a B.

Devuelve: JQuery

Parámetro:

content (<Content>): Contenido a insertar al elemento seleccionado.

Ejemplo:

Inserte todos los párrafos al elemento con "foo"

```
$("p").prependTo("#foo");
```

Antes:

```
<p>Quisiera decir: </p><div id="foo"><b>Hola</b></div>
```

Resultado:

```
<div id="foo"><p>Quisiera decir: </p><b>Hola</b></div>
```

remove(expr)

Quita todos los elementos que han coincidido del DOM. Esto no los quita del objeto JQuery, permitiendo que se utilicen los elementos más adelante. Se pueden filtrar con las expresiones opcionales.

Devuelve: JQuery

Parámetro:

expr (String): Expresión JQuery para filtrar elementos.

Ejemplo:

```
$("p").remove();
```

Antes:

```
<p>Hola</p> como estas <p>tu?</p>
```

Resultado:

```
como estas?
```

Ejemplo:

```
$("p").remove(".hello");
```

Antes:

```
<p class="hola">Hola</p> como estas <p>tu?</p>
```

Resultado:

```
Como estas <p>tu?</p>
```

2.11 CSS

css(name)

Brinda el acceso a una característica del estilo en el primer elemento del conjunto. Con este método se hace fácil recuperar un valor, de características, estilo, del primer elemento del conjunto.

Devuelve: String

Parámetro:

name (String): El nombre de la característica de acceso.

Ejemplo:

Recupera el estilo del color del primer párrafo

```
$("p").css("color");
```

Antes:

Texto del Párrafo.

Resultado:

"rojo"

Ejemplo:

Recupera el estilo de la fuente del primer párrafo.

$("p").css ("font-weight");

Antes:

Texto del Párrafo.

Resultado:

"negrita"

css(properties)

Establece un objeto con determinadas características y propiedades con el cual se van a comparar los elementos. Esto sirve como la mejor manera de fijar una gran cantidad de características de estilo en todos los elementos del conjunto.

Devuelve: JQuery

Parámetro:

properties (Mapa): Pares de llave/valor a fijar como características del estilo.

Ejemplo:

Fija estilos del color y del fondo a todos los elementos p

$("p").css ({color: "rojo", background: "azul”});

Antes:

Texto del Párrafo.

Resultado:

Texto del Párrafo.

css(key, value)

Establece una sola característica de estilo a un valor, en todos los elementos del conjunto. Si se proporciona un número, se convierte automáticamente en un valor del píxel.

Devuelve: JQuery

Parámetros:

key (String): El nombre de la característica a fijar.

value (String|Number): El valor para fijar la característica.

Ejemplo:

Cambia el color de todos los párrafos a rojo

```
$("p").css("color", “red");
```

Antes:

```
<p>Texto del Párrafo.</p>
```

Resultado:

```
<p style="color: red;">Texto del Párrafo.</p>
```

Ejemplo:

Cambia el margen izquierdo de todos los párrafos a "30px"

```
$("p").css("left",30);
```

Antes:

```
<p>Texto del Párrafo.</p>
```

Resultado:

```
<p style="left: 30px;">Texto del Párrafo. </p>
```

height()

Consigue la corriente computada, el píxel, altura del primer elemento del conjunto.

Devuelve: String

Ejemplo:

```
$("p").height();
```

Antes:

```
<p>Este es justo el texto.</p>
```

Resultado:

```
300
```

height(val)

Establece la altura del CSS de cada elemento del conjunto. Si no se especificó ninguna unidad explícita (como el ' em ' o ' % ') entonces el "px" se agrega a la anchura.

Devuelve: JQuery

Parámetro:

val (String|Number): Fija la característica del CSS al valor especificado.

Ejemplo:

```
 $("p").height(20);
```

Antes:

```
 <p>Este es justo el texto.</p>
```

Resultado:

```
 <p style="height:20px;">Este es justo el texto.</p>
```

Ejemplo:

```
 $("p").height("20em");
```

Antes:

```
 <p>Este es justo el texto.</p>
```

Resultado:

```
 <p style="height:20em;">Este es justo el texto.</p>
$("p").width("20em");
```

Antes:

```
 <p>Este es justo el texto.</p>
```

Resultado:

```
 <p style="width: 20em;">Este es justo el texto. </p>
```

2.12 Efectos

2.12.1 Animaciones (Parámetros, velocidad, facilidad, callback)

Esta función se puede utilizar para realizar una animación personalizada. El aspecto clave de esta función es el estilo y las características del objeto que será animado. Cada llave dentro del objeto representa una característica del estilo que también será animada (por ejemplo: "height", "top", or "opacity").

Observe que las características se deben especificar usando el marginLeft del camel case lejos de utilizar el margin-left.

El valor asociado a la llave representa a qué extremo será animada la característica. Si un número se proporciona como el valor, entonces la característica del estilo transitará de su estado actual a ese nuevo número. Si no se proporciona la secuencia "hide", "show", o "toggle", será construida una animación por defecto para esa característica.

Devuelve: JQuery

Parámetros:

params (Hash): Un sistema de cualidades del estilo se desea animar, y a qué extremo.

speed (String|Number): Secuencia de A que representa una de las tres velocidades predefinidas ("retárdese", "normal", o "rápido") o del número de milisegundos para funcionar la animación (e.g. 1000).

easing (String): El nombre del efecto fácil que se desea utilizar (plugin requerida).

callback (Function): La función (opcional) de A que se ejecutará siempre que la animación termine, se ejecuta una vez para cada elemento animado. A esta función se le pasa un solo parámetro, el elemento que será animado.

Ejemplo:

```
 $("p").animate ({
  height: 'toggle', opacity: 'toggle'
}, "slow");
```

Ejemplo:

```
 $("p").animate ({
  left: 50, opacity: 'show'
}, 500);
```

Ejemplo:

Un ejemplo de utilizar una función fácil para proporcionar un estilo diferente a la animación. Esto trabajará solamente si se tiene un plugin que proporcione esta función fácil (' linear ' es proporcionado solamente por defecto, en JQuery).

```
$("p").animate ({
  opacity: 'show'
}, "slow", "easein");
```

2.12.2 fadeIn (speed, callback)

Solamente la opacidad se ajusta según esta animación, significando que todos los elementos del conjunto deben ya tener cierta altura y de anchura asociados a ellos.

Devuelve: JQuery

Parámetros:

speed (String|Number): Secuencia de A que representa uno de las tres velocidades predefinidas ("retárdese", "normal", o "rápido") o del número de milisegundos para funcionar la animación (e.g. 1000).
callback (Function): La función (opcional) de A que se ejecutará siempre que la animación termine, se ejecuta una vez para cada elemento animado.

Ejemplo:

```
$("p").fadein ("slow");
```

Ejemplo:

```
$("p").fadeIn ("slow", function () {
   alert ("Animation Done.");
 });
```

2.12.3 fadeout (speed, callback)

Solamente la opacidad se ajusta según esta animación, significando que todos los elementos del conjunto deben ya tener cierta altura y de anchura asociados a ellos.

Devuelve: JQuery

Parámetros:

speed (String|Number): Secuencia (opcional) de A que representa uno de las tres velocidades predefinidas ("retárdese", "normal", o "rápido") o del número de milisegundos para funcionar la animación (e.g. 1000).
callback (Function): La función (opcional) de A que se ejecutará siempre que la animación termine, se ejecuta una vez para cada elemento animado.

Ejemplo:

```
$("p").fadeout ("slow");
```

Ejemplo:

```
$("p").fadeOut ("slow", function () {
   alert ("Animation Done."); });
```

2.12.4 fadeTo (speed, opacity, callback)

Solamente la opacidad se ajusta según esta animación, significando que todos los elementos del conjunto deben ya tener cierta altura y de anchura asociados a ellos.

Devuelve: JQuery

Parámetros:

speed (String|Number): Una secuencia que representa uno de las tres velocidades predefinidas ("retárdese", "normal", o "rápido") o del número de milisegundos para funcionar la animación (e.g. 1000).

opacity (Number): La opacidad se descolora (un número a partir de la 0 a 1).

callback (Function): La función (opcional) de A que se ejecutará siempre que la animación termine, se ejecuta una vez para cada elemento animado.

Ejemplo:

```
 $("p").fadeto ("slow", 0.5);
```

Ejemplo:

```
$("p").fadeTo ("slow", 0.5, function () {
   alert ("Animation Done.");
 });
```

2.12.5 hide ()

Oculta cada uno de los elementos del conjunto si estos se demuestran.

Devuelve: JQuery

Ejemplo:

```
 $("p").hide ()
```

Antes:

```
 < p>Hello</p >
```

Resultado:

```
 [< p style="display: none">Hello</p >]
```

hide (speed, callback)

2.12.6 hide (speed, callback)

Oculta todos los elementos del conjunto usando una animación agraciada y enciende un servicio de callback completamente opcional al terminar.

La altura, la anchura, y la opacidad de cada uno de los elementos del conjunto se cambian dinámicamente según la velocidad especificada.

Devuelve: JQuery

Parámetros:

speed (String|Number): Una secuencia que representa uno de las tres velocidades predefinidas ("retárdese", "normal", o "rápido") o del número de milisegundos para funcionar la animación (e.g. 1000).

callback (Function): La función (opcional) de A que se ejecutará siempre que la animación termine, se ejecuta una vez para cada elemento animado.

Ejemplo:

```
 $("p").hide ("slow");
```

Ejemplo:

```
$("p").hide ("slow", function () {
   alert ("Animation Done.");
 });
```

2.12.7 show ()

Exhibe cada uno de los elementos del conjunto si se ocultan.

Devuelve: JQuery

Ejemplo:

```
 $("p").show ()
```

Antes:

```
 < p style="display: none">Hello</p >
```

Resultado:

```
 [< p style="display: block">Hello</p >]
```

2.12.8 show (speed, callback)

Demuestra todos los elementos del conjunto usando una animación agraciada y enciende un servicio de callback completamente opcional al terminar.

La altura, la anchura, y la opacidad de cada uno de los elementos emparejados se cambian dinámicamente según la velocidad especificada.

Devuelve: JQuery

Parámetros:

speed (String|Number): Una secuencia que representa uno de las tres velocidades predefinidas ("retárdese", "normal", o "rápido") o del número de milisegundos para funcionar la animación (e.g. 1000).

callback (Function): La función (opcional) de A que se ejecutará siempre que la animación termine, se ejecuta una vez para cada elemento animado.

Ejemplo:

```
$("p").show ("slow");
```

Ejemplo:

```
$("p").show ("slow", function () {
  alert ("Animation Done.");
});
```

2.12.9 slideDown (speed, callback)

Revela todos los elementos del conjunto usando una animación agraciada y enciende un servicio de callback completamente opcional al terminar.

Solamente la altura se ajusta en esta animación, haciendo que todos los elementos del conjunto que coincidan sean revelados de manera descendiente.

Devuelve: JQuery

Parámetros:

speed (String|Number): Secuencia (opcional) de A que representa uno de las tres velocidades predefinidas ("retárdese", "normal", o "rápido") o del número de milisegundos para funcionar la animación (e.g. 1000).

callback (Function): La función (opcional) de A que se ejecutará siempre que la animación termine, se ejecuta una vez para cada elemento animado.

Ejemplo:

```
$("p").slideDown ("slow");
```

Ejemplo:

```
$("p").slideDown ("slow", function () {
  alert ("Animation Done.");
});
```

2.12.10 slideToggle (speed, callback)

Acciona la visibilidad de todos los elementos del conjunto usando una animación agraciada y enciende un servicio de callback completamente opcional al terminar.

Solamente la altura se ajusta en esta animación, haciendo que todos los elementos del conjunto que coincidan sean revelados de manera descendiente.

Devuelve: JQuery

Parámetros:

speed (String|Number): Secuencia (opcional) de A que representa uno de las tres velocidades predefinidas ("retárdese", "normal", o "rápido") o del número de milisegundos para funcionar la animación (e.g. 1000).

callback (Function): La función (opcional) de A que se ejecutará siempre que la animación termine, se ejecuta una vez para cada elemento animado.

Ejemplo:

```
$("p").slidetoggle ("slow");
```

Ejemplo:

```
$("p").slideToggle ("slow", function () {
   alert ("Animation Done.");
 });
```

2.12.11 slideUp (speed, callback)

Acciona la visibilidad de todos los elementos del conjunto usando una animación agraciada y enciende un servicio de callback completamente opcional al terminar.

Solamente la altura se ajusta según esta animación, hace que todos los elementos del conjunto sean ocultados de manera descendiente.

Devuelve: JQuery

Parámetros:

speed (String|Number): Secuencia (opcional) de A que representa uno de las tres velocidades predefinidas ("retárdese", "normal", o "rápido") o del número de milisegundos para funcionar la animación (e.g. 1000).

callback (Function): La función (opcional) de A que se ejecutará siempre que la animación termine, se ejecuta una vez para cada elemento animado.

Ejemplo:

```
$("p").slidetoggle ("slow");
```

Ejemplo:

```
$("p").slideUp ("slow", function () {
  alert ("Animation Done.");
});
```

2.13 AJAX

2.13.1 Características de $.ajax ()

Carga una página remota usando una petición del HTTP.

Si se revisa $.get, $.post etc. para las abstracciones de alto nivel que son a menudo más fáciles de entender y de utilizar, pero no ofrecen tanta funcionalidad (tal como servicios callback del error).

$.ajax () retorna el XMLHttpRequest que crea. En la mayoría de los casos no se necesitará manipular ese objeto directamente, pero está disponible si se necesita abortar la petición manualmente.

Nota: Si se especifica la opción del dataType descrita abajo, es necesario cerciorarse de que el servidor envíe el MIME (Extensiones de Correo de Internet de Propósitos Múltiples) correcto como respuesta (eg. xml como "text/xml"). Enviar el tipo incorrecto del MIME puede conducir a los problemas inesperados en su escritura.

2.13.1.1 Opciones

$.ajax () coge como argumento un par de objetos llave/valor, que se utilizan para inicializar y manejan la petición. Éstos son todos los llaves/valores que pueden ser utilizados:

async (boleano): Por defecto, todas las peticiones se envían de manera asincrónica (e.g. éste está fijado para retornar verdad por defecto). Si se necesitan peticiones síncronas, fije esta opción a falso. Observe que las peticiones síncronas pueden trabar temporalmente el buscador, inhabilitando cualquier acción mientras que la petición este activa.

beforeSend (función): Una pre-callback para modificar el objeto de XMLHttpRequest antes de que se envíe. El XMLHttpRequest se pasa como el único argumento.

complete (función): Una función que se llamará cuando la petición se haya acabado (después de los callbakcs de éxito y de error se ejecutan). A esta función se le pasan dos argumentos: El objeto de XMLHttpRequest y una cadena que especifica el tipo de petición.

contentType (String): Al enviar datos al servidor, se utiliza este content-type. Por defecto es "application/x-www-form-urlencoded", que está muy bien para la mayoría de los casos.

datos (Object|String): Son datos que se enviarán al servidor, lo convierte a una secuencia de JQuery, sino es ya una secuencia, se añade al URL para el Get -requests. Vea la opción del processData para prevenir este proceso automático. El objeto debe ser pares de llave/valor. Si el valor es un atributo, el JQuery serializa valores múltiples con la misma llave es decir {foo: ["bar1", "bar2"]} se convierte en
' &foo=bar1&foo=bar2'.

dataType (String): El tipo de datos que se espera del servidor. Si no se especifica ninguno, el JQuery pasará inteligentemente el responseXML o el responseText que permite un exitoso callback, basado en el tipo del MIME de la respuesta. Los tipos disponibles (y el resultado pasado como primer argumento al callback) son:

"xml": Devuelve un documento de XML que se pueda procesar vía JQuery.

"HTML": Devuelve HTML como texto en claro; se evalúan los scripts incluidos los de la escritura.

"script": Evalúa la respuesta como JavaScript y la devuelve como texto en claro.

"json": Evalúa la respuesta como JSON y devuelve un objeto de JavaScript.

error (función): Una función que se llamará si la petición falla. La función tiene tres argumentos: El objeto de XMLHttpRequest, una secuencia que describe el tipo de error que ocurrió y un objeto opcional de la excepción, si ocurrió uno.

ifModified (boleano): Permiten la petición de ser acertado solamente si la respuesta ha cambiado desde la petición pasada. Esto es hecho comprobando el encabezado que fue modificado por última vez. Si el valor predeterminado es falso, se ignora el encabezado.

processData (boleano): Por el defecto, datos pasados a la opción de los datos como objeto (técnicamente, cualquier cosa con excepción de una secuencia) será procesado y transformado en una secuencia, dando lugar al contenido-tipo "application/x-www-form-urlencoded" del defecto. Si usted desea enviar DOMDocuments, u otros datos no-procesados, fije esta opción a falso.

success (función): Una función que se llamará si la petición tiene éxito. A la función se le pasa un argumento: Los datos que devuelve el servidor, ajustado al formato del parámetro ' dataType'.

timeout (número): Fija un descanso local para la petición. Esto eliminará el descanso global, si uno se fija vía $.ajaxtimeout. Por ejemplo, se podría utilizar esta característica para dar a una sola petición un

descanso más largo que el resto de las peticiones que se han fijado al tiempo de espera en un segundo. Vea $.ajaxtimeout () para los descansos globales.

type (String): El tipo de petición de hacer ("POST" o "GET"), por defecto es "GET"

Nota: Otros métodos de la petición del HTTP, tales como PUT y DELETE, se pueden también utilizar aquí, pero no son soportados por todos los browsers.

URL (secuencia) - El URL a solicitar.

Devuelve: XMLHttpRequest.

Parámetro:

properties (Map): Un par llave/valor para inicializar la petición.

Ejemplo:

Buscar y ejecutar un archivo JavaScript

```
$.ajax ({
  type: "GET",
  url: "test.js",
  dataType: "script"
 })
```

Ejemplo:

Salvar algún dato al servidor y notificar al usuario una vez que se haya completado

```
 $.ajax ({
  type: "POST",
  url: "some.php",
  data: "name=John&location=Boston",
  success: function (msg) {
    alert (“Data Saved: “+ msg );
  }
 });
```

Ejemplo:

Buscar datos sincrónicos. Bloquear el buscador mientras la petición esté activada. Es mejor que bloquear la interacción con el usuario cuando otro tipo de sincronización es necesaria.

```
var html = $.ajax ({
```

```
  url: "some.php",
  async: false
 }).responseText;
```

Ejemplo:

Enviar un documento XML como dato al servidor. Enviando la opción del processData en falso, automáticamente la conversión de datos a string se detiene.

```
var xmlDocument = [create xml document];
 $.ajax ({
   url: "page.php",
   processData: false,
   data: xmlDocument,
   success: handleResponse
 });
```

$.ajaxSetup (settings)

Conjunto global de elementos para las peticiones de AJAX.

Ver $.ajax para una descripción de todas las opciones disponibles.

Devuelve: indeterminado

Parámetro:

settings (Map): Pares de llave/valor a utilizar para todas las peticiones de AJAX.

Ejemplo:

```
$.ajaxSetup ({
   url: "/xmlhttp/",
   global: false,
   type: "POST"
 } );
 $.ajax ({data: myData});
```

$.ajaxTimeout (time)

Fija el tiempo de espera de todas las peticiones de AJAX a una cantidad de tiempo específica. Esto hará que se amontonen todas de las peticiones de AJAX en el futuro después de cantidad de tiempo

especificada. Esto fija todas las peticiones a cero para deshabilitarlas por defecto. Se puede abortar las peticiones manualmente con el XMLHttpRequest's.
Devuelve: undefined

Parámetro:
time (Number): Números de milisegundos antes de que AJAX solicite tiempo de espera agotado.
Ejemplo:
Hacer todos los tiempos de espera agotados después de 5 segundos
```
$.ajaxTimeout (5000);
```
$.get (url, params, callback)
Buscar una página remota usando una petición del HTTP GET. Esto es una manera fácil de enviar una petición GET al servidor sin tener que utilizar funciones más complejas de $.ajax. Permite que se ejecute una función callback para especificar cual se va a ejecutar cuando la petición se haya completado (solamente si la respuesta ha sido exitosa, en el completamiento del código).
Devuelve: XMLHttpRequest
Parámetros:
url (String): El URL de la página a buscar.
params (Map): (opcional) Par llave/valor que será enviado al servidor.
callback (Function): (opcional) Una función que será ejecutada siempre que los datos hayan sido cargados exitosamente.
Ejemplo:
```
$.get ("test.cgi");
```
Ejemplo:
```
$.get ("test.cgi", {name: "John", time: "2pm”});
```
Ejemplo:
```
$.get ("test.cgi", function (data) {
  alert ("Data Loaded: " + data);
});
```
Ejemplo:
```
$.get ("test.cgi",
```

```
  {name: "John", time: "2pm"},
  function (data) {
    alert ("Data Loaded: " + data);
  }
);
```

$.getIfModified (url, params, callback)

Buscar una página remota utilizando una petición HTTP GET, solo si ha sido modificada hace poco

Devuelve: XMLHttpRequest

Parámetros:

url (String): El URL de la página a buscar.

params (Map): (opcional) Par llave/valor que será enviado al servidor.

callback (Function): (opcional) Una función que será ejecutada siempre que los datos se hayan sido cargados exitosamente.

Ejemplo:

```
$.getIfModified ("test.html");
```

Ejemplo:

```
$.getIfModified ("test.html", {name: "John", time: "2pm"});
```

Ejemplo:

```
$.getIfModified ("test.cgi", function (data) {
  alert ("Data Loaded: " + data);
});
```

Ejemplo:

```
$.getIfModified ("test.cgi",
  {name: "John", time: "2pm"},
  function (data) {
  alert ("Data Loaded: " + data);}
```

$.getScript (url, callback)

Buscar y ejecutar un archivo JavaScript usando una petición HTTP GET.

Devuelve: XMLHttpRequest

Parámetros:

url (String): El URL o de la página que será buscada.
callback (Function): (opcional) Una función.

$.getJSON (url, params, callback)

Buscar datos JSON usando una petición del HTTP GET.
Devuelve: XMLHttpRequest
Parámetros:
url (String): El URL de la página a buscar.
params (Map): (opcional) Par Key/value que será enviado el servidor.
callback (Function): Una función que será ejecutada siempre que los datos hayan sido cargados exitosamente.
Ejemplo:
$.getJSON ("test.js", función que será ejecutada siempre que los datos hayan sido cargados exitosamente.
Ejemplo:
$.getScript ("test.js");
Ejemplo:

```
$.getScript ("test.js", function () {
  alert ("Script loaded and executed.");
});
```

$.post (url, params, callback)

Buscar una página remota usando una petición HTTP POST.
Es tan fácil como enviar una petición POST al servidor sin tener que utilizar funciones más complejas de $ajax. Esto permite que se ejecute una función callback para especificar cual se va a ejecutar cuando la petición se haya completado (solamente si la respuesta ha sido exitosa, en el completamiento del código).
Devuelve: XMLHttpRequest
Parámetros:
url (String): El URL de la página a buscar.
params (Map): (opcional) Pares llave/valor que serán enviados al servidor.

callback (Function): (opcional) Una función que será ejecutada siempre que los datos hayan sido cargados exitosamente.

Ejemplo:

```
$.post ("test.cgi");
```

Ejemplo:

```
$.post ("test.cgi", {name: "John", time: "2pm”});
```

Ejemplo:

```
$.post ("test.cgi", function (data) {
 alert ("Data Loaded: " + data);
});
```

Ejemplo:

```
$.post ("test.cgi",
 {name: "John", time: "2pm”},
 function (data) {
 alert ("Data Loaded: " + data);
}
);
```

ajaxComplete (callback)

Hace que una función se ejecute siempre que se haya completado la petición de AJAX.

El XMLHttpRequest y sus características utilizadas para esta petición se pasan como argumentos al callback.

Devuelve: JQuery

Parámetro:

callback (Function): La función a ser ejecutada.

Ejemplo:

Mostar un mensaje cuando se haya completado los requerimientos de AJAX.

```
 $("#msg").ajaxComplete (function (request, settings) {
   $(this).append ("<li>Request Complete. </li>");
 });
```

ajaxError (callback)

Hace que una función se ejecute siempre que haya fallado la petición de AJAX.

El XMLHttpRequest y sus características utilizadas para esta petición se pasan como argumentos al callback. Un tercer argumento se le pasa cuando ocurre una excepción mientras se procesa la petición, un objeto.

Devuelve: JQuery

Parámetro:

callback (Function): La función a ejecutar.

Ejemplo:

Mostrar un mensaje cuando la petición ha fallado.

```
$("#msg").ajaxError (function (request, settings) {
  $(this).append ("<li>Error requesting page” + settings.url + "</li>");
});
```

ajaxSend (callback)

Hace que una función se ejecute antes de que se haya enviado la petición de AJAX.

El XMLHttpRequest y sus características utilizadas para esta petición se pasan como argumentos al callback.

Devuelve: JQuery

Parámetro:

callback (Function): La función a ejecutar.

Ejemplo:

Muestra un mensaje antes de que sea enviada la petición AJAX.

```
$("#msg").ajaxSend (function (request, settings) {
  $(this).append ("<li>Starting request at” + settings.url + "</li>");
});
```

ajaxStart (callback)

Hace que una función se ejecute siempre que la petición haya comenzado y no este activa aún.

Devuelve: JQuery

Parámetro:

callback (Function): La función a ejecutar.

Ejemplo:

Mostrar un mensaje siempre que la petición haya comenzado (y no este activa aún).

```
 $("#loading").ajaxStart (function () {
   $(this).show ();
 });
```

ajaxStop (callback)

Hace que una función se ejecute siempre que todas las peticiones AJAX hayan terminado.

Devuelve: JQuery

Parámetro:

callback (Function): La función a ejecutar.

Ejemplo:

Oculta el mensaje después que se hayan detenido las peticiones AJAX.

```
 $("#loading").ajaxStop (function () {
   $(this).hide ();
 });
```

ajaxSuccess (callback)

Hace que una función se ejecute siempre que todas las peticiones AJAX hayan terminado exitosamente. El XMLHttpRequest y sus características utilizadas para esta petición se pasan como argumentos al callback.

Devuelve: JQuery

Parámetros:

callback (Function): La función a ejecutar.

Ejemplo:

Muestra un mensaje cuando la petición AJAX se ha completado exitosamente.

```
 $("#msg").ajaxSuccess (function (request, settings) {
   $(this).append ("<li>Successful Request! </li>");
 });
```

load (url, params, callback)

Busca HTML de un archivo remoto y lo introduce en el DOM

Devuelve: JQuery

Parámetros:

url (String): El URL del archivo HTML a buscar.

params (Object): (opcional) Un conjunto de pares de llave/valor que serán enviados al servidor.

callback (Function): (opcional) Una función que será ejecutada siempre que los datos hayan sido cargados (parámetros: responseText, status y response itself).

Ejemplo:

```
$("#feeds").load ("feeds.html");
```

Antes:

```
<div id="feeds"></div>
```

Resultado:

```
<div id="feeds"><b>45</b> feeds found. </div>
```

Ejemplo:

Lo mismo de antes, pero con parámetros adicionales y un callback que se ejecuta cuando los datos son cargados.

```
$("#feeds").load ("feeds.html",
  {limit: 25},
  function () {alert ("The last 25 entries in the feed have been loaded") ;}
);
```

2.14 Pluggins

Pluggins de JQuery que han sido creados por usuarios de esta librería, que se pueden incorporar después de haber instalado JQuery.

Thick box: Para contenido externo e imágenes.

Tabs: Rápido y fácil, construye una interfaz de navegación para un sitio Web accesible y sin obstrucciones. Provee además animaciones predefinidas y personalizadas en la selección de tab, así como callbakcs.

Accessible News Slider: Es un componente que permite visualizar los encabezados y las fotos, el cual fue construido para que sea accesible acorde WCAG 1.0.

AnimateClass, AnimateStyle: Elementos animados de una clase de CSS a otra como se describe en las tablas de estilos o usando las expresiones básicas de CSS.

Auto: Provee mecanismos para auto-tab el próximo campo, auto-image toggle, auto-select, etc.
Autohelp: Muestra los textos de ayuda específicos en cada forma, usando nada más que el título de los elementos de los diferentes campos.
blockUI: Una alternativa para sincronizar Ajax. Bloquea la interfaz del usuario sin bloquear el buscador.
FontSizer: Un plugins diseñado para redimensionar el tamaño del fondo del sitio.

2.15 Drupal y JQuery

Los sistemas de gestión de contenidos se han caracterizado en los últimos tiempos por presentarse en versiones con los requerimientos mínimos para su funcionamiento y cualquier otra funcionalidad que se requiera se añade a través de extensiones o plugins. Drupal no es una excepción a esto, por eso a un usuario común le puede parecer después de una instalación básica de este CMS que carece de muchos de los elementos que necesitaría.
La mayoría de estos elementos se encuentran disponibles en extensiones que hacen uso de JavaScript. Desde su creación Drupal cuenta con una serie de funciones JavaScript agrupadas en un archivo que han ido consolidándose en una especie de biblioteca, pero que brinda funcionalidades muy básicas. A través del uso de JQuery como librería de JavaScript se puede ampliar el espectro de las extensiones a usar en Drupal y así mejorar muchas de sus funcionalidades.
Integrar Drupal y JQuery es tan sencillo como poner en uso el archivo jquery.js en aquellas páginas donde deseemos hacer uso de las funcionalidades de esta librería. A continuación veremos algunas de las funcionalidades que podemos añadirle a Drupal haciendo uso de JQuery.

2.15.1 Editor de contenidos.

La instalación por defecto de Drupal carece de un editor WYSIWYG para el tratamiento de los contenidos a publicar. Existen varios módulos que permiten la inclusión de un editor como TinyMCE o FCK Editor pero a través de JQuery nos llega un editor muy sencillo y que es fácilmente escalable.
Su instalación es muy simple, veamos un ejemplo:

```
<! DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
 "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html>
<head>
<title>WYMeditor</title>
```

```
<link rel="stylesheet" type="text/css" media="screen" href="wymeditor/skins/default/screen.css" />
<script type="text/javascript" src="jquery/jquery.js"></script>
<script type="text/javascript" src="wymeditor/jquery.wymeditor.js"></script>
<script type="text/javascript">
var $j = jquery.noConflict ();
$j (function () {
    $j (".wymeditor").wymeditor ();
});
</script>
</head>
<body>
<form method="post" action="">
<textarea class="wymeditor"></textarea>
<input type="submit" class="wymupdate" />
</form>
</body>
</html>
```

Este código coloca un editor en cada etiqueta textarea que tengamos en nuestra página. Si lo aplicamos en la página de edición de Drupal obtendríamos un editor sencillo para el texto a publicar con una apariencia similar a la de la figura 1:

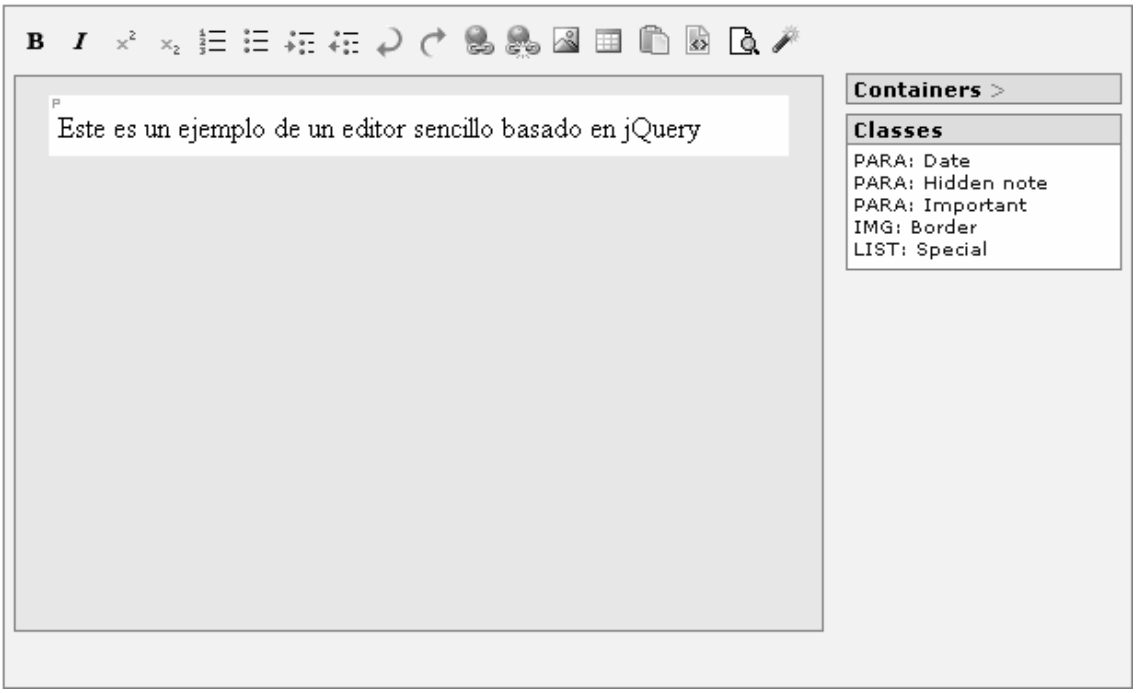

Figura 1. Editor WYMeditor.

WYMeditor está basado en XHTML y CSS, esto le da una propiedad especial de ignorar cualquier tipo de formato para los textos que provienen de páginas o documentos externos y que en la mayoría de los editores heredan propiedades de tamaño o color del texto.

Tiene soporte para la inserción de imágenes, tablas y vínculos; así como la creación de plantillas para adaptar su diseño al entorno en el cual se coloque.

2.15.2 Ajax.

El uso de Ajax se ha popularizado en los últimos tiempos y no es solo por seguir una moda. Esta variante de implementación de interfaces permite la interacción con el servidor sin necesidad de recargar la página completa, haciendo de esta forma más rápida y natural el intercambio con nuestra aplicación, mas allá de los efectos visuales que se puedan lograr haciendo uso de la misma.

La librería de Drupal provee algunas funciones básicas para la explotación de funcionalidades Ajax, pero como pudimos apreciar JQuery tiene una variedad de funciones para el tratamiento de Ajax que hacen que su uso pueda ser mucho más extensivo.

Veamos un ejemplo sencillo de la suma de dos números, si tenemos en la página un vínculo con ID “botón _ sumar”, un DIV para mostrar el resultado con ID “resultado” y un script en php que nos devuelva la suma de dos números que recibirá a través de term_1 y term_2 el código sería tan sencillo como este:

```
$(document).ready (
	function () {
		//capturamos el clic de vínculo
		$('#boton_sumar').click (function () {_sumar_dos_ajax () ;});
	}
);
function _sumar_dos_ajax () {
	// obtener los resultados
	var val1 = $('#edit-term-1').val ();
	var val2 = $('#edit-term-2').val ();
	// cargar los contenidos en el div
	$('#resultado').load ('/drupal/ajax/add_two_numbers.php', {term_1: val1, term_2: val2});
}
```

En muchas ocasiones necesitamos mostrar opciones a partir de una selección realizada por el usuario, veamos una solución a un problema similar:

```
$(document).ready (
	function () {
		$('#provincia').change (
			function ()  {
				$('#opciones').load ('/drupal/ajax/get_dropdown_options', {prov:
$('#provincia').val ()});
				}
			);
		}
);
```

En este caso vemos la actualización del contenido del DIV “opciones” a partir del resultado que nos brindaría el script get_dropdown_options al recibir la provincia seleccionada en una lista de selección (SELECT).

Son múltiples las soluciones que se pueden dar con Ajax sin explotar en demasía este recurso ya que en ocasiones debemos respetar que el usuario aun espera navegación clásica por parte de nuestra página y el uso de los botones de ir atrás, hacia delante o refrescar no tienen el efecto deseado por el usuario, en la mayoría de las ocasiones, cuando estamos usando Ajax.

Ventanas emergentes no invasivas, ThickBox.

Uno de los grandes problemas en el desarrollo de una interfaz Web era el uso excesivo de ventanas emergentes o PopUps, los cuales en la mayoría de las ocasiones producían una reacción adversa por parte del usuario ya que esto originaba o la generación de un nuevo proceso en la máquina del cliente o el aumento del consumo de recursos por parte del proceso del navegador.

La mayoría de los navegadores hoy en día incluyen bloqueadores para estas ventanas emergentes, lo que en la mayoría de los casos es útil, pero en otras ocasiones evita que el usuario reciba informaciones o realice acciones vitales para nuestra aplicación.

ThickBox es un plugin de JQuery que permite mostrar contenidos en la misma página, incluso cargados a partir del resultado de una aplicación en el servidor (a través de Ajax).

Drupal lo ha incluido como un módulo, fundamentalmente para brindar la posibilidad de mostrar al detalle imágenes que por encontrarse dentro del contenido no se pueden apreciar correctamente.

Figura 2. Uso de ThickBox para mostrar una imagen en detalle.

2.15.3 Drupal 5.

La versión actual de Drupal ya incluye el uso de JQuery e incluso reescribió su librería interna para hacer uso de las funcionalidades de JQuery.

Esto le permite a Drupal poner en práctica las facilidades para el trabajo con el DOM de esta librería de JavaScript así como convertir fácilmente en módulos para el CMS gran cantidad de las extensiones desarrolladas para la librería.

Como se puede apreciar el aporte de JQuery a Drupal es importante sobre todo para el desarrollo de elementos de interfaz Web que hagan de nuestro sitio o portal una herramienta de fácil uso por parte de usuario final, y que sin duda alguna provee todos los elementos necesarios para que los desarrolladores creen nuevas extensiones que cumplan con los requerimientos particulares de un diseño en específico.

CONCLUSIONES GENERALES

La concepción, diseño e implantación del CMS Drupal constituye un serio aporte en el desarrollo y creación de Portales Web en nuestro país. Mediante su exitosa introducción en las condiciones concretas de la Universidad de las Ciencias Informáticas se han creado condiciones materiales y subjetivas que garantizan el futuro perfeccionamiento en la creación de Portales Web. Durante el desarrollo de la presente investigación se han empleado tecnologías novedosas y de gran alcance en la Informática y en el desarrollo de nuevos Portales. Esto implica que necesariamente se utilicen nuevas herramientas y tecnologías, de las cuales se necesita información. Es debido a esto que para la realización y desarrollo de este trabajo se buscó, investigó y agrupó, un cúmulo de información referente a distintos materiales de soporte (como es el caso de las Librerías) para su posterior utilización por parte de la entidad en el desarrollo de Portales; y que además se adapten a las condiciones concretas de nuestro país. Es por lo que se realiza una extensa investigación de la librería JQuery con el objetivo general definido de: Realizar una propuesta de librería(s) de JavaScript que se encuentre(n) orientada(s) al cliente, y que sea(n) posible(s) de utilizar en Drupal con la finalidad de aumentar sus servicios y prestaciones en la creación de proyectos productivos.

Por lo cual se pueden establecer las siguientes conclusiones:

Se llevo a cabo un proceso investigativo extenso con la finalidad de proporcionarle al usuario la mejor opción para el desarrollo ágil de su trabajo.

La librería seleccionada cumple con los requisitos mínimos que exigía el proyecto para su posterior utilización como son: flexibilidad, usabilidad, facilidad de uso y adaptación a cambios que se pudieran producir en el futuro, así como su posterior actualización.

RECOMENDACIONES

Dada la sencillez de JQuery, su potencia para el trabajo con el DOM, lo ligera, adaptable y escalable que es, se recomienda su uso como librería de JavaScript no solo integrada a Drupal sino incluso a otras herramientas y sistemas que se desarrollen o usen dentro de la universidad.

Que se siga investigando sobre el tema puesto que es novedoso y sobre todo útil en la agilización del trabajo en la Web.

Que el presente trabajo investigativo sea publicado dada la necesidad de esta información que hay el la Universidad.

BIBLIOGRAFÍA

ALVAREZ, S. JQuery vs. Mootools. *http://xergio.net/escritos/pagina-36.html*, 2006.

BOURDETTE, J. Java partiendo desde cero. *http://www.wikilearning.com/empecemos_de_una_vez-wkccp-3830-7.htm*, 2004.

BUESING, G. JQuery, Ajax + Rails. http://mad.ly/2007/05/17/jquery-ajax-rails, 2007.

CICALI, C. Prototype & JQuery Benchmarked *http://claudio.cicali.org/article/100/prototype-and-jquery-benchmarked*, 2006.

CO, A. K. JQuery otra librería para Ajax. *http://www.anieto2k.com/2006/01/21/jquery-otra-libreria-para-ajax*, 2006.

FIREBOY. Cinco framework Ajax en JavaScript *http://www.caudalweb.com/blog/?p=39*, 2007.

GALVEZ, C. C. Efectos en JQuery. *http://blog.scriptia.net/tags/efectos%7Cjquery/page/2/*, 2006.

---. Introducción a JQuery. *http://dizque.lacalabaza.net/sotanos/2006/05/introduccion-a-jquery*, 2006.

GÁLVEZ, C. C. Eventos en JQuery. *http://blog.scriptia.net/articulos/2006/12/eventos-en-jquery.html*, 2006.

INC, Y. Yahoo! User Interface Library. *http://developer.yahoo.com/yui/index.html*, 2007.

JAIRO.SERRANO. Blog: un blog para cada usuario. *http://www.drupal.org.es/node/17*, 2005a.

---. Comment: Sistema de comentarios. *http://www.drupal.org.es/manuales/manual_del_administrador/sistema_de_comentarios*, 2005b.

---. Configuraciones Adicionales de los Comentarios. *http://www.drupal.org.es/manuales/manual_del_administrador/sistema_de_comentarios/configuraciones_adicionales_de_los_comentari*, 2005c.

JAVI. Primeros pasos con JQuery. *http://javi.interzonas.info/entrada.php?op=ViewArticle&articleId=691&blogId=13*, 2006.

JQUERY, J. R. A. T. Downloading JQuery. *http://docs.jquery.com/Downloading_jQuery*, 2007.

LINEADECODIGO. Hola Mundo en DOJO. *http://lineadecodigo.com/2007/01/01/hola-mundo-en-dojo*, 2007.

MARTINEZ, I. Editores Web Quanta Plus. *http://indira-informatica.blogspot.com/2007/04/editores-web-quanta-plus.html*, 2007.

MODERNMETHOD. Welcome to Sajax. *http://www.modernmethod.com/sajax*, 2005.

MOESKAU, B. Introducción a Ext. *http://extjs.com/tutorial/introducci-n-ext*, 2007.

MOLINA, I. M. Introducción a AJAX. *www.rediris.es/gt/middleware/coord/gt2006/IntroduccionAJAX_v1.0.ppt*, 2006.

NEDJO. Jtools. *http://drupal.org/project/jstools*, 2006.

PROIETTI, V. Mootools. *http://mootools.net*, 2007.

REYERO, J. A. Características de Drupal. *http://www.drupal.org.es/caracteristicas*, 2005.

SKINNER, J. Simplify Ajax development with JQuery. *http://www-128.ibm.com/developerworks/library/x-ajaxjquery.html*, 2007.

SOURCEFORGE.NET, I. Biblioteca javascript Jmol.js. *http://jmol.sourceforge.net/jslibrary/index.es.html*, 2007.

SWEDBERG, K. How to Get Anything You Want. *http://docs.jquery.com/Tutorials:How_to_Get_Anything_You_Want*, 2006.

---. Introducing $(document).ready (). *http://www.learningjquery.com/2006/09/introducing-document-ready*, 2006.

---. JQuery 1.1.2 Released. *http://www.learningjquery.com*, 2007.

---. Multiple $(document).ready (). *http://www.learningjquery.com/2006/09/multiple-document-ready*, 2006.

TEAM, D.: *http://drupal.org*, 2007.

---. Dojo Javascript Toolkit. *http://www.dojotoolkit.org*, 2007.

---. History of JQuery in Drupal. *http://drupal.org/node/88980*, 2007.

TEAM, J. R. A. T. J. Ajax. *http://docs.jquery.com/Ajax*, 2007.

---. Attributes. *http://docs.jquery.com/DOM/Attributes*, 2007.

---. Core. *http://docs.jquery.com/Core*, 2007.

---. Css. *http://docs.jquery.com/API/1.1.2/CSS*, 2007.

---. Effects. *http://docs.jquery.com/Effects*, 2007.

---. Events. *http://www.docs.jquery.com/API/1.1.2/Events*, 2007.

---. How JQuery Works. *http://docs.jquery.com/Tutorials:How_jQuery_Works*, 2007.

---. JavaScript Utilities. *http://docs.jquery.com/JavaScript*, 2007.

---. Manipulation. *http://docs.jquery.com/API/1.1.2/DOM/Manipulation*, 2007.

---. Pluggins. *http://docs.jquery.com/Plugins*, 2007.

---. Selectors. *http://docs.jquery.com/Selectors*, 2007.

---. Sites Using JQuery. *http://docs.jquery.com/Sites_Using_jQuery*, 2007.

---. Traversing. *http://docs.jquery.com/DOM/Traversing*, 2007.

---. Tutorials. *http://docs.jquery.com/Tutorials*, 2007.

TEAM, J. R. A. T. J. JQuery. *http://jquery.com*, 2007.

TEAM, P. C. Prototype Javascript Framework. *http://prototype.conio.net*, 2006 - 2007.

TECHKNOW, J. JQuery Library. *http://www.juixe.com/techknow/index.php/2006/01/29/jquery-library/*, 2007.

UNIJIMPE. Ajax Auto Suggest. *http://blog.unijimpe.net/category/ajax*, 2007.

VOLDER, K. D.: *http://jquery.cs.ubc.ca*, 2006.

WIKIMEDIA FOUNDATION, I. Adobe Dreamweaver. *http://es.wikipedia.org/wiki/Adobe_Dreamweaver*, 2007.

WIKIMEDIA FOUNDATION, I. Dojo toolkit. *http://es.wikipedia.org/wiki/Dojo_toolkit*, 2007a.

---. Drupal. *http://es.wikipedia.org/wiki/Drupal*, 2007b.

---. Herramienta de Autor. *http://es.wikipedia.org/wiki/Herramienta_de_Autor*, 2006.

---. Inspector Dom. *http://es.wikipedia.org/wiki/Inspector_DOM*, 2007c.

---. Quanta Plus. *http://es.wikipedia.org/wiki/Quanta_Plus*, 2007d.

---. Sajax. *http://es.wikipedia.org/wiki/Sajax*, 2007e.

---. Xajax. *http://es.wikipedia.org/wiki/Xajax*, 2007f.

WILSON, J. W. J. M. Introducing xajax. *http://xajax.sourceforge.net*, 2005.

GLOSARIO

Gecko: es un motor de renderizado libre diseñado para soportar los estándares abiertos de Internet. Es también una plataforma para aplicaciones multiplataforma, es decir: permite ejecutar aplicaciones sobre su motor que se sirvan de tecnologías como XUL, XBL, PNG, HTTP, POP3, SMTP, RDS, CSS en virtualmente cualquier sistema operativo. El Motor de renderizado de un navegador Web toma el contenido (HTML, XML, imágenes, etc.) y la información de formato (CSS, etc.) y crea una representación visual de una página Web. Todos los navegadores Web incluyen necesariamente algún tipo de motor de renderizado.

El Document Object Model: (en español 'Modelo de Objetos de Documento'), frecuentemente abreviado DOM, es una forma de representar los elementos de un documento estructurado (tal como una página Web HTML o un documento XML) como objetos que tienen sus propios métodos y propiedades. El responsable del DOM es el World Wide Web Consortium (W3C).

Widgets: Es una pequeña aplicación o programa, usualmente presentado en archivos o ficheros pequeños que son ejecutados por un motor de widgets o Widget Engine. Entre sus objetivos están los de dar fácil acceso a funciones frecuentemente usadas y proveer de información visual. Sin embargo los widgets pueden hacer todo lo que la imaginación desee e interactuar con servicios e información distribuida en Internet; pueden ser vistosos relojes en pantalla, notas, calculadoras, calendarios, agendas, juegos, ventanas con información del clima en su ciudad, etcétera.

MIME: (Acrónimo de Multi-purpose Internet Mail Extensions, Extensiones de Correo Internet Multipropósito), son una serie de convenciones o especificaciones dirigidas a que se puedan intercambiar a través de Internet todo tipo de archivos (texto, audio, vídeo, etc.) de forma transparente para el usuario. Una parte importante del MIME está dedicada a mejorar las posibilidades de transferencia de texto en distintos idiomas y alfabetos.

BBS: Un BBS o Bulletin Board System (Sistema de Tablón de Anuncios) es un software para redes de computadoras que permite a los usuarios conectarse al sistema (a través de Internet o a través de una línea telefónica) y utilizando un programa terminal (o telnet si es a través de Internet), realizar funciones tales como descargar software y datos, leer noticias, intercambiar mensajes con otros usuarios, disfrutar de juegos en línea, leer los boletines, etc.

WYSIWYG: Es el acrónimo de What You See Is What You Get (en español, "lo que ves es lo que obtienes"). Se aplica a los procesadores de texto y otros editores de texto con formato (como los editores de HTML) que permiten escribir un documento viendo directamente el resultado final, frecuentemente el resultado impreso. Se dice en contraposición a otros procesadores de texto, hoy en día poco frecuentes, en los que se escribía sobre una vista que no mostraba el formato del texto, hasta la impresión del documento. En el caso de editores de HTML este concepto se aplica a los que permiten escribir la página sobre una vista preliminar similar a la de un procesador de textos, ocupándose en este caso el programa de generar el código fuente en HTML.