Universidad Nacional de Misiones

Facultad de Ciencias Exactas, Químicas y Naturales

Carrera

Licenciatura en Sistemas de Información

Cátedra

Diseño de Aplicaciones en la Web
Titular: Lic. Rubén Urquijo

Nombre del Trabajo:

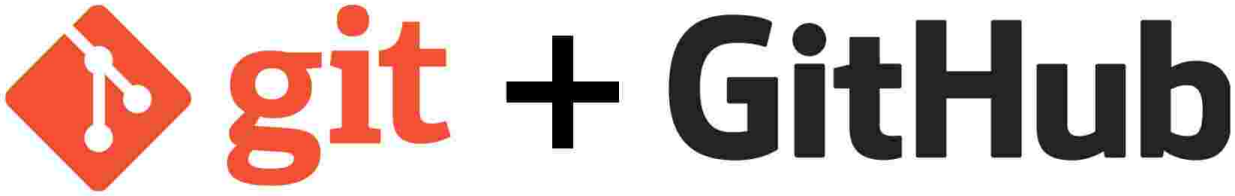
Trabajo de Investigación: Mootools vs JQuery
v0.1

Alumno:
Ulises C. Ramirez
Legajo Universitario: LS00704

Índice de Contenidos

Información Útil

Este documento como así también los demás que componen a esta monografía se encuentran versionados en el siguiente repositorio GitHub



Seguimiento de Entregas

Versión	Contenido	Modificaciones con respecto a versión anterior
0.1	1. Introduccion, JavaScript, Librerias JavaScript, Comprarativa: MooTools vs JQuery, Conclusión. 2. Anexo 1: Conceptos Generales 3. Anexo 2: Software de ejemplo	---

1. Introducción

1.1 Preliminares

1.1.1 Arquitectura Cliente-Servidor (C/S)

La **arquitectura cliente-servidor** es un modelo de aplicación distribuida en el que las tareas se reparten entre los proveedores de recursos o servicios, llamados servidores, y los demandantes, llamados clientes. Un cliente realiza peticiones a otro programa, el servidor, quien le da respuesta. Esta idea también se puede aplicar a programas que se ejecutan sobre una sola computadora.

La entidad en esta relación que recibe las peticiones del cliente, llámese servidor, correrá un software especial que se encargará de darle respuesta a las peticiones que se realicen desde los distintos dispositivos que se conecten a éste con el rol de cliente.

Este software puede ser destinado a dar respuesta a solicitudes de distinta índole, estas varían dependiendo de las necesidades que se tengan, ya que estos servidores pueden estar especializados para el manejo de un sistema de correo, un sistema de archivos, un sistema de directorios, un sistema web, un sistema de streaming, incluso uno que se dedique a virtualización, que permita tener todos estos servicios en diferentes maquinas virtuales, en una sola computadora física.

En este documento se va a tratar el tema haciendo referencia específica a varios aspectos y características los necesarios para el manejo de **sistemas web**.

1.1.2 Servidor web

Es un programa informático que se encarga de procesar un programa del lado del servidor, realizando conexiones síncronas o asíncronas con el cliente, que esta realizando peticiones mediante un protocolo HTTP o HTTPS. Estos dan respuesta mediante un documento con un formato especial que es interpretado en el cliente mediante un software también especializado, a los cuales se les da la denominación de **Navegadores Web.**

Podemos nombrar aquí algunos productos de software que se encuentran en el mercado que se encargan de brindar este servicio en ordenadores que cumplen el rol de servidor.

1.1.3 HTML

Este es un lenguaje que tiene la capacidad de crear y procesar **etiquetas**, las cuales se hacen útiles a la hora de la creación de páginas web. Define una estructura básica y un código (denominado código HTML) para la definición de contenido de las paginas mencionadas anteriormente, como texto y contenidos multimedia. **Es el estándar que se ha impuesto en la visualización de páginas en internet y es el que todos los navegadores actuales han adoptado**.

El objetivo de este código HTML es hacer que el documento que lo contenga, **contenga unicamente texto** por lo que **recae** en el navegador web la tarea de unir todos los elementos y visualizar la página final.

1.1.4 DOM

El Modelo de Objeto del Documento (por sus siglas en ingles, DOM: *Document Object Model*) es una API[1] de lenguaje independiente y multiplataforma que trata un documento HTML, XHTML, XML como una estructura de árbol en la cual cada nodo es un objeto que representa una parte del documento. Estos objetos pueden ser manipulados mediante programación y cualquier cambio que se realice en estos documentos será reflejado en la forma en el que el documento es mostrado.

1.1.4.1 Historia

El nacimiento del DOM se da en el contexto de una ‘guerra de navegadores’ que tuvo lugar en la década de 1990, en donde, había una disputa entre Netscape Navigator y Microsoft Internet Explorer como así también JavaScript y JScript estos últimos siendo los primeros lenguajes de ‘scripting’ a ser ampliamente implementados en los motores para los Layouts de los Navegadores web.

Dado al hecho que JavaScript es una marca registrada de Oracle Corporation (es usada con una licencia por Netscape Communications y la Fundación Mozilla), Microsoft dió como nombre a su implementación del ‘dialecto’ de ECMAScript, **JScript**, el cual fue adoptado a partir de la versión 3.0 de MS Internet Explorer.

JavaScript y JScript permitían a los desarrolladores crear paginas web con interactividad del lado del cliente.

Dada a estas diferentes implementaciones de ‘dialectos’ que eran incompatibles entre si, el **W3C**(World Wide Web Consortium) diseño el estándar que nos ocupa en este apartado, el **DOM**. Aunque cabe destacar que las ‘facilidades’ que brindaban JavaScript y JScript en sus etapas tempranas al momento de detectar eventos generados por usuarios y la modificación del HTML, fue conocido eventualmente como *DOM Nivel 0*.

1 Aplication Programming Interface, puede encontrar más información en el Anexo 1: Conceptos generales

1.1.5 Motores de navegador web

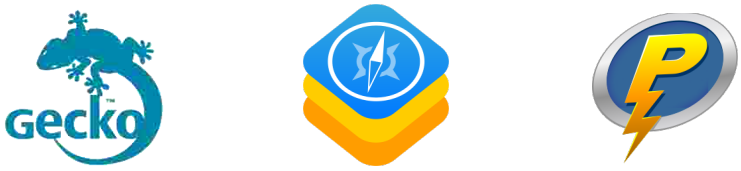
1.1.5.1 Introducción

Los primeros navegadores fueron monolíticos, usaban varias técnicas que tomaban del procesamiento de texto, tales como las expresiones regulares[2] para poder así interpretar el código HTML que se reciba y así poder transformarlo en una representación visual, mas adelante adoptaron un acercamiento mas modular en donde se separa la lógica para la representación del código HTML recibido y el navegador en si mismo.

1.1.5.2 Funcionamiento conceptual

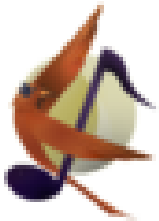
La lógica que permite la representación del código HTML recibido se puede generalizar en dos partes principales:

1. El motor (Engine): es el que hace la mayor parte del trabajo, esencialmente toma la URL y un conjunto de coordenadas de rectángulos de la ventana como argumentos. Luego obtiene el documento que corresponde a la URL y realiza la representación gráfica del mismo en los rectángulos dados. También se encarga del manejo de enlaces, formularios, cookies, scripting del lado del cliente, carga de plug-in's entre otras cosas. Ejemplos:



Software 1: Gecko; WebKit; Presto, mas info en Anexo 2: Software de Ejemplo

2. La aplicación host: esta provee de la barra de menús, la barra de direcciones la barra de estado el administrador de marcadores, el historial y las distintas preferencias en cuanto a algunos criterios que se pueden parametrizar desde el cliente. Esta aplicación, incorpora el motor y sirve como una interfaz entre el usuario, el motor, y el sistema operativo. Ejemplos:



Software 2: Nightingale, mas info en el Anexo 2: Software de Ejemplo

Software 3: Picasa. mas info Anexo 2: Software de Ejemplo

Software 4: DevHelp, mas info en Anexo 2: Software de Ejemplo

Software 5: Instantbird, mas info en Anexo 2: Software de Ejemplo

2 Información sobre las Expresiones regulares en el Anexo 1: Conceptos generales

1.1.5.3 Resumen

A veces denominado motor de layout web (*web layout engine*) o motor de renderizado web (*web rendering engine*), es un programa de computadora que ayuda a renderizar (Sinónimos: graficar, dibujar, representar) **contenido etiquetado** (contenido generado con lenguaje de etiquetas como puede ser HTML o XML) e información de formateo (por ejemplo CSS). Un motor de layout web es un componente típico en los navegadores, clientes de email o otras aplicaciones que requieren mostrar y editar el contenido de una pagina web

1.1.6 Clientes (Navegadores Web)

Para el renderizado de un documento tal como una página HTML, la mayoría de los navegadores, en la actualidad, utilizan un modelo de estructura interna, que vendrá dada por el motor que utilicen similar al DOM. Los nodos de cada documento están organizados en una estructura de árbol con el elemento mas 'Alto' denominado el *Objeto de Documento* . Cuando una pagina HTML es representada de manera visual en un navegador, éste descarga el HTML en una memoria local y automáticamente lo interpreta para así poder mostrar el contenido en la pantalla, **el DOM es también la forma en la que JavaScript transmite el estado del navegador en páginas HTML.**

Con el objeto de modelo, JavaScript es totalmente capaz de crear HTML dinámico:

- Puede agregar, cambiar, o quitar todos los elementos HTML y atributos en la página.
- Puede cambiar todo los estilos CSS en una página.
- Puede reaccionar a todos los eventos que existen en la página
- puede crear nuevos eventos dentro de la página.

Las funciones JavaScript que permiten a los Navegadores tener esta funcionalidad se encuentran **embebidas dentro del motor que tenga implementado el navegador** utilizado.

Producto de la discrepancia que surge dentro de los diferentes motores con respecto a la implementación de un dialecto para ECMAScript surgen también problemas de compatibilidad con el JavaScript dentro de diferentes navegadores, es decir, el contenido puede verse bien en algunos navegadores, pero a la hora tratar de verlo en otro navegador «que puede utilizar un motor con diferente implementaciones de la API para JavaScript» el resultado puede ser totalmente diferente.

2. JavaScript (JS)

JavaScript es un lenguaje de programación funcional (es decir que maneja las funciones como los objetos de mayor orden que pueden ser pasados a través de variables como cualquier otro objeto - strings o números), que brinda objetos nativos como Strings, Numbers, Functions, Arrays, Dates, Regular Expressions, y más. JavaScript también te brinda un modelo de herencia - una especie de modelo llamado prototypal inheritance. Estos bloques de construcción y el concepto de herencia son conceptos muy importantes para la base de cualquier lenguaje de programación y nada tienen que ver con los browsers. Esto da la pauta de que se puede escribir cualquier cosa en JavaScript. Ajedrez, edición de fotos, un web server, cualquier cosa. Lo que sucede es que el 99% de todo el JavaScript que anda por ahí se ejecuta en browsers y eso es como lo pensamos que es. El lenguaje de programación para los browsers.

Entendiendo que el browser, el DOM, es solo donde acostumbramos a usar JS la mayoria del tiempo pero saber que JS es actualmente un muy robusto y expresivo lenguaje de programación

2.1 Características a destacar en JS

Herencia: JavaScript tiene un modelo de herencia que es bastante poco común en lenguajes de programación. En vez de clases que son definidas y pueden tener subclases éste pasa los comportamientos a través de *herencia prototipada (prototypal inheritance)*. Esto significa que los objetos heredan directamente de otros objetos. Si se referencia una propiedad de un objeto que hereda de otro objeto, el lenguaje inspecciona el objeto hijo por esa propiedad y, si no la encuentra, busca a ella en el padre. De esta manera es como trabaja un array.

Ejemplo 1:

```
[1,2,3].forEach(function(item) { alert(item) }); //esto hace alert 1 despues  2 despues 3
```

---

el método **`forEach`** no es una propiedad del array que se declaró (*[1,2,3]*), éste es una propiedad del prototipo para todos los Arrays. Cuando se referencia a este método el lenguaje busca un método llamado ***`forEach`*** en el array declarado, y, si no lo encuentra, busca entonces en el prototipo de todos los arrays. Esto significa que el método ***`forEach`*** no está en memoria por cada array existente; éste está solo en la memoria del prototipo de arrays. Esto es una característica de JS, una de las cosas que lo hace muy eficiente y bastante potente.

Autoreferencia: Esta característica permite a los objetos que heredan de otros objetos referirse a si mismos.

Ejemplo 2:

```
var Auto = {
    caracteristicas: ['color rojo', 'comodo', 'rapido'],
    log: function(message) {
        console.log(message);
    },
    verCaracteristicas: function() {
        this.caracteristicas.each(function(caract) {//queremos que "this" esté apuntando a Auto...
            this.log('Este auto es ' + caract); // llamamos al metodo log definido arriba
        }, this); //entonces pasamos "this" (que es Auto) al método Array.each
    }
};
ninja.logInventory();
//Este auto es color rojo
//Este auto es comodo
//Este auto es rapido
```

Estos son solo algunos ejemplos de la expresividad que JavaScript tiene para ofrecer, herencia y autoreferencia, ademas de eficientes propiedades de prototipos. Lo malo es que JS básico, no hace accesibles estas poderosas cosas, es decir, hay que trabajar mucho para poder obtener esto, y aquí es donde las L*ibrerías* surgen, aunque, con sus respectivas diferencias, como puede ser el público al cual va dirigido y las cosas que se busca logre la herramienta. A continuación se analizará a cada herramienta y luego las pondremos en comparación.

3. Librerías[3] JavaScript

A raíz de lo expuesto en el punto **1.1.6**, con el tiempo surgieron proyectos que buscaban mitigar las consecuencias de la implementación del dialecto ECMAScript de los diferentes navegadores, lo cual hasta ese entonces ocupaba una parte de las funciones de las cuales se tenían que preocupar los programadores.

3.1 MooTools

> *"MooTools es un compacto, modular, framework Orientado a Objeto JavaScript diseñado para el programador JavaScript intermedio a avanzado. Este permite escribir un código potente, flexible, y cross-browser[4] con su elegante, bien documentada y coherente API."* – Página oficial MooTools

3.1.1 Resumen

MooTools es un **un Framework**, que proveen clases de programación orientada a objetos en JavaScript, para realizar una amplia gama de funcionalidades en páginas web, como por ejemplo, el trabajo con elementos del DOM, efectos diversos y Ajax. Con MooTools, se puede programar todo tipo de scripts estos se ejecutaran en el cliente rápidamente como cualquier código JavaScript, a diferencia de que en su interior MooTools ya tiene en cuenta la cuestión de los diferentes motores de navegadores web y lo soluciona, esto permite liberar al programador de la preocupación de conocer desde que navegador se accederá al contenido que esta creando y centrarse unicamente en la funcionalidad que desea ofrecer. MooTools está especialmente indicado para programar scripts complejos, que nos costaría mucho más trabajo de realizar si partiésemos de cero.

Una porción significante de la biblioteca principal esta dedicada mejorar Function, String, Array, Number, Element y otros prototipos y otra de las características que destacan a este Framework es que ofrece una función llamada *Class*. Esta última busca hacer el modelo de herencia prototipada de JavaScript más sencillo para de acceder, y así, poder tomar ventaja de ello.

3.2 JQuery

> *"jQuery es una Biblioteca de JavaScript rápida y concisa que simplifica el recorrido, manejo de eventos, animación, y interacciones Ajax en el documento HTML para un rápido desarrollo web. jQuery está diseñado para cambiar la manera que tu escribes JavaScript."* – Página oficial JQuery

3.2.1 Resumen

Al igual que MooTools, JQuery es una librería basada en JS (Aunque orientada hacia otro publico, ya que este es considerado como un **toolkit**), que dentro del desarrollo web nos brinda ya una funcionalidad mas digerida, y esto permite implementar por ejemplo funciones que son comunes como pueden ser las consultas mediante Ajax, de forma mucho mas rápida dado a que esta ya esta escrita, sumado a eso, se tiene que el la librería nos asegura la compatibilidad internavegador.

---

3 Librerías de Software, puede encontrar mas información en el Anexo 1: Conceptos Generales

4 Cross-browser: que funciona en distintos navegadores

4. Comparativa: MooTools vs JQuery

4.1 JS y el DOM

Si las tareas a realizar en JavaScript son única estrictamente del tipo "obtener cosas en la página y hacer cosas con eso", entonces jQuery es probablemente la mejor opción. Este se excede ofreciendo un sistema muy expresivo para describir comportamiento en la página en una manera muy intuitiva. Si se está enfocado en el DOM (cambiando propiedades de CSS, animando cosas, obteniendo contenido vía AJAX, etc), la mayoría de lo se finalice escribiendo será cubierto por jQuery. Aunque este provee algunos métodos que no aplican al DOM, por ejemplo, jQuery provee un mecanismo para iterar sobre arrays "*$.each(array, fn)*" o, por ejemplo, ofrece un método trim para strings "*$.trim(str)*". Pero no hay una gran cantidad de estos métodos los cuales tienen carácter de utilitarios, lo cual está perfecto, porque, para la mayor parte, si lo único que se esta realizando es Obtener cosas del DOM, iterando sobre el, y alterándolo de alguna manera (agregando HTML, cambiando estilos, agregando eventos y listeners para click y mouseover, etc). No serán necesarias muchas más utilidades que las mencionadas.

Cuando las diferentes necesidades empiezan a salir del entorno del DOM inician las limitaciones que vienen implicadas con JQuery. Esta, puede decirse, es una de las razones porqué es tan sencillo de aprender, pero esto también limita la forma en que permite escribir JS, por tanto se puede decir que JQuery es un toolkit que se enfoca en ofrecer las utilidades necesarias para el manejo del DOM, éste se desasocia de otras de las funcionalidades de JS como pueden ser la de Herencia y la de soporte para los tipos nativos del leguaje, dado que su principal objetivo es el DOM. Obviamente, que si se desea embarcarse en el manejo de Strings, Arrays, y demas tipos nativos ofrecidos por JS se puede sin ningún tipo de problema, **pero cabe destacar que no es el trabajo de JQuery proveer herramientas que ayuden en esta tarea**.

Aquí es donde MooTools es ampliamente diferente. En vez de enfocarse exclusivamente en el DOM, MooTools toma dentro de su alcance el lenguaje completo. **Esta es una de sus principales diferencias y la razón por la cual se dice de que MooTools es un *Framework* mientras que JQuery es un *Toolkit*.**

4.2 Mismas Tareas

Todas las cosas que se logran mediante la utilización de JQuery se pueden replicar con MooTools, sin embargo lo reciproco no es verdadero, dado a que MooTools tiene una mirada mas amplia de JS, Mientras que JQuery se enfoca en el DOM, independientemente de esto, si se prefiere JQuery ante MooTools se podrán lograr las mismas cosas mezclando JQuery con JS puro.

Ejemplo 3:

```
// JQuery
$(document).ready(function() {
    $("#JQ").click(function(event) {
        alert("Esto es JQuery!");
    });
}); // ventana que dice: Esto es JQuery al hacer click en el elemento con id=JQ
```

```
// MooTools
window.addEvent('domready', function() {
    $$('#MT').addEvent('click', function(event) {
        alert('Esto es MooTools!');
    });
});// ventana que dice: Esto es MooTools al hacer click en el elemento con id=MT
```

Podemos agregarle un poco mas de complejidad al estudio de los casos, a continuación se inserta otro ejemplo:

Ejemplo 4:

```
// JQuery
$(document).ready(function() {
    $("#divId").hover(function() {
        $(this).addClass("green");
    },
    function() {
        $(this).removeClass("green");
    });
}); // Al pasar el cursor del mouse por el elemento con id=divId se le agrega
una clase (green) a este elemento

// MooTools
window.addEvent('domready',function() {
    $$('divId').addEvents({
        mouseenter: function() {
            this.addClass('green');
        },
        mouseleave: function() {
            this.removeClass('green');
        }
    });
}); // Al pasar el cursor del mouse por el elemento con id=divId se le agrega
una clase (green) a este elemento
```

En el ejemplo 4 podemos observar, en la parte de JQuery cuando decimos ‘hover’ estamos esperando a que el cursor pase por encima del elemento que estamos definiendo. Luego definimos otra función que se va a disparar cuando el cursor del mouse ya no este sobre el elemento en cuestión.

En MooTools, esto se hace más verborrágico ya que estamos definiendo que sucede en ambos casos escribiéndolo explícitamente. En el evento ‘mouseenter’ establecemos, mediante el uso de una funcion anónima, que pasa cuando nos posicionamos con el cursor sobre el elemento y en el ‘mouseleave’ definimos, con la misma metodología, que sucede cuando el cursor ya no se encuentra sobre el elemento.

4.3 Sobre los estilos de la sintaxis

Como ya se pudo apreciar en los ejemplos puestos anteriormente, JQuery y MooTools cuentan con sintaxis diferente, en caso de que el programador prefiera la sintaxis de JQuery pero utilice MooTools, la flexibilidad que tiene este le va a permitir cambiar la sintaxis de funciones.

A continuación se implementara el metodo ‘hover’ de JQuery en MooTools.

Ejemplo 5:

```
Element.implement({
    hover : function(enter,leave){
       return this.addEvents({ mouseenter : enter, mouseleave : leave });
    }
});

// en otra parte del codigo se procede a llamar a la función con nueva sintaxis:
$$('#divId').hover(function(){
   this.addClass('green');
},
function(){
   this.removeClass('green');
});
```

De esta manera se ha implementado una mediante la sintaxis de JQuery un evento de ‘mouseenter’ y ‘mouseleave’ en MooTools, incluso, el programador no tiene por que preocuparse en escribir las diferentes implementaciones en los elementos para los cambios de sintaxis, se puede hacer referencia a un plugin que actualmente se encuentra en desarrollo denominado **Mooj**[5] que lo que busca es traer sintaxis similar a la de JQuery a MooTools.

Esto (emular JQuery en MooTools) es algo que jQuery no puede hacer. MooTools puede emular jQuery si lo quieres, pero jQuery no puede emular MooTools, se reitera, se pueden crear clases y extender los objetos nativos, **pero JQuery no ayudará en esta tarea**, aquí valdrá JS puro.

4.4 Re-usar Código

4.4.1 JQuery

Una buena practica para re-usar código dentro de JQuery es mantener la lógica dentro de la función y pasar los parámetros que son variables mediante argumentos.

Ejemplo 6:

```
// Definimos la funcion que vamos a reutilizar
function diccionario(contenedor, term, def) {
    $(contenedor).find(term).hide().end().find(def).click(function() {
        $(this).next().slideToggle();
    });
};

// utilizamos la función definida anteriormente
$(document).ready(function() {
    diccionario('#dic', 'dt', 'dd');
});
```

en este ejemplo se definió una función que lo que hace es recibir tres parámetros los cuales corresponden a un FAQ típico, en donde tenemos la siguiente jerarquía

5 Mooj, ver mas informacion en el Anexo 2: Software de Ejemplo

```
<dt>Esto es un termino</dt>
<dd>Esto es la definicion</dd>
```

Visualmente seria:

```
Esto es un termino
      Esto es la definición
```

Esto procederá a mostrar la definición cuando se hace click en el termino.

De esta manera si se necesita cambiar la lógica de funcionamiento de estas listas, solo se necesita cambiar la lógica dentro del método principal que en este caso se denomina `diccionario(contenedor,term,def).` Además de esto, lo que se esta haciendo es definir una función que sera reutilizable en todo el documento por lo tanto se escribe menos código.

Si bien esta forma de reutilización funciona, es muy 'cruda', JQuery recomienda la creación de JQuery Plug-ins los cuales permitirán la reutilización de código. Mediante el uso de JQuery plugins el ejemplo quedaría de la siguiente manera

---

Ejemplo 7:

```
JQuery.fn.diccionario = fuction(opc) {
    var settings = jQuery.extend({
        term: 'dt',
        def: 'dd'
    }, opc);
  //"this" es el contexto actual; en este caso, los elementos que queremos transformar a un layout
de diccionario
    $(this).find(settings.terms).hide().end().find(settings.definitions).click(function() {
        $(this).next().slideToggle();
    });
    return this;
};

// Procedemos a la utilización del plugin mediante:
$('#dic').diccionario();
```

4.4.2 MooTools

MooTools trata de definir su propia sintaxis y extender los patrones existentes en JavaScript. Una de las maneras de hacer esto es extendiendo los prototipos de los objetos nativos en el lenguaje y en el DOM. Esto significa que en caso se necesite un método para remover los espacios de un string, MooTools incita a los usuarios a agregar un método a String. Es decir que estamos modificando el prototipo nativo de JS. De forma practica quedaría de la siguiente manera:

Ejemplo 8:

```
String.implement({
    trim: function() {
        return this.replace(/^\s+|\s+$/g, '');
    }
});
// Lo unico que queda, es al tener un objeto de tipo String hacer lo siguiente:
objTipoString.trim();
```

Aquí se puede ver que MooTools incluye un robusto framework que facilita su propia extensión con la propia funcionalidad que uno desee agregar, con la intención de que la gente que incluya el framework en la página vaya a usarlo y no use algún otro framework.

Una vez que se entiende el funcionamiento de JavaScript y se logra el poder de extender los objetos nativos, tenemos muchas posibilidades nuevas. Se pueden escribir plugins que modifiquen Element, Date, String o Integer.

Lo expuesto anteriormente pone en claro cuan potente es modificar prototipos. La clave aquí es que en el núcleo del framework MooTools está la noción que está ahí para permitir al usuario programar lo que se desee. Y en caso haya alguna funcionalidad que no esta en el núcleo, se puede exteneder y ser agregado sin ningún inconveniente. El trabajo del framework no es proveer todas las funcionalidades que cada uno requiera, sino proveer las herrramientas que te permiten escribir las cosas que se necesiten.

4.5 Extensión e Implementación de Clases

4.5.1 JQuery

Para ilustrar esto, ser hará referencia al plugin creado en el Ejemplo 7.

Suponga que al código ya existente se quiere agregarle mas funcionalidad, pero se tiene la restricción de que esto se debe lograr sin hacer una copia del código ya existente y agregarle la funcionalidad, es decir, no se debe hacer un ‘fork’ del código existente. Se puede proceder de la siguiente manera:

Ejemplo 9:

```
jQuery.fn.ajaxDiccionario = function(opc) {
    var settings = jQuery.extend({
      //algunas opciones especificas de ajax para obtener la información
        url: '/getfaq.php'
        def: 'dd'
    }, opc);
  //"this" en el contexto actual; en este caso, el elemento que queremos
transformarlo en el layout diccionario
    $(this).find(settings.def).click(function() {
        $(this).load(.....); //la lógica que obtiene el contenido
    });
    this.diccionario(); //la llamada a nuestro plug-in original
});
```

Aquí, al código existente se le esta agregando una funcionalidad Ajax que realizara alguna logica para obtener el contenido que se requiere.

Cosas a notar:

- La clase *diccionario* va a repetir el selector para las definiciones, lo cual puede ser costoso, no hay forma de almacenar las definiciones obtenidas y pasarlas por segunda vez donde son requeridas.
- Dado a que la lógica ajax no se puede agregar al plugin para que se muestre la definición que se obtuvo mediante este método, y puesto que el plugin utiliza un metodo JQuery (slideToggle) que muestra la definición mediante el uso de un efecto, todo se hace más problemático, ya que el efecto puede finalizar antes de que nuestra lógica asíncrona termine de ejecutarse y cargar.

4.5.2 MooTools

Aquí se puede utilizar la parte de orientación a objetos de MooTools y se puede proceder a crear una clase, digase Persona:

Ejemplo 10:

```
var Persona = new Class({
  initialize: function(nom, edad) {
    this.nombre = nom;
    this.edad = edad;
  },
  energia: 1,
  comer: function() {
    this.energia = this.energia + 1;
  }
});

// procedemos a extender esta clase

var Alumno = new Class({
  extends: Persona,
  initialize: function(nom, edad) {
    this.grado = 0;
    this.materiasAprobadas = 0;
    this.parent(nom, edad);
  },
  aprueba: function() {
    this.materiasAprobadas = this.materiasAprobadas + 1;
  },
  pasaGrado: function () {
    this.grado = this.grado + 1;
  }
});

// Utilización
var unAlumno = new Alumno('Juan', 20);
// unAlumno.nombre = Juan
// unAlumno.materiasAprobadas = 0
unAlumno.aprueba;
// unAlumno.materiasAprobadas = 1
```

De esta manera simple e intuitiva hemos extendido la funcionalidad de la clase base Persona mediante una clase Alumno sin perder ninguna de las funcionalidades que se tenían anteriormente, así podemos crear modelos muy complejos sin que el Framework nos ponga alguna limitación.

5. Conclusión

Teniendo en cuenta lo expuesto en el presente documento se puede decir que las dos opciones tienen sus similitudes, pero también características que los hacen ampliamente diferentes. Por un lado se tiene a JQuery, cuyo foco se centra más que nada en el trabajo sobre el DOM, además de tener en mente la rápida codificación y el fácil entendimiento de como trabaja el toolkit para una rápida puesta en marcha de algún proyecto; por otro lado se tiene a MooTools que si bien tiene herramientas que permiten el manejo del DOM su concepción va mas allá del manejo de este,

teniendo fuerte inclinación hacia la orientación a objetos, la extensión de los prototipos nativos ademas de la propia, lo cual lo hace mas complejo a la hora de aprenderlo.

Ninguna de las dos herramientas cuenta con todas las prestaciones existentes, ni tampoco este es el objetivo de ninguna de ellas, lo que se busca es brindarle al usuario las herramientas necesarias para que este pueda implementar lo que necesite.

Anexo 1: Conceptos Generales

Concepto	**Descripción**
API	Aplication Programming Interface es un conjunto de funciones y procedimientos (o métodos, en la programación orientada a objetos) que ofrece cierta biblioteca para ser utilizado por otro *software* como una capa de abstracción.
Expresión Regular	es una secuencia de caracteres que forma un patrón de búsqueda, principalmente utilizada para la búsqueda de patrones de cadenas de caracteres u operaciones de sustituciones. Por ejemplo, el grupo formado por las cadenas *Handel*, *Händel* y *Haendel* se describe con el patrón "H(a|ä|ae)ndel"
Librerías	una **librería** se usa para referirse a un programa que contiene varias funciones para lograr un propósito bien definido y específico. Estas librerías están diseñadas de tal forma que son fácilmente integradas a otros programas que requieren usar la funcionalidad que la librería ofrece. Es posible que una librería utilice otras librerías para completar su funcionalidad.

Anexo 2: Software de Ejemplo

Software	Descripción
	Gecko: Este es un motor de navegador web que fue desarrollado como parte de Firefox, usado en muchas aplicaciones desarrolladas por la Fundación Mozilla y la Corporación Mozilla (Notablemente , el navegador web Firefox, incluyendo su versión móvil y su cliente de correo Thunderbird). Aquí, podrá encontrar el repositorio de desarrollo, y aquí la página oficial
	WebKit: Este es un motor de navegador web desarrollado por empresas como Apple Inc., Adobe Systems, Google y KDE, entre otras, para la renderización de paginas en navegadores web, es el encargado de la visualización de contenido en aplicaciones como el navegador web Safari de Apple, Google realizó una bifurcación de este proyecto para la creación de Blink, utilizado para navegadores web basados en Chromium, tales como Google Chrome y Opera. Aquí, podrá encontrar el repositorio de desarrollo, y aquí la página oficial
	Presto: Es un motor para la visualización de páginas web, fue el motor de Layout para el navegador web Opera por una década, y luego utilizado para darle utilidad como motor a los navegadores Opera Mini y Opera Mobile, en opera 15 el navegador de escritorio empezó a utilizar el backend del Chromium, y fue reemplazado por el motor Blink. Aquí podrá encontrar la página web oficial
	Nightindale: Este es un reproductor de audio y navegador web de código abierto basado en el código fuente del reproductor Songbird. Aquí podrá encontrar la página web oficial
	Picasa [Descontinuado desde el 15 de Marzo el 2016, sucesor: Google Photos]: fue un organizador y visor de imágenes para la organización y edición de fotos digitales, ademas contaba con un sitio web para la compartición de fotos. Aquí podrá encontrar la página web oficial.
	DevHelp: Es un navegador para documentación de API's. Aquí podrá encontrar la página web oficial.
	Instantbird: Es un cliente multiplataforma para mensajería instantánea basada en una librería (*libpurple*) utilizada en el Pidgin (otro cliente para

Anexo 2: Software de Ejemplo

Software	Descripción
	mensajería instantánea) Aquí podrá encontrar la página web oficial.
Mooj	Este plugin trae la sintaxis de JQuery a MooTools, actualmente esta en etapas de desarrollo. Aquí el repositorio oficial.