[Resumen] jQuery Mobile y JavaScript Avanzado

jQuery Mobile

es un proyecto que nace después de jquery ui con la intención de hacer algo como ui pero para móviles, una cantidad de widgets preparadas para telefonos, se puede en el computador pero es mas tedioso su uso, pues esta hecho para móviles.

Compatibilidad full

Utilizamos su CDN o puedes descargar todo aquí.

en html5 tenemos la capacidad de tener atributos propios que empiecen con la palabra data será totalmente compatible con html5 y no generará errores pero si lo colocamos sin el data nos generara errores.

```
<div data-nombre='pablito' data-apellido='rigazzi'></div>
```

Y esto es lo que utiliza jquery mobile pero con el atributo data-role en donde según lo que colocamos le asignará funciones y estilos determinados.

El primer data-role que define jquery mobile será el de page, la pagina la cual no da ningún atributo pero representa una pagina dentro de un documento, podemos tener muchos data-role='page' y con java script interactuar con el mismo sin generar errores.

una pagina normal tiene 3 partes un data-role header, navbar, content y

footer de la siguiente forma:

```
<div data-role='page'>
    <div data-role='header'>
        <h1>Home</h1>
        <div data-role="navbar">
            <ul>
            </ul>
        </div>
    </div>
    <div data-role='content'>
        <h1>Bienvenido a Mejorando el jQueryMobile</h1>
    </div>
    <div data-role='footer'>
    </div>
</div>
```

dentro de footer tendremos 3 data-role con el rol de button más un data-icon con una imagen que nos guste así:

```
<a href="#" data-role="button" data-icon="plus">Agregar</a>
<a href="#" data-role="button" data-icon="arrow-u">Subir</a>
<a href="#" data-role="button" data-icon="arrow-d">Bajar</a>
```

y el en navbar colocamos los botones con un li en donde el data-href que es el que indicara a donde navegará ese botón según el las páginas definidas con el data-role:

```
<li><a href="#" data-href="#home" class="ui-btn-active">Home</a></li>
<li><a href="#" data-href="#buscar" class="">Buscar</a></li>
<li><a href="#" data-href="#grito" class="">Grito</a></li>
```

la clase ui-btn-active sirve para que se marque el botón activo según la pagina que se esté viendo.

Debemos de meter todas nuestras páginas móviles dentro del mismo archivo y luego rotar entre estas.

Copiamos y pegamos el código que ya tenemos y le colocamos el id que

los represente según el botón de navegación, por ejemplo en la primera el id será home, el segundo será buscar y el tercero sera grito quedando así:

```
<div data-role='page' id='home'>
    <div data-role='header'>
        <h1>Home</h1>
        <div data-role="navbar">
            <ul>
                <li><a href="#" data-href="#home" class="ui-btn-
active">Home</a></li>
                <li><a href="#" data-href="#buscar"
class="">Buscar</a></li>
                <li><a href="#" data-href="#grito"
class="">Grito</a></li>
            </ul>
         </div>
    </div>
    <div data-role='content'>
        <h1>Bienvenido a Mejorando el jQueryMobile</h1>
    </div>
    <div data-role='footer'>
    </div>
</div>
<div data-role='page' id='buscar'>
    <div data-role='header'>
        <h1>Buscar</h1>
        <div data-role="navbar">
            <ul>
                <li><a href="#" data-href="#home" class="ui-btn-
active">Home</a></li>
                <li><a href="#" data-href="#buscar"
class="">Buscar</a></li>
                <li><a href="#" data-href="#grito"
class="">Grito</a></li>
            </ul>
         </div>
    </div>
    <div data-role='content'>
        <h1>Buscar</h1>
    </div>
    <div data-role='footer'>
    </div>
</div><div data-role='page' id='grito'>
    <div data-role='header'>
        <h1>Grito</h1>
        <div data-role="navbar">
            <ul>
                <li><a href="#" data-href="#home" class="ui-btn-
active">Home</a></li>
                <li><a href="#" data-href="#buscar"
class="">Buscar</a></li>
                <li><a href="#" data-href="#grito"
```

```
class="">Grito</a></li>
            </ul>
        </div>
    </div>
    <div data-role='content'>
        <h1>Grito</h1>
    </div>
    <div data-role='footer'>
    </div>
</div>
```

Si debemos cambiar la clase en el navbar según en donde estamos.

Ahora creamos un script después de haber llamado a jquery, en donde colocamos:

```
$('[data-role=navbar] a').on('click', function(){
    $.mobile.changePage($(this).attr('data-href'));
});
```

en donde va al data-role =navbar de cada página y con el objeto mobile que viene incluido con jquery mobile coge el elemento al que se le dio click y según su atributo en data-href cambia por el id de la página.

En la pagina de buscar, en el content crearemos unos elementos, entonces crearemos un fieldset con una clase ui-grid-a que será el contenedor de nuestro formulario con dos divs uno con clase ui-block-a y el segundo ui-block-b en donde se crearan dos columnas una al lado del otro.

En el div con id ui-block-a tendremos un input con type text y id cajatexto y en el dic con class ui-block-b crearemos un a con data-role button y un href con #, un data-icon check y un data-inline true y quedará de la siguiente forma:

```
<fieldset class="ui-grid-a">
    <div class="ui-block-a">
        <input type="text" id="cajatexto">
```

```
    </div>
    <div class="ui-block-b">
        <a      id="botonbuscar"
            href="#"
            data-role="button"
            data-icon="check"
            data-inline="true">Buscar</a>
    </div>
</fieldset>
```

el atributo data-inline true pasará a ocupar sólo su tamaño requerido y no ocupar el total del campo dado.

Cuando queremos ejecutar un código cuando se muestre nuestra page, como no podemos utilizar el .ready porque todas mis paginas están en el mismo documento usamos el evento pageshow sobre el id de la página a la cual se aplicará este código de la siguiente forma:

```
$('#grito').on('pageshow', function(){
    alert('Ay!');
});
```

Si damos click en grito aparecerá el alert, pero solo pasará cuando grito esté activo y no cuando cargue la página.

Ahora vamos a la pagina de buscar y creamos un div con id mensajes y crearemos un ul con data-role listview, id listado y 5 items en donde veremos aplicados estilos bonitos para las listas.

Si colocamos un data-role list-divider a un item se convertirá en un divisor de la lista, si le colocamos un a un item aplicará todos los estilos de un enlace con flechita y todo, si colocamos un span con clase ui-li-count con un número nos aparecerá una burbujita con el número y por ultimo si le colocamos un data-inset true al ul nos mostrará bordes redondeados automágicamente.

```
<ul id="listado"
    data-role="listview"
    data-inset="true">
```

```
    <li>Item 1</li>
    <li><a href="#">Item 2</a></li>
    <li>Item 3 <span class="ui-li-count">87</span></li>
    <li data-role="list-divider">Item 4</li>
    <li>Item 5</li>
</ul>
```

Ya teniendo esto listo podemos ver como crear una pagina para móviles muy bonita y con solo html y muy poco JavaScript.

Ahora veamos como podemos aplicarle JavaScript para hacerla mucho más funcional.

Primero nos tenemos que asegurar que el input tenga un id en este caso cajatexto y el boton de búsqueda también tenga un id botonbuscar creamos una función en donde decimos que al dar click al botonbuscar se ejecute una función.

En la funcion cojemos el valor que trael la cajatexto y le agregamos una clase ui-disabled al botonbuscar para que mientras se esté buscando se deshabilita el botón y en el id mensajes colocaremos que esta buscando el valor de caja texto en internet.

creamos un setTimeout que ejecutará una función a los 2000 segundos simulando una búsqueda en donde pondremos una variable con un array de resultados random y suponiendo que ya llegaron los resultados y en mensajes colocaremos que se encontraron x número de resultados sobre la palabra tipeada, luego eliminamos la clase ui-disabled para que se habilite de nuevo el botón.

Ahora vaciamos el ul con el data-role listview e imprimimos el resultado, esto se hace con el siguiente código.

```
$('#botonbuscar').on('click', function(){
    var buscar = $('#cajatexto').val();
    $('#botonbuscar').addClass('ui-disabled');
    $('#mensajes').html('Buscando '+buscar+' en Internet');
    setTimeout(function(){
```

```
        var res = [
            'Javascript Avanzado',
            'Aprendiendo Javascript',
            'Closures for Dummies',
            '¿Qué es un Mixin, mami?',
            'RFTM: The Good Parts',
        ];
        $('#mensajes').html('Se encontraron '+res.length+' resultados
sobre '+buscar);
        $('#botonbuscar').removeClass('ui-disabled');
        $('[data-role=listview]').html('');
        $.each(res, function(i, elem){
            $('#listado').append('<li><a href="#">'+elem+'</a></li>');
        });
        $('#listado').listview('refresh');

    }, 2000);
});
```

Recuerda que la función $.each recorre la variable res uno por uno y le asigna el valor a la variable elem.

```
$('#listado').listview('refresh');
```

Esta línea nos sirve para aplicar de nuevo los estilos de jQuery mobile y que nuestro listado sea nuevamente bonitos.

Esto en una forma resumida podemos ver cómo podemos manejar jQuery mobile con muy poco javascript y una interfaz muy funcional.

Tenemos otras opciones como sencha touch quien se diferencia por estar embebido con una serie mayor de aplicaciones, con este en realidad podemos hacer lo mismo que jquery mobile pero es mucho mas enterprise compilando a IOS Android o Blackberry (si todavía se usa), sus archivos de ejemplos pesan demasiado y es mucho más complejo llegar a el, es mucho más preparada al momento de hacer aplicaciones web y compilarla a sistemas operativos y requiere más trabajo que jQuery mobile.

Recuerda esto es según tu gusto.
Ahora iremos con un poco mas de teoría que te ayudará a repasar lo

que nos dijo @prigazzi.

Closures

El amigo incomprendido de javascript.

Primero debemos hablar de funciones quienes junto a los objetos son los más importantes de javascript ya que una función proveen un contexto y al crear una de esta ya estamos generando un closures, es decir, tenemos el contexto global y al crear una función creamos un nuevo contexto con las variables externas y las que tengamos dentro de ésta mas no a contextos fuera de estas, y si creamos una función dentro de esta función, estaremos creando un nuevo contexto con las variables de la función anterior a esta y luego al contexto global, es decir que si tengo una variable en el contexto global y lo cambiamos en la primera función, la última función cogerá el valor del contexto más cercano.

JavaScript utiliza un contexto léxico. Utilizando el valor de las variables al momento de definir la función/método.

Es decir que si esta en un contexto y llamó a una variable, irá escalando entre contextos hasta encontrar la variable o tirar error en el caso más extremo.

Entonces un Closure es una función que es evaluada a través del contexto léxico en la que fue definida, junto a las variables libres asociadas, es decir, una declaración de función más el contexto en el que fue declarada.

Usando Closures

Fabricar funciones

Ocultar Código
Definir Módulos
Extender el lenguaje

FABRICANDO FUNCIONES

Creamos una función que retorna una función y de esta forma generaremos funciones que cambian en una forma muy pequeña pero que será muy funcional para no repetir código:

```
function resizer (size)
{
    return function() {
        document.body.style.fontSize = size+'px';
    }
}
var size14 = resizer(14);
var size16 = resizer(16);
var size18 = resizer(18);
```

en donde solo cambiamos el tamaño de la letra sin necesidad de escribir varias veces la misma función sino con una sola función.

OCULTAR CODIGO:

Tenemos el siguiente código:

```
var Contador = (function() {
    var _contador = 0;
    return {
        incrementar : function() {
            _contador++;
        },
        decrementar : function() {
            _contador--;
        },
        ver : function () {
            console.log(_contador);
        }
```

```
    }
})  ();
```

En donde tenemos una función anónima pero, contador no se le asigna una función por que si vemos tiene un paréntesis al inicio y al final cierra paréntesis y coloca otros dos paréntesis que lo que hace es que ejecuta la función y lo que estará en contador será el resultado mas no la función.

dentro de la función devolvemos un objeto que tiene funciones y que se le asignará a contador, tendremos 3 atributos con 3 funciones que ya tienen un closure es decir un contexto y que en este caso será la variable contador, el objeto será mi interfaz pública, es decir lo que quiero devolver a el público, y tendrá contexto de _contador y solo será para ellos mas no para los demás.

DEFINIR MÓDULOS:

El ejemplo anterior del contador será un módulo lógico para javascript en donde lo que estamos creando es 3 funciones dentro del módulo contador a lo que podemos llamar namespace y que no interfiere al nivel nombre con el resto del programa, lo que hacíamos de los paréntesis en contador y que quedamos como What?? son closure anónimo en donde se ejecuta la función pero no se le asigna a contador sino que se le asigna lo que devuelve la función.

Lo que nos permite definir módulos es manejar algo llamado mixins, ya que javascript no maneja la misma definición de herencia ya que javascript se maneja todo con objetos definidos e instanciados y si están instanciados están vivos y sacar de aqui objetos que me sirven.

MIXINS:

```
var Emerfencias = (function ($ , mod2) {
    function privado_alertar() {
        $('#alerta').html('Socorro');
        mod2.notificar();
}
return {
    alertar : privado_alertar
}
}) (jQuery, otroModulo);
```

Aqui lo que hacemos es pasar dos parámetros en nuestro closure anónimo en el que será jQuery y un módulo ya definido, al recibirlo ya se vuelve problema nuestro y lo recibimos según nosotros queramos en este caso $ y mod2 y solo se usarán acá adentro, tenemos una función que tomara ambos módulos y los usara y devolverá un objeto con un atributo alertar y como valor la función privado_alertar.

Emergencias a final de cuentas será un módulo en donde alertar ejecutará la función privado_alertar.

Este modulo puede estar recibiendo así mismo como parámetro y extender la funcionalidad de este módulo de la siguiente manera:

```
var Emergencia = (function (mod) {
    function llamar911 () {
        window.location.href = "http://911.com";
    }
    mod.llamar 911 = llamar911;
    return mod;
}
```

Entonces lo que devolverá este código será el módulo extendido y en caso de ser vacío se llenará con el módulo que se devuelve.

Así es complicado pero es un Mixin, leelo y re leerlo para entenderlo.

Diseño Modular de aplicaciones

Esto es lo que debe estar basado tu app o desarrollo web en este momento:

Utilizando lo que nosotros queramos usar, lo que debemos encargarnos de hacer es crear el núcleo de la aplicación que debemos crear o utilizar de terceros y que será todo lo que podemos hacer en nuestra app cosa que el día de mañana yo pueda cambiar jQuery por backbone, no deba cambiar o por lo menos no mucho.

Hacia arriba tendremos todo lo individual con funciones únicas a sí mismas basadas del núcleo, lo de abajo lo que usamos para que el núcleo ande.

Un módulo debe ser capaz de recibir objetos en este caso sandbox le debe dar la facilidad al módulo de hacer lo que debe hacer e

internamente usar el núcleo de la app para funcionar..

Necesito una manera de comunicar los módulos sin necesidad de depender entre si para no generar confusiones entre si, esta manera es el sandbox el cual recibió llamados o notificaciones de evento con las cuales procederá a ejecutar o no eventos del módulo y así mismo avisar sobre este.

Este es el concepto de una arquitectura de mensajes y de observación entre módulos independientes que esperan señales para entrar en acción.