

JavaScript. Nivel II

Septiembre, 2017



Objetivos del nivel

- Manipulación de arreglos
- Programación orientada a objetos
- Json
- Efectos con JQuery
- Ajax con JQuery

Prerrequisitos del nivel

- JavaScript Nivel I
- POO y UML Nivel I
- Lógica de Programación Nivel III

Acerca de este manual

Este manual pertenece al Centro de Asesoramiento y Desarrollo Informático C.A. (CADIF1). Para obtener más información sobre este u otros cursos visite nuestra sitio Web www.cadif1.com, escribanos a la dirección de correo cadi@cadif1.com o visítenos en nuestra sede ubicada en la Av. Pedro León Torres con calle 59, Centro Comercial Sotavento, piso 2 oficina 27, Barquisimeto estado Lara, Venezuela. Tlf. 0251-7179247, 0251-4410268.

Las marcas mencionadas en este manual son propiedad de sus respectivos dueños. Copyright 2017. Todos los derechos reservados.



Contenido del nivel

Capítulo 1. Arreglos. Parte 1

- 1.1.- Definición de Arreglos.
- 1.2.- Acceso a Los Elementos.
- 1.3.- Recorrido del Arreglo.

Capítulo 2. Arreglos. Parte 2

- 2.1.- Funciones de Arreglos.
- 2.2.- Funciones Para Añadir Elementos o Concatenar Array.
- 2.3.- Funciones Para Extraer Elementos.

Capítulo 3. Arreglos en el Dom

- 3.1.- Elements.
- 3.2.- Forms.
- 3.3.- Childs.

Capítulo 4. Objetos. Parte 1

- 4.1.- Declaración.
- 4.2.- Uso de objetos.

Capítulo 5. Objetos. Parte 2

- 5.1.- Funciones Anónimas.
- 5.2.- Métodos.

Capítulo 6. Objetos. Parte 3

- 6.1.- Arreglos Como Atributos.
- 6.2.- Objetos Como Atributos.
- 6.3.- Arreglos de Objetos.

Capítulo 7. Introducción jQuery

- 7.1.- ¿Qué es jQuery?.
- 7.2.- Ventajas de jQuery.

- 7.3.- Insertar jQuery en la página HTML.
- 7.4.- La función jQuery().
- 7.5.- Ejecución de JQuery.
- 7.6.- Uso de selectores.
- 7.7.- Eventos.

Capítulo 8. Manipular el Html Con Jquery

- 8.1.- Get.
- 8.2.- Set.
- 8.3.- Add.
- 8.4.- Remove.

Capítulo 9. Manipular Estilos Con Jquery

- 9.1.- Asignar Clases de Css.
- 9.2.- Agregar Estilos Css.
- 9.3.- Manipular Tamaños.

Capítulo 10. Efectos Con Jquery. Parte 1

- 10.1.- Fade.
- 10.2.- Slide.
- 10.3.- Animate.

Capítulo 11. Efectos Con Jquery. Parte 2

- 11.1.- Stop.
- 11.2.- Callback.
- 11.3.- Encadenamiento.

Capítulo 12. Ajax

- 12.1.- ¿qué es Ajax?.
- 12.2.- ¿cómo Funciona?.
- 12.3.- Ventajas e Inconvenientes de Ajax.
- 12.4.- Objeto Xmlhttprequest - Ajax.

Capítulo 13. Ajax Usando Jquery

- 13.1.- Introducción.
- 13.2.- Métodos Ajax de Jquery - `$.ajax`.
- 13.3.- Utilización Del Método `$.ajax`.
- 13.4.- Opciones Del Método `$.ajax`.

Capítulo 14. Ajax Usando Jquery – Parte 2

- 14.1.- Métodos Convenientes/abreviados.
- 14.2.- Función Load - `$.fn.load`.
- 14.3.- Funciones `$.get()` y `$.post()`.
- 14.4.- Funciones `$.Getscript ()` y `$.Getjson ()`.

Capítulo 15. Ajax Usando Jquery – Parte 3

- 15.1.- Ajax y Formularios.
- 15.2.- Ejemplo de Formularios Con Ajax.
- 15.3.- Eventos Globales en Ajax.



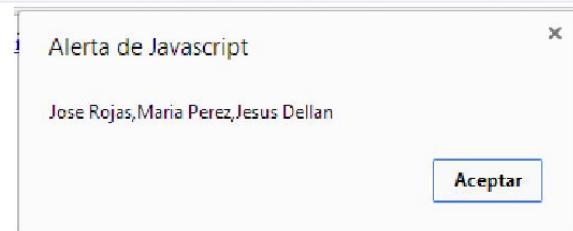
Capítulo 1. ARREGLOS. PARTE 1

1.1.- Definición de Arreglos

Un arreglo es una variable que contiene varios elementos, los cuales están organizados uno después del otro y pueden accederse con un identificador. En JavaScript, los arreglos se declaran usando los corchetes []. A continuación se muestran unos ejemplos:

```
<script language="javascript">
    var nombres=["Jose Rojas","Maria Perez","Jesus Dellan"];
    var edades=[32,25,41];
    var aprobados=[true,false,false];

    alert(nombres);
</script>
```



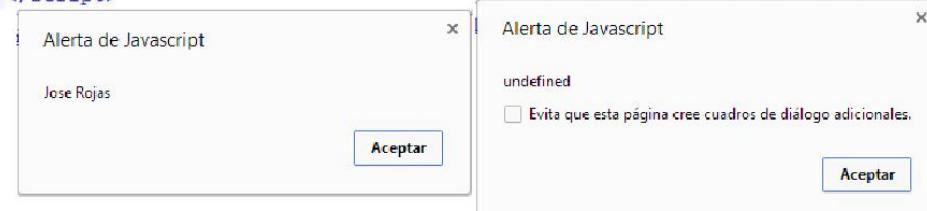
ACADEMIA DE SOFTWARE

1.2.- Acceso a Los Elementos

Cada uno de los elementos del arreglo pueden ser accedidos haciendo uso de los corchetes y la posición donde está ubicado. La posición del primer elemento es la 0 y la posición del último elemento es la cantidad de elementos del arreglo menos 1 ($n-1$). A continuación se muestran un ejemplo de cómo acceder los elementos de un arreglo:

```
<script language="javascript">
    var nombres=["Jose Rojas","Maria Perez","Jesus Dellan","Aura Garcia"];

    alert(nombres[0]);
    alert(nombres[3]);
    alert(nombres[4]); // No existe. retornara undefined
</script>
```



1.3.- Recorrido del Arreglo

Es muy frecuente la necesidad de recorrer todos los elementos de un arreglo. Teniendo claro del punto anterior de cuáles son las posiciones que ocupan los elementos, es necesario realizar un ciclo, que generalmente es el "for". A continuación se muestra un ejemplo:

```
<script language="javascript">
    var nombres=["Jose Rojas","Maria Perez","Jesus Dellan","Aura Garcia"];

    for (i=0;i<4;i++)
        alert("El nombre "+i+" es "+nombres[i]);

</script>
```

Capítulo 2. ARREGLOS. PARTE 2

2.1.- Funciones de Arreglos

Un arreglo en JavaScript es un objeto, el cual automáticamente obtiene unos métodos (funciones) que permiten manipularlos más fácilmente. Algunos de estos métodos son las siguientes:

- length: indica el tamaño del arreglo. Es una propiedad, por lo tanto no lleva paréntesis al llamarlo.
- concat(): concatena 2 arreglos en un tercer arreglo.
- join(): retorna todos los elementos del arreglo como una cadena con los valores separados entre sí con un separador especificado.
- pop(): extrae del arreglo el último elemento.
- push(): agrega un elemento al final del arreglo.
- reverse(): invierte los elementos de un arreglo.
- sort(): ordena los elementos del arreglo en forma lexicográfica (alfabeticamente) de menor a mayor.
- indexOf(x): Busca x dentro del array y devuelve la posición de la primera ocurrencia.
- lastIndexOf(): Busca x dentro del array empezando por el final y devuelve la posición de primera ocurrencia.

En la siguiente imagen se muestra un ejemplo de cómo usar algunos métodos de la clase arreglos:

```
<script language="javascript">
    var nombres=["Jose","Maria","Jesus","Aura"];

    nombres.push("Blanca");

    alert("en el arreglo hay "+nombres.length+" elementos");
    alert(nombres.join(";"));

    nombres.sort();
    alert(nombres.join("#"));
    nombres.pop();
    alert(nombres.join("-"));
</script>
```

2.2.- Funciones Para Añadir Elementos o Concatenar Array

Las siguientes funciones nos permiten agregar elementos a nuestro array así como unir o concatenar varios arreglos.

`concat(it1, it2, ... , itN)`: Devuelve la concatenación de it1, it2, ..., itN con el array sobre el que se invoca.

`push(x)` , Añade x al final del array como nuevo (o nuevos) elemento, y devuelve la nueva longitud del array.

```
var nombre1 = ["Cecilia", "Luis"];
var nombre2 = ["Emilio", "Tobias", "Jose"];
var todos = nombre1.concat(nombre2);

/*
el valor de todos quedaría así:
["Cecilia", "Luis", "Emilio", "Tobias", "Jose"]
*/

var nombre = "Carlos";
todos.push(nombre);

/*
el valor de todos ahora es:
["Cecilia", "Luis", "Emilio", "Tobias", "Jose", "Carlos"]
*/
```

ACADEMIA DE SOFTWARE

`unshift(x)` , Añade x al principio del array como nuevo (o nuevos) elementos.

`splice (ind, 0, it1, it2, ... , itN)` Modifica el array añadiendo los elementos it1, it2, ..., itN, que son insertados en la posición ind (desplazando a los existentes).

```

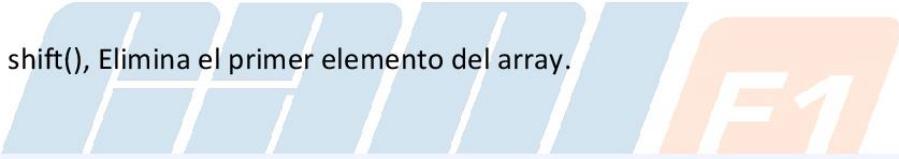
/*
el valor de todos es:
["Cecilia", "Luis", "Emilio", "Tobias", "Jose", "Carlos"]
*/
todos.unshift("Maria", "Karla");
/*
el valor de todos ahora es:
["Maria", "Karla", "Cecilia", "Luis", "Emilio", "Tobias", "Jose", "Carlos"]
*/
todos.splice(2, 0, "Pedro", "Jorge");
/*
el valor de todos despues de splice es:
["Maria", "Karla", "Pedro", "Jorge", "Cecilia", "Luis", "Emilio", "Tobias", "Jose", "Carlos"]
*/

```

2.3.- Funciones Para Extraer Elementos

Las siguientes funciones son utilizadas para extraer elementos o partes de un array con eliminación

- `pop()`, Elimina el último elemento del array .
- `shift()`, Elimina el primer elemento del array.



```

/*
el valor de todos es:
["Maria", "Karla", "Pedro", "Jorge", "Cecilia", "Luis", "Emilio", "Tobias", "Jose", "Carlos"]
*/
todos.pop();
/*
luego de pop() el valor de todos es:
["Maria", "Karla", "Pedro", "Jorge", "Cecilia", "Luis", "Emilio", "Tobias", "Jose"]
*/
todos.shift();
/*
luego de shift() el valor de todos ahora es:
["Karla", "Pedro", "Jorge", "Cecilia", "Luis", "Emilio", "Tobias", "Jose"]
*/

```

- `splice (ind, cuant)`,Modifica el array borrando cuant elementos a partir del índice ind.
- `splice (ind, cuant, it1, it2, ... , itN)`, Modifica el array eliminando cuant elementos e insertando it1, it2, ..., itN, desde el índice ind.
- `delete A[ind]`: Elimina el elemento con índice ind del array , ahora A[ind] es undefined

```
todos.splice(3,2);
/*
luego de splice(3,2) el valor de todos es:
["Karla","Pedro","Jorge","Emilic","Tobias","Jose"]
*/

todos.splice(2,2,"Bernardo","Freddy");
/*
luego de splice(4,2,"Bernardo","Freddy") el valor de todos es:
["Karla","Pedro","Bernardo","Freddy","Tobias","Jose"]
*/

delete todos[1];

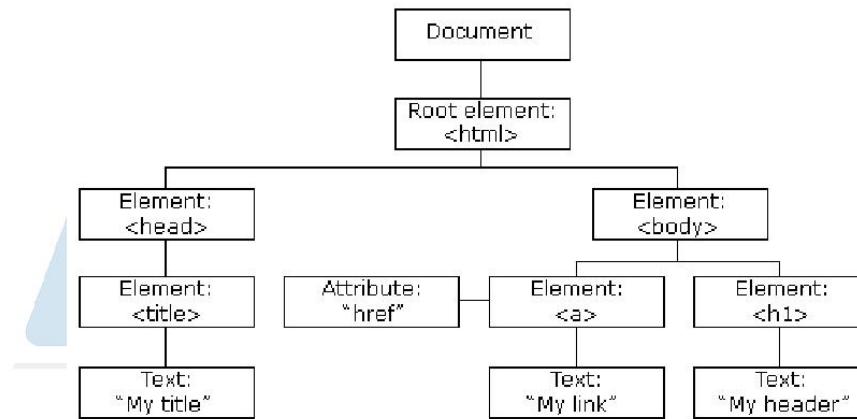
/*
luego de delete todos[1] el valor de todos es:
["Karla",undefined,"Bernardo","Freddy","Tobias","Jose"]
*/
```



Capítulo 3. ARREGLOS EN EL DOM

3.1.- Elements

Element es la clase base más general de la que heredan todos los objetos de un Documento. Sólo tiene métodos y propiedades comunes a todo tipo de elementos. En el DOM, un Element representa una etiqueta HTML. Adicionalmente, los elementos se relacionan entre estos según como estén contenidos unos dentro de otros. La siguiente imagen muestra la relación que existe entre los elementos de un documento HTML:



Los elementos contenidos en un objeto `Element` pueden ser accedidos como un arreglo de objetos (nodos child) llamados `NodeList`. Los principales métodos para buscar elementos en el DOM son:

- `getElementsByName`: devuelve los elementos especificados en `TagName`.
- `getElementsByName`: devuelve los elementos especificados en `Name`.
- `getElementsByClassName`: devuelve los elementos de la clase especificada.
- `document.forms`: devuelve un arreglo de elementos de tipo `<form>`.
- `document.images`: devuelve un arreglo de elementos de tipo ``.

El siguiente ejemplo muestra como buscar todos los elementos HTML de tipo `<p>` y cambiar el color del fondo a través de la propiedad `style` del elemento.

```
var listaNodo = document.getElementsByTagName("p");
for (var i = 0; i < listaNodo.length; i++) {
    listaNodo[i].style.backgroundColor = "red";
}
```

3.2.- Forms

Uno de los elementos más importantes del DOM es elemento HTML <form>, el objeto document.forms es un arreglo que contiene todo los formularios del documento, y cada form a su vez contiene un arreglo de elementos form.elements que contiene todos los input, select, textarea, checkbox, etc. Que forman parte del formulario.

Un ejemplo de cómo buscar todos los forms en el documento HTML es el siguiente:

```
for (var i = 0; i < document.forms.length; i++) {
    console.log(document.forms[i].name); //muestra el nombre del formulario identificado en el name
    console.log(document.forms[i].method); // muestra el metodo de envio : get/post
    console.log(document.forms[i].action); // muestra el action ej: procesar.php
}
```

El arreglo forms puede ser accedido por su índice numérico y asociativo, es decir form[0] nos devuelve el primer formulario, y form['form_login'] accede a un formulario con el name='form_login'.

Un ejemplo de cómo recorrer todos los elementos de cada form seria:

```
var form=document.forms['f1'];
for (var i = 0; i < form.elements.length; i++) {
    console.log(form.elements[i].type); //muestra el tipo de input : text, select, submit, checkbox, etc.
    console.log(form.elements[i].value); //muestra el valor del input
}
```

3.3.- Childs

Un objeto Element pueden tener nodos hijos (child) que pueden ser de tipo element, texto y comentario.

Un objeto NodeList representa una lista de nodos, es una colección de nodos hijos (child) de un elemento HTML. Además un nodo puede tener atributos que se denotan justo después del nombre de la etiqueta por ejemplo el elemento posee un atributo "src" que indica la ruta de la imagen.

Propiedades y métodos más usados:

- createElement: creará un elemento HTML que se insertará posteriormente en un documento HTML.
- createTextNode: crea un nodo de texto con el texto especificado.
- appendChild: Añade un nuevo nodo al elemento como el último nodo hijo.
- insertBefore : inserta un elemento antes del nodo child especificado.
- childNodes: Devuelve el elemento NodeList de los nodos secundarios.
- getAttribute: Devuelve el valor de la propiedad especificada de un elemento.
- removeAttribute: Elimina el atributo especificado del elemento.
- element.id : Establece o devuelve el identificador del elemento.
- element.style: Establece o devuelve el atributo de estilo del elemento.
- element.className: Establece o devuelve el atributo de clase del elemento.
- element.tagName: Devuelve el nombre de la etiqueta del elemento.
- element.innerHTML: Establece o devuelve el contenido del elemento.
- element.textContent: Establece o devuelve el texto del nodo.

El siguiente ejemplo muestra cómo crear un elemento child e insertarlo en el html.

```
// Buscar elemento padre
var elemento_paadre = document.body;
// Crear elemento
var titulo = document.createElement('h1');
// Crear el nodo de texto
var texto = document.createTextNode("mi título");
// Adjuntar el nodo de texto al elemento h1
titulo.appendChild(texto);
// Ahora si, insertar (adjuntar) el elemento hijo (título) al elemento padre (body)
elemento_paadre.appendChild(titulo);
```

Capítulo 4. OBJETOS. PARTE 1

4.1.- Declaración

Una de las características de JavaScript es que permite programar orientado a objetos. La forma de declarar una variable como objeto es usando las {}, encerrando entre ellas las propiedades y los métodos separados por comas (,). El siguiente ejemplo muestra cómo declarar un objeto con varios atributos:

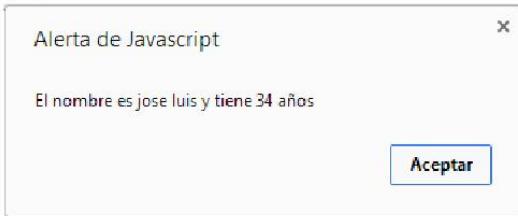
```
<script language="javascript">
    var objeto=
    {
        "nombre":"jose luis",
        "apellido":"rojas dellan",
        "edad":34
    }
</script>
```

4.2.- Uso de objetos

Para usar los objetos se utiliza el punto (.) para acceder a los atributos, ya sea para conocer el valor que tiene o para cambiar el valor. El siguiente ejemplo muestra como acceder a los atributos nombre y edad del objeto "objeto":

```
<script language="javascript">
    var objeto=
    {
        "nombre":"jose luis",
        "apellido":"rojas dellan",
        "edad":34
    }

    alert("El nombre es " + objeto.nombre+
          " y tiene "+objeto.edad+" años");
</script>
```



Capítulo 5. OBJETOS. PARTE 2

5.1.- Funciones Anónimas

Existe otra forma de declarar las funciones, que se denomina funciones anónimas. Básicamente es lo mismo, solo que la declaración varía ligeramente. El siguiente es un ejemplo de cómo se declara una función anónima:

```
<script language="javascript">
    var calcular_bono= function (nro, monto)
    {
        if (nro>0)
            return monto*0.5;
        else
            return monto*0.1;
    }
    var bono= calcular_bono(5,1500);
    alert("El bono es de "+bono+" bs");
</script>
```

5.2.- Métodos

ACADEMIA DE SOFTWARE

Los métodos se declaran usando funciones anónimas en la definición del objeto. Estas funciones pueden o no tener parámetros y pueden o no retornar algo. En el siguiente ejemplo se muestra la declaración de 2 métodos dentro del objeto:

```
<script language="javascript">
    var objeto=
    {
        "nombre":"jose luis",
        "apellido":"rojas dellan",
        "edad":34,
        "saludar": function ()
        {
            alert("Hola ");
        },
        "calcular": function (monto)
        {
            return monto*0.1;
        }
    }
</script>
```

Para ejecutar los métodos, se debe hacer igual que con los atributos. Se coloca el objeto, un punto y luego el nombre del método:

```
<script language="javascript">
    var objeto=
    {
        "nombre":"jose luis",
        "apellido":"rojas dellan",
        "edad":34,
        "saludar": function ()
        {
            alert("Hola ");
        },
        "calcular": function (monto)
        {
            return monto*0.1;
        }
    }
    objeto.saludar();
    alert("El monto es "+objeto.calcular(2500));
</script>
```

En los métodos se puede hacer referencia a los atributos del objeto. Para ello se debe usar el identificador "this". El siguiente ejemplo muestra cómo hacerlo:

```
<script language="javascript">
    var objeto=
    {
        "nombre":"jose luis",
        "apellido":"rojas dellan",
        "edad":34,
        "saludar": function ()
        {
            alert("Mi nombre es "+this.nombre);
        },
        "calcular": function (monto)
        {
            if (this.edad>=18)
                return monto*0.1;
            else
                return monto*0.25;
        }
    }
</script>
```

ACADEMIA DE SOFTWARE

Capítulo 6. OBJETOS. PARTE 3

6.1.- Arreglos Como Atributos

Uno de los atributos de un objeto puede ser un arreglo. En este caso deben usarse los [] al asignarle valor al atributo. Para acceder a los elementos del arreglo se debe colocar el objeto, la propiedad y la posición del valor que se desea acceder:

```
<script language="javascript">
    var objeto=
    {
        "nombre":"jose luis",
        "apellido":"rojas dellan",
        "edad":34,
        "notas": [15,18,20,14],
        "saludar": function ()
        {
            alert("Mi nombre es "+this.nombre);
        }
    }
    objeto.saludar();
    alert("La primera nota fue "+objeto.notas[0]);
</script>
```

ACADEMIA DE SOFTWARE

6.2.- Objetos Como Atributos

Un objeto también puede tener como atributo otro objeto. En ese caso deben colocarse otro par de llaves en la declaración del atributo. Para acceder a los valores de los atributos del objeto interno se debe colocar otro punto y el nombre del atributo. La forma general es objeto.objetointerno.atributo. El siguiente es un ejemplo:

```
<script language="javascript">
    var objeto=
    {
        "nombre":"jose luis",
        "apellido":"rojas dellan",
        "edad":34,
        "notas": [15,18,20,14],
        "fecha": {
            "dia":11,
            "mes":1,
            "anno":1980
        },
        "saludar": function ()
        {
            alert("Mi nombre es "+this.nombre);
        }
    }
    objeto.saludar();
    alert("Nacio el dia "+objeto.fecha.dia);
</script>
```

6.3.- Arreglos de Objetos

Es posible necesitar arreglos de objetos, es decir, que cada elemento de un arreglo sea un objeto. En realidad, todo esto se puede combinar según la necesidad del programador. El siguiente es un ejemplo de un arreglo de objetos:

```
<script language="javascript">
    var alumnos =
    [
        {
            "nombre":"jose",
            "nota":15
        },
        {
            "nombre":"maria",
            "nota":17
        }
    ];
    alert("La nota del primer alumno fue "+alumnos[0].nota);
    alert("El nombre del segundo alumno es "+alumnos[1].nombre);
</script>
```

Capítulo 7. INTRODUCCIÓN JQUERY

7.1.- ¿Qué es jQuery?

jQuery es librería de JavaScript que permite simplificar la manera de interactuar con los documentos HTML, manipular el árbol DOM, manejo de eventos, desarrollar animaciones y agregar interacción con la técnica AJAX en páginas web. jQuery, al igual que otras librerías ofrece una serie de funcionalidades basadas en JavaScript que de otra manera requerirían de mucho más código, es decir, con las funciones propias de esta librerías se logran grandes resultados en menos tiempo y espacio.



jQuery es considerado un Framework, que sirve como base para la programación avanzada de aplicaciones, que aporta una serie de funciones o códigos para realizar tareas habituales. Por decirlo de otra manera, framework son unas librerías de código que contienen procesos o rutinas ya listos para usar.

Los programadores utilizan los frameworks para no tener que desarrollar ellos mismos las tareas más básicas, puesto que en el propio framework ya hay implementaciones que están probadas, funcionan y no se necesitan volver a programar.



jQuery posee las siguientes características:

- Selección de elementos DOM.
- Interactividad y modificaciones del árbol DOM, incluyendo soporte para CSS 1 y CSS 3.
- Manejar y manipular eventos.
- Manipulación de la hoja de estilos CSS.
- Efectos y animaciones.
- Compatible con los navegadores Mozilla Firefox 2.0+, Internet Explorer 6+, Safari 3+, Opera 10.6+ y Google Chrome 8+.
- AJAX.

7.2.- Ventajas de jQuery

La librería jQuery en resumen nos aporta las siguientes ventajas:

- Cross-Browser: es un framework compatible con muchos navegadores. Con jQuery la mayor parte del código que escribas funcionará perfectamente en los principales navegadores incluyendo IE 6.
- Selectores CSS3: jQuery puede trabajar perfectamente con los selectores de la nueva especificación de CSS3.

- Funciones de utilidad: en jQuery se incluye un número de funciones de utilidad que no se incluyen en JavaScript.
- Plugins: el framework jQuery es extensible, esto significa que se puede ampliar para diseñar nuevos plugins que se puedan reutilizar en diferentes proyectos.
- Nos provee de un mecanismo para la captura de eventos.
- Provee un conjunto de funciones para animar el contenido de la página en forma
- muy sencilla.
- Integra funcionalidades para trabajar con AJAX.

7.3.- Insertar jQuery en la página HTML

Lo primero que se debe hacer es descargar la librería de JQuery. El sitio oficial da dos posibilidades para descargar, una que llamada PRODUCTION, que es la adecuada para páginas web en producción, puesto que está minimizada y ocupa menos espacio, con lo que la carga de nuestro sitio será más rápida.

La otra posibilidad es descargar la versión que llaman DEVELOPMENT, que está con el código sin comprimir, con lo que ocupa más espacio, pero se podrá leer la implementación de las funciones del framework, que puede ser interesante en etapa de desarrollo, porque podremos bucear en el código de jQuery por si tenemos que entender algún asunto del trabajo con el framework. La descarga es un archivo .js que contiene el código completo del framework.



Una vez que se ha descargado la librería jQuery, se coloca en la carpeta del sitio y lo siguiente es incluir la librería en los documentos HTML. Esto se logra haciendo referencia al archivo de script en la cabecera del documento.

```
<!doctype html>
<html lang="es">
<head>
    <meta charset="utf-8" />
    <title>jQuery</title>
    <script type="text/javascript" src="jquery.js"></script>
</head>
<body>
    <h1>jQuery</h1>
</body>
</html>
```

Por otra parte, también se puede hacer referencia a la librería directamente desde un CDN (Content Delivery Network que son servidores donde se almacenan archivos para compartir). El CDN ofrece una ventaja de rendimiento al cargar jQuery en servidores repartidos por todo el mundo, ya que si el visitante de su página web ha descargado una copia de jQuery del mismo CDN, no tendrá que volverla a descargar. Para utilizar el CDN oficial de jQuery, simplemente se hace referencia al archivo directamente desde <http://code.jquery.com> en la etiqueta de script:

```
<!doctype html>
<html lang="es">
<head>
    <meta charset="utf-8" />
    <title>jQuery</title>
    <script
        type="text/javascript"
        src="//code.jquery.com/jquery-1.11.0.min.js">
    </script>
</head>
<body>
    <h1>jQuery</h1>
</body>
</html>
```

Existen otros CDN en la web. En el siguiente ejemplo se muestran algunos CDN que también puede utilizar:

```
<!-- CDN de Google -->
<script
  type="text/javascript"
  src="//ajax.googleapis.com/ajax/libs/jquery/1.11.1/jquery.min.js" />
</script>
<script
  type="text/javascript"
  src="//ajax.googleapis.com/ajax/libs/jquery/2.1.1/jquery.min.js" />
</script>
<!-- CDN de Microsoft -->
<script
  type="text/javascript"
  src="http://ajax.aspnetcdn.com/ajax/jquery/jquery-2.1.1.min.js" />
</script>
```

Cuando se está trabajando localmente (desarrollo), se recomienda incluir la librería desde la ruta local del archivo .js (ejemplo: nombre-carpeta/jquery.js), una vez que el proyecto esté listo para llevarlo a producción, se debe colocar la ruta absoluta de cualquiera de los CDN nombrados anteriormente.

Esto se hace con la finalidad de agilizar los tiempos de cargas cuando se trabaja en desarrollo, ya que se accede a la librería directamente. En producción se coloca la ruta del CDN para no sobrecargar el servidor con otra petición, y si el visitante anteriormente ha visitado un sitio web que esté utilizando la misma dirección del CDN, ya tiene descargado la librería jQuery.

El CDN más utilizado y recomendado es el de Google, por tener servidores en toda parte del mundo.

7.4.- La función jQuery()

Todas las instrucciones jQuery empiezan con la llamada a la función "jQuery()", ya que es la puerta de entrada al framework jQuery. Esto es así para evitar que los nombres de las funciones entren en conflicto con otras funciones de otras bibliotecas enlazadas en nuestro proyecto. Así podemos decir que la biblioteca está contenida en un namespace (espacio de nombre) llamado jQuery. jQuery ofrece un alias más corto, `$()`, el signo del dólar. El alias se utiliza mucho más debido a que es más corto.

Esta función es la encargada de buscar y seleccionar los elementos del DOM que coinciden con el selector pasado como parámetro. Es una manera de encapsular los elementos DOM del documento en un objeto jQuery al que se le pueden aplicar todas los métodos definidos en el framework. La función por lo tanto devuelve un

objeto jQuery con el/los elementos coincidentes. La función `$()` por sí sola no realiza nada, simplemente encapsula los elementos en objetos jQuery para poder utilizar los métodos del framework, por lo que esta función se suele utilizar en combinación con otros métodos.

Técnicamente jQuery es un objeto que define un conjunto de funciones (ó métodos) en él. La instrucción típica de jQuery sigue el siguiente esquema:

```
jQuery("selector").metodo(parametro);
```

es equivalente a

```
$( "selector" ).metodo( parametro );
```

7.5.- Ejecución de JQuery

Cuando hacemos ciertas acciones complejas con Javascript tenemos que estar seguros que la página haya terminado de cargar y esté lista para recibir comandos Javascript que utilicen la estructura del documento con el objetivo de cambiar cosas, como crear elementos, quitarlos, cambiar sus propiedades, etc. Si no se espera que la página esté lista para recibir instrucciones, corremos un alto riesgo de obtener errores de Javascript en la ejecución.

Cuando queremos hacer acciones con Javascript que modifiquen cualquier cosa de la página, tenemos que esperar a que la página esté lista para recibir esos comandos.

```
window.onload = function ()  
{  
    .....  
    alert("El documento está listo");  
}
```

No es posible interactuar de forma segura con el contenido de una página hasta que el documento no se encuentre preparado para su manipulación. jQuery permite detectar dicho estado a través de la declaración `$(document).ready()` de forma tal que el bloque se ejecutará sólo una vez que la página esté disponible.

Se trata de detectar el momento en que la página está lista para recibir comandos Javascript que hacen uso del DOM.

`$(document).ready();`

Pero en realidad no hace falta esperar todo ese tiempo de carga de los elementos de la página para poder ejecutar sentencias Javascript que alteren el DOM de la página. Sólo habría que hacerlo cuando el navegador ha recibido el código HTML completo y lo ha procesado al renderizar la página. Para ello, jQuery incluye una manera de hacer acciones justo cuando ya está lista la página, aunque haya elementos que no hayan sido cargados del todo. Esto se hace con la siguiente sentencia.

Con `$(document)` se obtiene una referencia al documento (la página web) que se está cargando. Luego, con el método `ready()` se define un evento, que se desata al quedar listo el documento para realizar acciones sobre el DOM de la página.

```
$(document).ready(function()
{
    console.log('El documento está listo');
});
```

7.6.- Uso de selectores

Lo más simple que podemos hacer con jQuery es obtener algunos elementos y hacer algo con ellos. Si usted entiende los selectores CSS, se le va hacer muy sencillo obtener algunos elementos, sólo tiene que pasar el selector apropiado para su posterior uso. Existen varias formas de seleccionar elementos de la página, los más usados son:

- Selector de elemento
- Selector de id
- Selector de clase

Al seleccionar un elemento, jQuery retorna un objeto que hace referencia a este. Luego de seleccionado el elemento, se puede manipular como quiera. Uno de las tareas más sencillas que se puede realizar con un elemento es esconderlo con hide() o mostrarlo show().

Los selectores de elemento seleccionan todos los elementos que tengan una etiqueta HTML coincidente con la etiqueta pasada como argumento. Se utilizan de la siguiente forma:

```
$(“elemento”)
```

Internamente llama a la función getElementsByTagName() de JavaScript para devolver los elementos coincidentes.

```
$(‘h1’).hide(); //Selecciona todas las etiquetas H1  
$(‘p’).hide(); //Selecciona todas las etiquetas p  
$(‘div’).hide(); //Selecciona todas las etiquetas div  
$(‘strong’).hide(); //Selecciona todas las etiquetas strong
```

El selector de id selecciona el único elemento que tenga un atributo ID cuyo valor sea igual al ID pasado como argumento. Su sintaxis:

```
$(“#id”);
```

Internamente llama a la función getElementById() de JavaScript para devolver el elemento coincidente. Recuerda que el atributo id es único, por lo que no pueden haber dos elementos con el mismo valor en el atributo id.

```
$(‘#contenedor’).hide(); //Selecciona el elemento con el ID cuyo nombre es contenedor  
$(‘#menu’).hide(); //Selecciona el elemento con el ID cuyo nombre es menu  
$(‘#sliders’).hide(); //Selecciona el elemento con el ID cuyo nombre es sliders  
$(‘#barra’).hide(); //Selecciona el elemento con el ID cuyo nombre es barra
```

El selector de clase selecciona todos los elementos que tengan un atributo class cuyo valor sea igual al pasado como argumento. Su sintaxis es la siguiente:

```
$(“.clase”);
```

Internamente llama a la función `getElementsByClassName()` de JavaScript si el navegador la soporta. Si no la soporta comprueba el atributo `class` de cada elemento para devolver los elementos coincidentes. Se permite el uso de múltiples clases. Internet Explorer 6 no soporta el selector múltiple de clases pero a través de jQuery si funciona. Aquí tenemos una de las ventajas de utilizar jQuery, la implementación de una solución cross-browser.

```
$('.lista').hide(); //Selecciona el elemento con la clase cuyo nombre es lista  
$('.icono').hide(); //Selecciona el elemento con la clase cuyo nombre es icono  
$('.items').hide(); //Selecciona el elemento con la clase cuyo nombre es items  
$('.columna').hide(); //Selecciona el elemento con la clase cuyo nombre es columna
```

Selector múltiple

Sintaxis: `$(“selector 1, selector 2, selector 3”);`

Función: selecciona y combina todos los elementos del documento que coincidan con los selectores pasados como argumento.

Descripción: se puede especificar cualquier número de selectores pero hay que tener en cuenta que es ineficiente si se quieren seleccionar en el orden en el que están situados en el documento, ya que se seleccionan en el orden en el que van coincidiendo, pudiendo darse una situación de desorden.

```
ACADEMIA DE SOFTWARE  
$('h1, #menu, .lista').hide(); //Selecciona el elemento H1,  
// el elemento con el ID de nombre menu  
// y el elemento con la clase de nombre lista
```

La elección de buenos selectores es un punto importante cuando se desea mejorar el rendimiento del código. Una pequeña especificidad — por ejemplo, incluir el tipo de elemento (como `div`) cuando se realiza una selección por el nombre de clase — puede ayudar bastante. Por eso, es recomendable darle algunas “pistas” a jQuery sobre en qué lugar del documento puede encontrar lo que desea seleccionar. Por otro lado, demasiada especificidad puede ser perjudicial.

Es importante entender que cualquier selección que realice sólo contendrá los elementos que existían en la página al momento cuando hizo la selección.

```
$( '#contenedor article div p.menu' ).hide(); // Exceso  
$( '#contenedor p.menu' ).hide(); // Ideal
```

7.7.- Eventos

jQuery provee métodos para asociar controladores de eventos (en inglés event handlers) a selectores. Cuando un evento ocurre, la función provista es ejecutada. Dentro de la función, la palabra clave this hace referencia al elemento en que el evento ocurre. La función del controlador de eventos puede recibir un objeto. Este objeto puede ser utilizado para determinar la naturaleza del evento o, por ejemplo, prevenir el comportamiento predeterminado de éste.

jQuery ofrece métodos para la mayoría de los eventos — entre ellos \$.fn.click, \$.fn.focus, \$.fn.blur, \$.fn.change, etc. Estos últimos son formas reducidas del método \$.fn.on de jQuery (\$.fn.bind en versiones anteriores a jQuery 1.7).



```
//Vincular un evento utilizando un método reducido  
$('p').click(function() {  
    console.log('click');  
});  
  
//Vincular un evento utilizando el método $.fn.on  
$('p').on('click', function() {  
    console.log('click');  
});
```

Muy a menudo, elementos en una aplicación estarán vinculados a múltiples eventos, cada uno con una función diferente. En estos casos, es posible pasar un objeto dentro de \$.fn.on con uno o más pares de nombres claves/valores. Cada clave será el nombre del evento mientras que cada valor será la función a ejecutar cuando ocurra el evento.

```
//Vincular múltiples eventos a un elemento
$('p').on({
    'click': function() {
        console.log('clickeado');
    },
    'mouseover': function() {
        console.log('sobrepasado');
    }
});
```

Eventos

- .click() - al hacer clic sobre un elemento.
- .submit() - cuando se pulsa sobre el botón de enviar de un formulario.
- .dblclick() - al hacer doble clic sobre un elemento.
- .hover() - cuando al pasar el ratón por encima de un elemento. Permite pasar una o dos funciones que se ejecutarán cuando los eventos mouseenter y mouseleave ocurran en el elemento seleccionado. Si se pasa una sola función, ésta será ejecutada en ambos eventos; en cambio si se pasan dos, la primera será ejecutada cuando ocurra el evento mouseenter, mientras que la segunda será ejecutada cuando ocurra mouseleave.
- .mouseenter() - cuando el cursor entra en un elemento.
- .mouseleave() - cuando el cursor sale de un elemento.
- .load() - cuando se termina de cargar el elemento.

Capítulo 8. MANIPULAR EL HTML CON JQUERY

8.1.- Get

Hay muchos métodos que puede llamar una vez que haya realizado una selección. Estos métodos generalmente se dividen en dos categorías: los captadores (getters) y definidores (setters). jQuery “sobrecarga” sus métodos, en otras palabras, el método para establecer un valor posee el mismo nombre que el método para obtener un valor. Cuando un método es utilizado para establecer un valor, es llamado método establecedor (en inglés setter). En cambio, cuando un método es utilizado para obtener (o leer) un valor, es llamado obtenedor (en inglés getter).

Getters recuperan un pedazo de información de la selección, y setters alteran la selección de alguna manera. En casi todos los casos, getters operan sólo en el primer elemento de una selección (.text () es una notable excepción); setters operan en todos los elementos de una selección, utilizando lo que se conoce como iteración implícita. Iteración implícita significa que jQuery repite automáticamente todos los elementos de una selección cuando se llama a un método pionero en esa selección. Esto significa que, cuando se quiere hacer algo para todos los elementos de una selección, usted no tiene que llamar a un método definidor (setter) en cada item de su selección.

Existen tres métodos para obtener el contenido:

- `text ()`: Obtiene o devuelve el contenido de texto de los elementos seleccionados
- `html ()`: Obtiene o devuelve el contenido de los elementos seleccionados (incluyendo el formato HTML)
- `val ()`: Obtiene o devuelve el valor de los campos del formulario.

```
// Obtiene el texto del elemento HTML  
// con el id con el valor texto  
$("#btn1").click(function(){  
    $("#texto").text();  
});  
  
// Obtiene el selector HTML con el  
// id con el valor texto  
$("#btn2").click(function(){  
    $("#texto").html();  
});  
  
// Obtiene el valor Cadí del elemento del formulario  
// con el id con el valor texto  
$("#btn3").click(function(){  
    $("#texto").val();  
});
```

Veamos cómo podemos utilizar jQuery para modificar nuestra página web. jQuery incluye una manera útil de obtener y establecer propiedades CSS a los elementos.

Las propiedades CSS que incluyen como separador un guión del medio, en JavaScript deben ser transformadas a su estilo CamelCase. Por ejemplo, cuando se la utiliza como propiedad de un método, el estilo CSS font-size deberá ser expresado como fontSize. Sin embargo, esta regla no es aplicada cuando se pasa el nombre de la propiedad CSS al método \$.fn.css — en este caso, los dos formatos (en CamelCase o con el guión del medio) funcionarán.

```
// Obtenemos el tamaño de la fuente  
// del elememto con el valor texto  
// y luego lo mostramos en pantalla  
$("#btn1").click(function(){  
    var mostrar = $("#texto").css("font-size");  
    alert(mostrar);  
});
```

8.2.- Set

Existen tres métodos para establecer el contenido:

text () - Establece o devuelve el contenido de texto de los elementos seleccionados
html () - Establece o devuelve el contenido de los elementos seleccionados (incluyendo el formato HTML)
val () - Establece o devuelve el valor de los campos del formulario.

```
// Agrega el texto Hola mundo! al elemento HTML
// con el id con el valor texto
$("#btn1").click(function(){
    $("#texto").text("Hola mundo!");
});

// Agrega el elemento <strong>Hola mundo!</strong>
// al elemento HTML con el id con el valor texto
$("#btn2").click(function(){
    $("#texto").html("<strong>Hola mundo!</strong>");
});

// Agrega el valor Cadi al elemento del formulario
// con el id con el valor texto
$("#btn3").click(function(){
    $("#texto").val("Cadi");
});
```

Los atributos de los elementos HTML que conforman una aplicación pueden contener información útil, por eso es importante poder establecer y obtener esa información. El método \$.fn.attr actúa tanto como método establecedor como obtenedor. Además, al igual que el método \$.fn.css, cuando se lo utiliza como método establecedor, puede aceptar un conjunto de palabra clave-valor o un objeto contenido más conjuntos.

```
$( 'a' ).attr( 'href' , 'contacto.html' );
$( 'a' ).attr( {
    'title' : 'Servicios las 24 horas del día',
    'href' : 'contacto.html'
});
```

8.3.- Add

Con jQuery, hay una manera fácil de agregar un nuevo elemento y nuevo contenido. Tenemos cuatro métodos que podemos usar para agregar contenido nuevo:

- append() - Inserta el contenido en el final del elemento seleccionado
- prepend() - Inserta el contenido en el inicio del elemento seleccionado
- after() - Inserta contenido después del elemento seleccionado
- before() - Inserta contenido antes del elemento seleccionado



```
$( "h1" ).append(" CADI. "); // <h1>jQuery CADI </h1>
$( "h1" ).prepend(" CADI. "); // <h1> CADI jQuery</h1>
$( "h1" ).after(" CADI. "); // <h1>jQuery</h1> CADI
$( "h1" ).before(" CADI. "); // CADI<h1>jQuery</h1>
```

En todos los casos, podemos insertar solo algunos elementos HTML o texto en el principio, ante, después o al final del elemento HTML seleccionado. Sin embargo, los métodos pueden tomar un número infinito de nuevos elementos pasados como parámetros. Los nuevos elementos generados con texto o HTML se pueden hacer con jQuery, o con JavaScript puro y con elementos del DOM.

```
function agregarTexto() {
    var texto1 = "<p>Texto.</p>";
    var texto2 = $("<p></p>").text("Texto.");
    var texto3 = document.createElement("p");
    texto3.innerHTML = "Texto.";
    $("p").append(texto1, texto2, texto3); // Agrega los nuevos elementos
}
```

8.4.- Remove

Existen dos formas de remover elementos de una página: Utilizando `$.fn.remove` o `$.fn.detach`. Cuando deseé remover de forma permanente al elemento, utilice el método `$.fn.remove`. Por otro lado, el método `$.fn.detach` también remueve el elemento, pero mantiene la información y eventos asociados al mismo, siendo útil en el caso que necesite reinsertar el elemento en el documento.

Por otro lado, si se desea mantener al elemento pero se necesita eliminar su contenido, es posible utilizar el método `$.fn.empty`, el cual “vaciará” el contenido HTML del elemento.

```
$('#menu').remove(); // Quita el elemento permanentemente  
$('#menu').detach(); // Quita el elemento pero mantiene información y eventos  
$('#menu').empty(); // Mantiene el elemento y vacía el contenido
```

El método `css ()` establece o devuelve una o más propiedades de estilo para los elementos seleccionados.

```
// Removemos todos los elementos  
// <p> con la clase "italic"  
$("p").remove(".italic");
```

Capítulo 9. MANIPULAR ESTILOS CON JQUERY

9.1.- Asignar Clases de Css

Digamos que usted quiere cambiar el código HTML de todos los elementos de la lista en su página. Para ello, usaríamos el método `.html ()`, que va a cambiar el HTML de todos los elementos de la lista seleccionados.

También usted puede pasar una función con los métodos setters de jQuery. El valor de retorno de esta función, se utiliza como el nuevo valor, y recibe dos argumentos: el índice del elemento en la selección, y el antiguo valor del elemento que usted está tratando de cambiar. Esto es útil cuando se necesita información sobre el estado actual de un elemento con el fin de configurar correctamente el nuevo estado.

```
var $h1 = $('h1');

$h1.addClass('cadi');           // Añade la clase cadi
$h1.removeClass('cadi');        // Quita la clase cadi
$h1.toggleClass('cadi');         // Añade o quita la clase cadi,
                                // dependiendo del estado

if ($h1.hasClass('cadi')) { ... } // Verifica si tiene la clase cadi
```

ACADEMIA DE SOFTWARE

9.2.- Agregar Estilos Css

El método `css()` establece o retorna una o más propiedades del elemento seleccionado.

Para retornar el valor específico de la propiedad CSS, usamos la siguiente sintaxis:

```
css("nombreNombrePropiedad");
```

Y para establecer una propiedad CSS, usamos la siguiente sintaxis:

```
css("nombreNombrePropiedad","valor");
```

Para establecer varias propiedades CSS, usamos la siguiente sintaxis:

```

css({"nombreNombrePropiedad":"valor","nombreNombrePropiedad":"valor",..
.});

// Retorna el valor del color del fondo
$("p").css("background-color");

// Establecemos el color del fondo en amarillo
$("p").css("background-color","yellow");

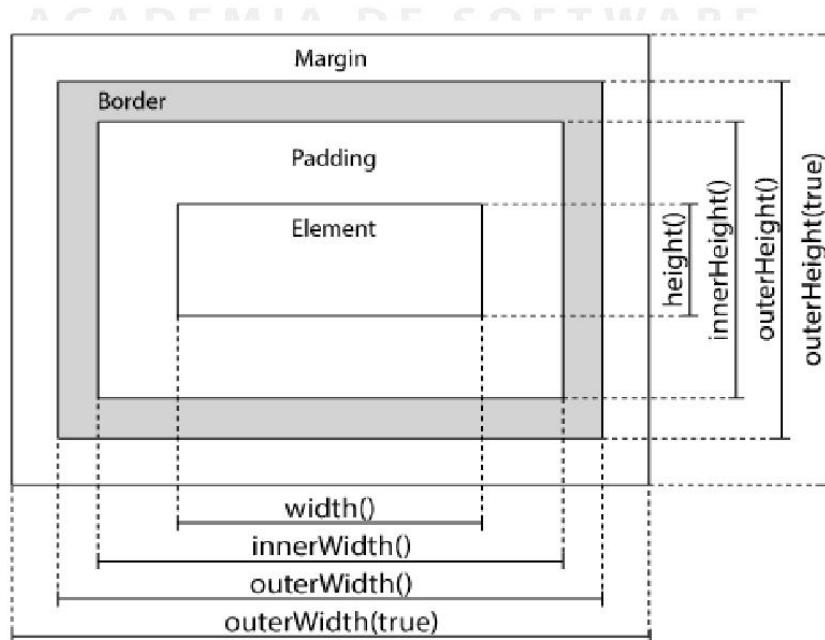
// Establecemos el color del fondo en amarillo
// y el tamaño de la fuente a 40px
$("p").css({"background-color":"yellow","font-size":"40px"});

// Para leerlo más fácilmente, lo podemos tabular
$("p").css({
  "background-color":"yellow",
  "font-size":"40px"
});

```

9.3.- Manipular Tamaños

Para manipular las dimensiones de los elementos HTML es importante conocer el modelo de cajas de HTML. La siguiente imagen muestra los elementos que son parte del modelo de cajas, que son propiedades CSS de cualquier etiqueta :



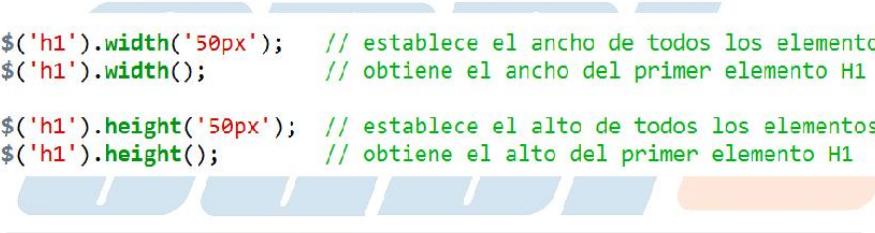
jQuery tiene varios métodos importantes para trabajar con las dimensiones (tamaños) de los elementos:

- width(): establece o devuelve el ancho de un elemento (no incluye el padding, border o margin)
- height(): establece o devuelve la altura de un elemento (no incluye el padding, border o margin).
- innerWidth() devuelve el ancho de un elemento (incluyendo el padding).
- innerHeight() devuelve la altura de un elemento (incluyendo el padding).
- outerWidth() devuelve el ancho de un elemento (incluyendo padding y border)
- outerHeight() devuelve la altura de un elemento (incluyendo padding y border)

Los siguientes son ejemplos de como conocer y establecer dimensiones con JQuery:

```
$('h1').width('50px'); // establece el ancho de todos los elementos H1
$('h1').width(); // obtiene el ancho del primer elemento H1

$('h1').height('50px'); // establece el alto de todos los elementos H1
$('h1').height(); // obtiene el alto del primer elemento H1
```



En el siguiente ejemplo se muestra como usar las propiedades width, innerWidth, height e innerHeight:

```
// En este ejemplo se retorna solo el
// ancho y altura de un elemento específico
$("#inicio").click(function(){
    var anchoAlto="";
    anchoAlto+="Ancho: " + $("#caja").width() + "<br>";
    anchoAlto+="Alto: " + $("#caja").height();
    $("#caja").html(anchoAlto);
});

// En este ejemplo se retorna el ancho y la
// altura más el padding de un elemento específico
$("#inicio").click(function(){
    var anchoAlto="";
    anchoAlto+="Ancho + padding: " + $("#caja").innerWidth() + "<br>";
    anchoAlto+="Alto + padding: " + $("#caja").innerHeight();
    $("#caja").html(anchoAlto);
});
```

Capítulo 10. EFECTOS CON JQUERY. PARTE 1

10.1.- Fade

Con jQuery, agregar efectos a una página es muy fácil. Estos efectos poseen una configuración predeterminada pero también es posible proveerles parámetros personalizados. Además es posible crear animaciones particulares estableciendo valores de propiedades CSS. Los efectos más utilizados que ya vienen incorporados dentro de la biblioteca en forma de métodos son:

- `$.fn.show` : muestra el elemento seleccionado.
- `$.fn.hide` : oculta el elemento seleccionado.
- `$.fn.fadeIn` : de forma animada, cambia la opacidad del elemento seleccionado al 100%.
- `$.fn.fadeOut` : de forma animada, cambia la opacidad del elemento seleccionado al 0
- `$.fn.slideDown` : muestra el elemento seleccionado con un movimiento de deslizamiento vertical.
- `$.fn.slideUp` : oculta el elemento seleccionado con un movimiento de deslizamiento vertical.
- `$.fn.slideToggle` : muestra o oculta el elemento seleccionado con un movimiento de deslizamiento vertical, dependiendo si actualmente el elemento está visible o no.

Con jQuery puede desaparecer elementos dentro y fuera de la visibilidad. jQuery tiene los siguientes métodos:

- `fadeIn()`
- `fadeOut()`
- `fadeToggle()`
- `fadeTo()`

La sintaxis para aplicar un efecto sigue la siguiente sentencia:

```
$(selector).fadeIn(velocidad,disparador);
```

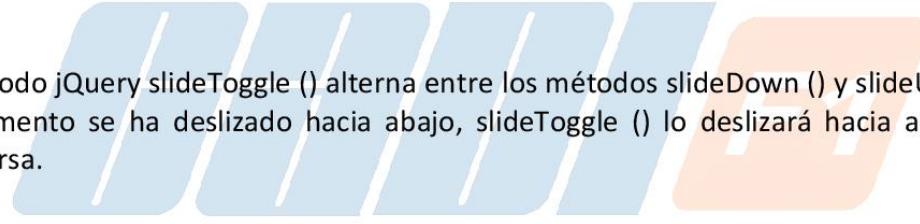
El parámetro de velocidad es opcional y especifica la duración del efecto y puede tomar los siguientes valores: "slow", "fast", o milisegundos. La opción del disparador es una función que se ejecutará luego que el efecto sea completado.

```
$("#menu").click(function(){
    $("#div1").fadeIn();
    $("#div2").fadeOut("slow");
    $("#div3").fadeIn(3000);
});
```

10.2.- Slide

Con jQuery puede crear un efecto de deslizamiento en los elementos hacia arriba y hacia abajo. jQuery tiene los siguientes métodos para los slide:

- slideDown()
- slideUp()
- slideToggle()



El método jQuery slideToggle () alterna entre los métodos slideDown () y slideUp(). Si el elemento se ha deslizado hacia abajo, slideToggle () lo deslizará hacia arriba y viceversa.

```
$("#menu").click(function(){
    $("#panel").slideDown();
});

$("#menu").click(function(){
    $("#panel").slideUp();
});

$("#flip").click(function(){
    $("#panel").slideToggle();
});
```

10.3.- Animate

El método jQuery animate() permite crear animaciones personalizadas. La sentencia es la siguiente:

```
$(selector).animate({parámetros},velocidad,disparador);
```

El parámetro define las propiedades CSS para ser animadas. El parámetro de velocidad especifica la duración del efecto. La velocidad puede tomar los siguientes valores: "slow", "fast", o milisegundos. El disparador es una función opcional que se ejecutará una vez finalizada la animación.

```
// El siguiente ejemplo demuestra un uso sencillo
// del método animado (), se mueve un elemento de
// clase .caja a la derecha hasta que haya alcanzado
// los 250px de la propiedad CSS left
$("#boton").click(function(){
    $(".caja").animate({left:'250px'});
});

// Ahora se le pasan varios parámetros
// Note que múltiples propiedades pueden
// ser animadas en el mismo momento
$("#boton").click(function(){
    $(".caja").animate(
        {
            left: '250px',
            opacity: '0.5',
            height: '150px',
            width: '150px'
        }
);
});
```

También puede especificar a la propiedad los valores como "show", "hide", o "toggle".

Las propiedades relacionadas al color no pueden ser animadas utilizando el método \$.fn.animate, pero es posible hacerlo a través de la extensión color plugin.

```
$('span').animate(  
{  
    left : "50px",  
    opacity : 0.25,  
    height : "toggle"  
}, 300, // duración  
function() {  
    console.log('realizado');  
});
```



Capítulo 11. EFECTOS CON JQUERY. PARTE 2

11.1.- Stop

El método stop() es usado para detener la animación o el efecto antes de que haya finalizado, trabaja sobre todas las funciones de efectos que tiene jQuery, incluyendo los sliding, fading y animaciones personalizadas. Su sintaxis es la siguiente:

```
$(selector).stop(stopAll,goToEnd);
```

El parámetro stopAll es opcional y especifica si también la cola de la animación debe ser limpiado o no. El valor predeterminado es falsa, lo que significa que sólo la animación activa se detuvo, permitiendo que cualquiera animación en la cola pueda llevarse a cabo después. El parámetro goToEnd es opcional y especifica si o no se completa la animación actual inmediatamente. El valor predeterminado es falso. Así que, por defecto, el método stop () mata a la animación actual que se realiza en el elemento seleccionado.



```
$("#boton-parar").click(function(){
    $("#panel").stop();
});
```

ACADEMIA DE SOFTWARE

11.2.- Callback

Una función disparadora es ejecutada después que el efecto actual es 100% finalizado.

Las sentencias de JavaScript se ejecutan línea por línea. Sin embargo, con los efectos, la siguiente línea del código se puede ejecutar a pesar de que el efecto no ha terminado. Esto puede crear errores. Para evitar esto, se puede crear una función disparadora.

```
// El ejemplo debajo tiene un disparador  
// que será ejecutado después que el efecto  
// de ocultamiento haya sido completado  
$("#ocultar").click(function(){  
    $("p").hide("slow",function(){  
        alert("El párrafo ha sido ocultado");  
    });  
});  
  
// A diferencia de este ejemplo que no tiene  
// un disparador, y el cuadro de alerta  
// se mostrará antes de que se complete  
// el efecto de ocultamiento  
$("#ocultar").click(function(){  
    $("p").hide(1000);  
    alert("El párrafo ha sido ocultado");  
});
```

11.3.- Encadenamiento

Hasta ahora se ha manipulado con jQuery un elemento con una operación a la vez (uno tras otro). Sin embargo, existe una técnica llamada encadenamiento, que permite ejecutar varios comandos de jQuery, uno tras otro, en el mismo elemento(s). Los encadenamiento permiten correr múltiples métodos jQuery en una única sentencia. De esta forma, los navegadores no tienen que encontrar el mismo elemento(s) más de una vez. Para encadenar una acción, sólo tiene que anexar la acción a la acción anterior.

```
// El elemento "#cadena" primero cambia a rojo,  
// luego se desliza hacia arriba,  
// y luego se desliza hacia abajo  
$("#cadena").css("color","red").slideUp(2000).slideDown(2000);
```

Capítulo 12. AJAX

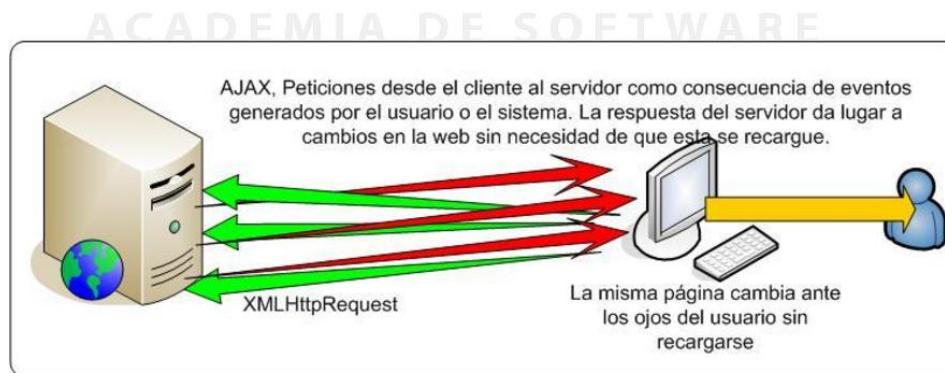
12.1.- ¿qué es Ajax?

AJAX son las siglas de Asynchronous JavaScript And XML, (Javascript asíncrono y XML). No es en sí un lenguaje de programación, sino una nueva técnica que combina varios lenguajes de programación.

La ventaja de ajax respecto a otros lenguajes de programación web es la asincronía. Esto consiste en que cuando queremos intercambiar datos con el servidor (por ejemplo enviar o comprobar un formulario, consultar una base de datos, etc), la página no se queda parada esperando la respuesta, sino que se pueden seguir ejecutando acciones mientras tanto.

Podríamos dar esta definición de Ajax: "Ajax es un conjunto de métodos y técnicas que permiten intercambiar datos con un servidor y actualizar partes de páginas web sin necesidad de recargar la página completamente".

Aunque Ajax se pensó inicialmente para transferir datos en un solo formato (XML), actualmente Ajax permite la transmisión de datos en múltiples formatos: XML, JSON, EML, texto plano, HTML, etc.



Ejemplos de Ajax:

-La función de autocompletar de google, la página va mostrándonos sugerencias acorde al término escrito.



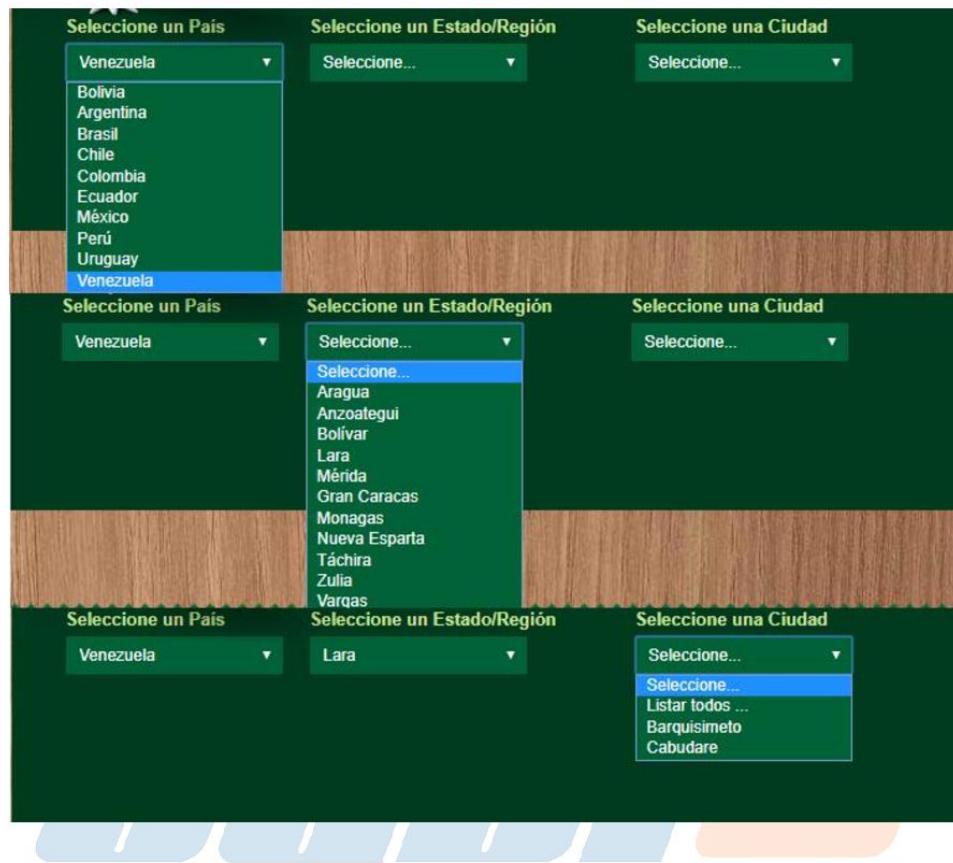
-Consulta de información almacenadas en base de datos del lado del servidor.

The screenshot shows a web interface for the CNE (Consejo Nacional Electoral) of Venezuela. On the left, there's a sidebar with links like "RESPUESTA DEL SERVIDOR", "Normativa Electoral", "Sistema Electoral", "La Institución", etc. The main content area has a header "REGISTRO ELECTORAL - Consulta de Datos". It displays a "DATOS DEL ELECTOR" section with the following information:

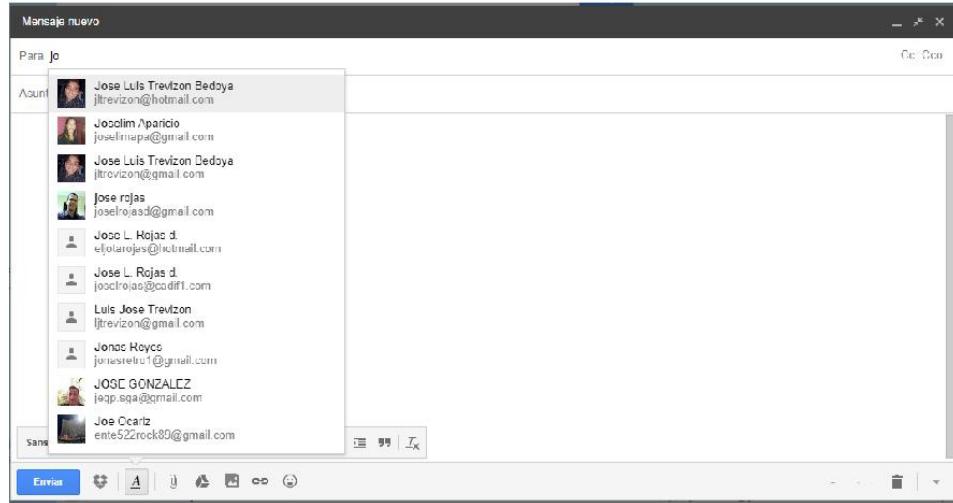
- Cédula: V-14631498
- Nombre: JOSE LUIS TREVIZON BEDOYA
- Estado: EDO. LAURA
- Municipio: MP. PALAVECINO
- Parroquia: PG. JOSE C. BAS VIDAS
- Centro: UNIDAD EDUCATIVA ESTADAL NINA RAMONA VICTORIA OROZCO URBANIZACÓN FIEDRA AZUL FRONTE AVENIDA FINAL DE LA
- Dirección: AV. NICA PRINCIPAL MIRIA A/U DRH-CHA (AIT) - 1 VOLVIENDA CALLE 5 FRENTE AL SECTOR 6 CASA

Below this, there's a "Registro Electoral Julio 2017" section with "Imprimir" and "Cerrar" buttons. To the right, there's a "PETICIÓN DEL USUARIO" section with a "Consulte sus Datos" form containing a dropdown menu set to "V" and the number "14631498", with a "Buscar" button. Red arrows point from the text "RESPUESTA DEL SERVIDOR" to the sidebar and from "PETICIÓN DEL USUARIO" to the "Consulte sus Datos" form.

-Listados de selección dependientes Ej: País, Estado, Ciudad.



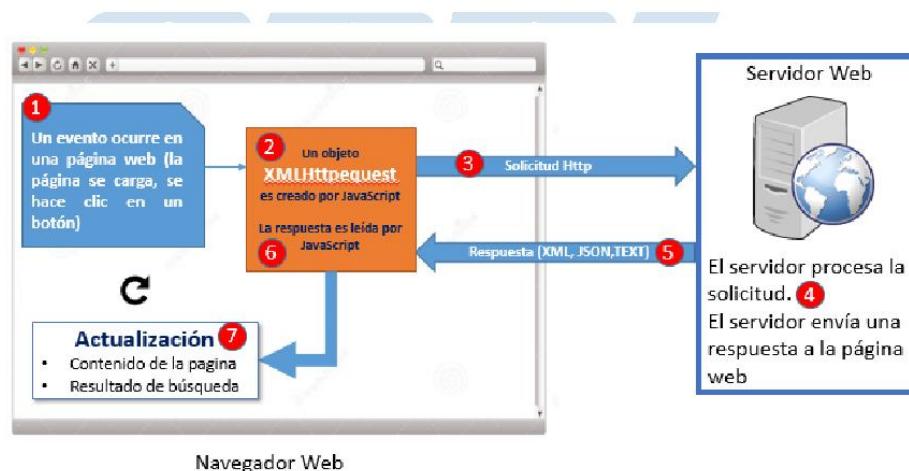
-Filtrando un correo de la libreta de contactos del correo electrónico



12.2.- ¿cómo Funciona?

Con ajax podemos crear páginas interactivas. En éstas solicitamos datos al servidor, los cuales podemos tener guardados en otras páginas o en bases de datos. Usando sólo PHP u otros lenguajes de servidor, al hacer una petición, el servidor realiza una serie de tareas y después nos devuelve los datos. Mientras se realiza este proceso la página permanece en espera, es decir está parada. Esto puede que no tenga importancia si se manejan pocos datos y el servidor tiene potencia para responder rápidamente. Sin embargo si se manejan muchos datos o hay muchas peticiones a la vez (páginas muy visitadas), el tiempo de respuesta puede ser más largo. Mientras se espera la respuesta la página permanece parada.

Con ajax al trabajar de forma asíncrona, permite que el usuario pueda seguir haciendo otras cosas o la página pueda mostrar otras cosas mientras se produce la respuesta.



12.3.- Ventajas e Inconvenientes de Ajax

Las ventajas que proporciona Ajax son varias:

- No es necesario recargar y redibujar la página web completa, con lo que todo es más rápido.
- El usuario no percibe que haya demoras: está trabajando y al ser las comunicaciones en segundo plano no hay interrupciones.

- Los pasos que antes podía ser necesario dar cargando varias páginas web pueden quedar condensados en una sola página que va cambiando gracias a Ajax y a la información recibida del servidor.

Ajax también tiene inconvenientes:

- El usuario puede perder la capacidad para hacer cosas que hacía con webs tradicionales puesto que no hay cambio de página web. Por ejemplo usar los botones de avance y retroceso del navegador o añadir una página a favoritos puede dejar de ser posible. Esto en algunos casos no es deseable.
- El desarrollo de aplicaciones web se puede volver más complejo. Supongamos que antes tuviéramos un proceso en el que avanzábamos a través de varias páginas web como 1, 2, 3. De este modo la organización resulta sencilla. Si condensamos todo en una sola página web: 1, escribir y depurar el código puede volverse más complicado. En sitios complejos, puede ser muy difícil depurar errores.
- Existen problemas y restricciones de seguridad relacionados con el uso de Ajax. Hay que tener en cuenta que por motivos de seguridad no todos los procesos se pueden realizar del lado del cliente (que por su propia naturaleza es "manipulable"). También existen restricciones de seguridad para impedir la carga de contenidos mediante Ajax desde sitios de terceras partes.
- La indexación para los motores de búsqueda se ve dificultada, con lo cual nuestros sitios web pueden perder visibilidad en los buscadores. No es lo mismo un contenido "constante" o aproximadamente estático, fácilmente rastreable para un buscador, que un contenido "cambiable" en función de la ejecución de JavaScript, difícilmente rastreable para un buscador.

12.4.- Objeto XMLHttpRequest - Ajax

Este objeto es fundamental para la programación con ajax. Podríamos decir que ajax consiste sobre todo en aprovechar, mediante varios lenguajes de programación, todas las posibilidades que tiene este objeto.

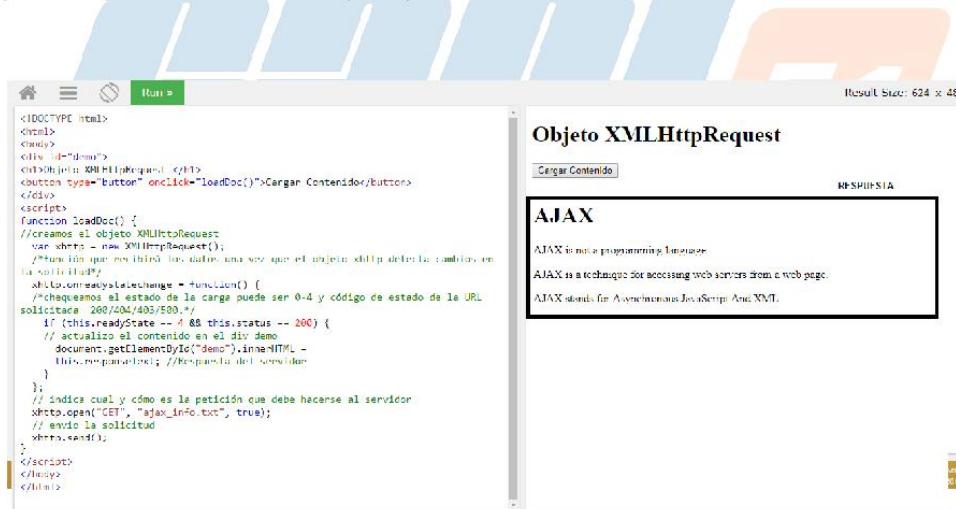
Por tanto nos sirve tanto para incorporar datos en la página que tengamos en otros archivos del servidor, como para enviar datos de la página al servidor, y todo esto de forma asíncrona.

La diferencia entre utilizar XMLHttpRequest o simplemente utilizar un lenguaje como PHP, ASP, JSP es que XMLHttpRequest permite mandar los datos en segundo plano, sin tener que parar el flujo de la página mientras se envian o reciben los datos.

XMLHttpRequest es creado por Microsoft y aparece por primera vez como un objeto ActiveX (Microsoft.XMLHTTP), en Internet Explorer 5. posteriormente se incorpora al resto de navegadores como XMLHttpRequest. Internet Explorer también usa esta forma a partir de la versión 7.

El objeto XMLHttpRequest está incorporado de forma nativa a la mayoría de navegadores, por lo que en javascript depende directamente del objeto window.

Ejemplo de Ajax usando XMLHttpRequest:



Capítulo 13. AJAX USANDO JQUERY

13.1.- Introducción

Uno de los objetivos principales que los desarrolladores tenemos cuando decidimos usar jQuery es implementar comportamientos AJAX para crear aplicaciones web más parecidas a las aplicaciones de escritorio, lo que se llama habitualmente como RIA (aplicación de Internet enriquecida).

En jQuery existen diversas funciones para producir AJAX, con diversas aplicaciones que nos ofrecen soluciones diferentes y que serán acordes a distintos tipos de necesidades.

En jQuery existen métodos AJAX con diversos niveles de complejidad. En la propia documentación de AJAX, los creadores los clasifican en métodos de "alto nivel" y "bajo nivel", alto nivel se refiere a funciones de uso simple para el usuario pero con una configuración muy reducida, y bajo nivel son más cercanas a lo que sería el AJAX del navegador, más personalizables y más potente.

Alto nivel: `$.get`, `$.post`, `$.getScript`, `$.getJSON`, `$.fn.load`

Bajo nivel: `$.ajax`

13.2.- Métodos Ajax de Jquery - `$.ajax`

jQuery posee varios métodos para trabajar con Ajax. Sin embargo, todos están basados en el método `$.ajax`, por lo tanto, su comprensión es obligatoria.

El método `$.ajax` es configurado a través de un objeto, el cual contiene todas las instrucciones que necesita jQuery para completar la petición. Dicho método es particularmente útil debido a que ofrece la posibilidad de especificar acciones en caso que la petición haya fallado o no. Además, al estar configurado a través de un objeto, es posible definir sus propiedades de forma separada, haciendo que sea más fácil la reutilización del código.

Generalmente, es preferible utilizar el método `$.ajax` en lugar de los otros, ya que ofrece más características y su configuración es muy comprensible.

13.3.- Utilización Del Método \$.ajax

Estos son los métodos más utilizados de \$.ajax. Puede visitar api.jquery.com/jQuery.ajax para consultar la documentación sobre todas las opciones disponibles en el método.

```
$.ajax({
    // la URL para la petición
    url : 'post.php',

    // la información a enviar
    // (también es posible utilizar una cadena de datos)
    data : { id : 123 },

    // especifica si será una petición POST o GET
    type : 'GET',

    // el tipo de información que se espera de respuesta
    dataType : 'json',

    // código a ejecutar si la petición es satisfactoria;
    // la respuesta es pasada como argumento a la función
    success : function(json) {
        $('<h1>').text(json.title).appendTo('body');
        $('<div class="content">');
        .html(json.html).appendTo('body');
    },

    // código a ejecutar si la petición falla;
    // son pasados como argumentos a la función
    // el objeto de la petición en crudo y código de estatus de la petición
    error : function(xhr, status) {
        alert('Disculpe, existió un problema');
    },
    // código a ejecutar sin importar si la petición falló o no
    complete : function(xhr, status) {
        alert('Petición realizada');
    }
});
```

13.4.- Opciones Del Método \$.ajax

El método \$.ajax posee muchas opciones de configuración, y es justamente esta característica la que hace que sea un método muy útil. Para una lista completa de las opciones disponibles, puede consultar api.jquery.com/jQuery.ajax; a continuación se muestran las más comunes:

- **async:** Establece si la petición será asíncrona o no. De forma predeterminada el valor es true. Debe tener en cuenta que si la opción se

establece en false, la petición bloqueará la ejecución de otros códigos hasta que dicha petición haya finalizado.

- cache: Establece si la petición será guardada en la cache del navegador. De forma predeterminada es true para todos los dataType excepto para script y jsonp. Cuando posee el valor false, se agrega una cadena de caracteres anti-cache al final de la URL de la petición.
- complete: Establece una función de devolución de llamada que se ejecuta cuando la petición esta completa, aunque haya fallado o no. La función recibe como argumentos el objeto de la petición en crudo y el código de estatus de la misma petición.
- data: Establece la información que se enviará al servidor. Esta puede ser tanto un objeto como una cadena de datos (por ejemplo foo=bar&baz=bim)
- dataType: Establece el tipo de información que se espera recibir como respuesta del servidor. Si no se especifica ningún valor, de forma predeterminada, jQuery revisa el tipo de MIME que posee la respuesta.
- error: Establece una función de devolución de llamada a ejecutar si resulta algún error en la petición. Dicha función recibe como argumentos el objeto de la petición y el código de estatus de la misma petición.
- success: Establece una función a ejecutar si la petición a sido satisfactoria. Dicha función recibe como argumentos la información de la petición (convertida a objeto JavaScript en el caso que dataType sea JSON), el estatus de la misma y el objeto de la petición.
- timeout: Establece un tiempo en milisegundos para considerar a una petición como fallada.
- type: De forma predeterminada su valor es GET. Otros tipos de peticiones también pueden ser utilizadas (como PUT y DELETE), sin embargo pueden no estar soportados por todos los navegadores.
- url: Establece la URL en donde se realiza la petición. La opción url es obligatoria para el método \$.ajax;

Capítulo 14. AJAX USANDO JQUERY – PARTE 2

14.1.- Métodos Convenientes/abreviados

Son los llamados métodos de alto nivel, ya que son macro instrucciones que suelen tener pocas opciones de configuración y son de muy fácil uso.

En caso que no quiera utilizar el método `$.ajax`, y no necesite los controladores de errores, estos métodos son más convenientes para realizar peticiones Ajax (aunque, como se indicó antes, estos están basados en el método `$.ajax` con valores pre establecidos de configuración).

Los métodos que provee la biblioteca son:

- `$(selector).load(url)` obtiene el código HTML de una URL y rellena a los elementos seleccionados con la información obtenida.
- `$.get` Realiza una petición GET a una URL provista.
- `$.post` Realiza una petición POST a una URL provista.
- `$.getScript` Añade un script a la página.
- `$.getJSON` Realiza una petición GET a una URL provista y espera que un dato JSON sea devuelto.

Los métodos deben tener los siguientes argumentos, en orden:

- url: La URL en donde se realizará la petición. Su valor es obligatorio.
- data: La información que se enviará al servidor. Su valor es opcional y puede ser tanto un objeto como una cadena de datos (como `foo=bar&baz=bim`).
- Nota: esta opción no es válida para el método `$.getScript`.
- success callback: Una función opcional que se ejecuta en caso que la petición haya sido satisfactoria. Dicha función recibe como argumentos la información de la petición y el objeto en bruto de dicha petición.

- data type: El tipo de dato que se espera recibir desde el servidor. Su valor es opcional. Nota: esta opción es solo aplicable para métodos en que no está especificado el tipo de dato en el nombre del mismo método.

14.2.- Función Load - \$.fn.load

El método `$.fn.load` es el único que se puede llamar desde una selección. Dicho método obtiene el código HTML de una URL y rellena a los elementos seleccionados con la información obtenida. En conjunto con la URL, es posible especificar opcionalmente un selector, el cual obtendrá el código especificado en dicha selección.

```
//Utilizar el método $.fn.load para llenar un elemento
$('#newContent').load('/foo.html');

//Utilizar el método $.fn.load para llenar un elemento basado en un selector
$('#newContent').load('/foo.html #myDiv h1:first', function(html) {
    alert('Contenido actualizado');
});
```

14.3.- Funciones \$.get() y \$.post()

Son dos métodos muy parecidos y hacen prácticamente lo mismo: una conexión AJAX enviando datos al servidor. La única diferencia es que `$.get()` envía los datos por GET (en la URL) y `$.post()` los envía por POST (en las cabeceras del HTTP).

A estos métodos les enviamos un juego de parámetros un poco más complejo, aunque solamente es obligatorio el primero de ellos.

```
/*
Esto llamaría a la URL "url.php" pasando por método GET los datos y valores indicados en el segundo parámetro.
Cuando se reciba la respuesta se ejecutará la función indicada en el tercer parámetro y el dato que se espera
recibir del servidor como respuesta tiene el formato indicado en el cuarto parámetro, existiendo varios formatos
de respuesta, como JSON, script, HTML o XML.
*/
$.get("url.php", {dato: "valor", dato2: "valor2"}, function(respuesta){
    $('#resultado').html( data );
}, "html");
```

4.4.- Funciones \$.Getscript () y \$.Getjson ()

\$.getScript(): Se trata de otro método "shorthand" (abreviado, corto) , interesante por ser un atajo para cargar un script Javascript y ponerlo en ejecución cuando se haya recibido correctamente. Podemos configurar un envío de datos y este método lo hará por GET, igual que se enviaban en \$.get(). Podemos decir, en resumen, que se usa igual que \$.get(), pero la respuesta del servidor estás diciéndole que se recibirá en un script Javascript que se debe ejecutar según se tenga.

```
// añade un script a la página y luego ejecuta la función especificada
$.getScript('/static/js/myScript.js', function() {
    functionFromMyScript();
});
```

\$.getJSON(): Es otro método abreviado bastante utilizado que recibe una respuesta en JSON. Es exactamente igual que lo comentado con "getScript()", solo que en la respuesta recibiremos un objeto Javascript. El contenido del objeto nos lo debe enviar el servidor como respuesta, siempre en un JSON que jQuery de manera interna se encargará de interpretar y convertir en un objeto nativo de Javascript, al que podremos acceder para recuperar cualquiera de sus datos.

```
ACADEMIA DE SOFTWARE
// obtiene información en formato JSON desde el servidor
$.getJSON('/details.php', function(resp) {
    $.each(resp, function(k, v) {
        console.log(k + ' : ' + v);
    });
});
```

Capítulo 15. AJAX USANDO JQUERY – PARTE 3

15.1.- Ajax y Formularios

Las capacidades de jQuery con Ajax pueden ser especialmente útiles para el trabajo con formularios. Por ejemplo, la extensión jQuery Form Plugin es una extensión para añadir capacidades Ajax a formularios.

Existen dos métodos que debe conocer para cuando esté realizando este tipo de trabajos: `$.fn.serialize` y `$.fn.serializeArray`.

`$.fn.serialize`: Transformar la información de un formulario a una cadena de datos

Ejemplo: `$("#myForm").serialize();`

`$.fn.serializeArray`: Crear un array de objetos contenido información de un formulario

Ejemplo: `$("#myForm").serializeArray();`

// crea una estructura como esta:

[ACADEMIA DE SOFTWARE

```
{ name : "field1", value : 123 },  
  
{ name : "field2", value : "hello world" }  
  
]
```

La respuesta a este método `serialize()` la puedes usar directamente como datos que envias al servidor al hacer una llamada AJAX.

```
//La respuesta a este método serialize() la puedes usar directamente  
//como datos que envias al servidor al hacer una llamada AJAX.  
$.post(url, $( "#formulario" ).serialize(), FuncionExito);
```

15.2.- Ejemplo de Formularios Con Ajax

En ocasiones queremos hacer que nuestros formularios se procesen de manera dinámica sin necesidad de recargar la página, luego de enviar los datos podemos mostrar un mensaje de éxito o fracaso con algunos efectos como transiciones, desapareciendo, moviéndose.

Veamos el siguiente formulario:

```
<!DOCTYPE html>
<html>
<head>
<title>Formulario con Ajax</title>
<script src="//ajax.googleapis.com/ajax/libs/jquery/1.8.3/jquery.min.js"></script>
</head>
<body>
<div id="formulario">
<form method="post" id="formdata">
<label for="nombre">Nombre: </label><input type="text" name="nombre" id="nombre" required="required"><br>
<label for="apellidos">Apellidos: </label><input type="text" name="apellidos" id="apellidos" required="required"><br>
<label for="direccion">Dirección: </label><input type="text" name="direccion" id="direccion" required="required"><br>
Género: <input type="radio" name="genero" id="hombre" checked="checked"><label for="hombre"> Hombre -<br>
<input type="radio" name="genero" id="mujer"><label for="mujer"> Mujer</label>
<label for="mayor">Es mayor de 18 años: </label><input type="checkbox" name="mayor" id="mayor" required="required">
<input type="button" id="botonenviar" value="Enviar">
</form>
</div>
<div id="exito" style="display:none">
    Sus datos han sido recibidos con éxito.
</div>
<div id="fracaso" style="display:none">
    Se ha producido un error durante el envío de datos.
</div>
</body>
</html>
```

ACADEMIA DE SOFTWARE

`$(document).ready` se llama automáticamente cuando la web (document) está lista (ready).

`$("#botonenviar").click` asocia la función que contiene como acción para el botón Enviar del formulario.

Seguidamente, valida el formulario, y si este devuelve true, se llama a la función `$.post`:

`$.post` es la función que enviará, vía `$_POST`, la información a la página “enviar.php” para que ser procesada. “enviar.php”, ya lo hemos dicho, es la página que procesa la información que recibe.

`$("#formdata").serialize()` prepara los datos para que la página “enviar.php” los reciba como valores de `$_POST`.

Finalmente, abrimos una función para ejecutar cuando el proceso de datos termine, con el parámetro “res” que viene a ser el resultado.

En nuestro caso, el resultado va a ser un 1 (éxito) o un 0 (fallo).

```
<script>
$(document).ready( function() { // Esta parte del código se ejecutará automáticamente cuando la página esté lista.
    $("#botonenviar").click( function() { // Con esto establecemos la acción por defecto de nuestro botón de enviar.
        if(validaForm()){
            // Primero validará el formulario.
            $.post("enviar.php",$("#frmdata").serialize(),function(res){
                $("#formulario").fadeOut("slow"); // Hacemos desaparecer el div "formulario" con un efecto fadeOut lento.
                if(res == 1){
                    $("#exito").delay(500).fadeIn("slow");// Si hemos tenido éxito, hacemos aparecer el div "exito"
                    // con un efecto fadeIn lento tras un delay de 0,5 segundos.
                } else {
                    $("#fracaso").delay(500).fadeIn("slow");// Si no, lo mismo, pero haremos aparecer el div "fracaso"
                }
            });
        }
    });
});
</script>
```

15.3.- Eventos Globales en Ajax

A menudo, querrás ejecutar una función cuando una petición haya comenzado o terminado, como por ejemplo, mostrar o ocultar un indicador. En lugar de definir estas funciones dentro de cada petición, jQuery provee la posibilidad de vincular eventos Ajax a elementos seleccionados. Para una lista completa de eventos Ajax, puede consultar docs.jquery.com/Ajax_Events.

```
//Mostrar/Ocultar un indicador utilizando Eventos Ajax
$('#loading_indicator')
    .ajaxStart(function() { $(this).show(); })
    .ajaxStop(function() { $(this).hide(); });
```

Existen eventos para cuando se produce una solicitud, como para cuando se produce un error, éxito en la respuesta, etc. Estos eventos los añades al objeto jQuery del documento extendido.

```
$( document ).AJAXError(function() {
    alert("error en una solicitud AJAX (no me preguntes en cuál)");
});
```