Plugins

- Utilidad a disposición de los desarrolladores para ampliar las funcionalidades del framework. Para hacen cosas más complejas para resolver necesidades específicas, pero las de manera que puedan utilizarse en el futuro en cualquier parte y por cualquier web.
- No es más que una función que se añade al objeto jQuery, para que a partir de ese momento responda a nuevos métodos. Con los plugins podemos añadirle nuevas utilidades.
- Se crean asignando una función a la propiedad "fn" del objeto jQuery, se podrán utilizar en cualquier objeto jQuery.

```
jQuery.fn.desaparece = function() {///Plugin para hacer desaparecer elementos de la web.
        this.each(function(){
            elem = $(this);
            elem.css("display", "none");
        });
        return this;
};
$("h1").desaparece(); ///asi se llama
```

Reglas para el desarrollo de plugins

1. El archivo con el código de plugin se nombrara como jquery.[nombre del plugin].js.
2. Añadir las funciones como nuevos métodos por medio de la propiedad fn del objeto jQuery, para que se conviertan en métodos del propio objeto jQuery.
3. Dentro de los métodos, "this" será una referencia al objeto jQuery que recibe el método. utilizar "this" para acceder a cualquier propiedad del elemento de la página con el que se esta trabajando.
4. Colocar un punto y coma ";" al final de cada método del plugin, para poder comprimir el codigo. Hay que colocarlo después de cerrar la llave del código de la función.
5. El método debe retornar el objeto jQuery sobre el que se solicitó la ejecución del plugin.return this;.
6. Se debe usar this.each para iterar sobre todo el conjunto de elementos que puede haber seleccionados.
7. Asignar el plugin al objeto jQuery, en vez de ha $, así se descartan problemas si dos plugin tienen el mismo nombre.

Ejemplo de un plugin

- hacer que los elementos de la página parpadeen,que desaparezcan y vuelvan a aparecer en un breve instante. con fadeOut() desapareceran y con fadeIn() apareceran.jQuery

```
jQuery.fn.parpadea = function() {
        this.each(function(){
                elem = $(this);
                elem.fadeOut(250, function(){
                        $(this).fadeIn(250);
                });
        });
        return this;
};
```

- Con this.each creamos un bucle para cada elemento que pueda haberse seleccionado. Con elem=$ (this) conseguimos extender a this con todas las funcionalidades del framework y el objeto jQuery resultante guardarlo en una variable. Invocamos fadeOut(), enviando como parámetro los milisegundos que durará el efecto. Luego enviamos una nueva función callback para ejecutar un fadeIn()

Ejemplo de un plugin

- invocar este plugin:

```
$(document).ready(function(){ //parpadean los elementos de clase "parpadear"
      $(".parpadear").parpadea();
})

//evento clic para un botón. Al pulsar parpadeán los de clase parpadear
$("#botonparpadear").click(function(){
      $(".parpadear").parpadea();
})
```

-

-

Ejercicio

- Ejercicio: Plugin jQuery, textarea con cuenta de caracteres

  Queremos que en los textareas aparezca información al lado con el número de caracteres que hay escritos dentro. Debe mostrarse nada más cargarse la página y actualizarse cuando se escriba algo dentro.

- En el plugin tenemos que hacer varias cosas.

  1. Un bucle con each para recorrer todos los objetos.

  2. Dentro del bucle podemos iterar con los elementos. Creamos un DIV y lo inicializamos con la cuenta de caracteres del textarea. Ese elemento lo añadiremos al cuerpo de la página, justo después de la etiqueta del textarea. (v_textarea.after(v_contador);)

  3. Se hara un evento, para cuando el usuario escriba en el textarea, la cuenta de caracteres se actualice automáticamente. (con keyup)

  4. El numero de caracteres se puede guardar en cada textarea con el metodo data() ya que guarda variables dentro del objeto jQuery (v_textarea.data(”contador”, v_contador);)

Gestión de opciones en plugins

- Cargar opciones para configurar el comportamiento de los plugins. Las recibirá el plugin como parámetro, hay que definir cuáles van a ser esas opciones de configuración y qué valores tendrán por defecto por medio de un objeto de "options", con las opciones se consigue que los plugins sean más configurables y por lo tanto más versátiles.
- Se hace al dar de alta el plugin, indicando una serie de datos con notación de objeto:

```
$("#capa").crearCaja({
      titulo: "titulo",
      anchura: 400,
      altura: 200,
      ...
});
```

- Se colocará en el código fuente un objeto con las variables de configuración y sus valores por defecto. Luego, cuando se cree el plugin, se mezclará con el objeto de options enviado por parámetro, con una única sentencia.

Gestión de opciones en plugins

```
jQuery.fn.miPlugin = function(cualquierCosa, opciones) { //opciones por defecto
        var configuracion = {
            dato1: "lo que sea",
            dato2: 78
         }
        jQuery.extend(configuracion, opciones);//combinar
         //resto del plugin
}
```

- Recibe dos parámetros, "cualquierCosa" y "opciones". La configuración, se ha recibido en el parámetro "opciones".
- Se define una variable "configuracion" con las opciones de configuración por defecto y se mezclan con las recibidas. Se puede acceder por medio de "configuracion" a todas las opciones.

```
$("#elemento).miPlugin({ //invocar al plugin pasando opciones
                dato1: "Hola amigos!",
                dato2: true  });
$("<div></div>").miPlugin({ //enviar sólo algun dato, el resto por defecto:
        dato2: 2.05  });
$("p").miPlugin(); //No enviar ningún dato todo por defecto.
```

Funciones y variables en plugins

```
jQuery.fn.miPlugin = function() {
    mivariableComun = "comun"; //variables comunes a todos los elementos del plugin
    this.each(function(){   //CÓDIGO DEL PLUGIN
       elem = $(this); //Elemento sobre el que estoy aplicando el plugin
            //variables específicas
        var miVariable = "x"; //Variable accesible en todo el código
            //funcion accesible en cualquier parte del plugin
        function miFuncion(){
            miVariable = elem.text();
            alert("mi variable local y particular de cada plugin: "+miVariable); //Contenido
           mivariableComun="Otra comun!";//cambio mivariableComun en todos los elementos
         }
        miFuncion();//puedo invocar las funciones del plugin
           //evento, que tiene una función. Puedo acceder a variables y funciones del plugin
        elem.click(function(){
            //puedo acceder a variables del plugin
        alert("Dentro del evento: " + miVariable);
           //puedo acceder a funciones
        miFuncion();
     }); //funcion
  }; //each
});//plugin
```

Exercici Plugin Amb Opcions

Exercici.

- Plugin para mostrar un tip con un mensaje que aparecería en una capa al pasar el ratón sobre un elemento caliente.
- opciones a implementar:
  - Velocidad de la animación de mostrar y ocultar el tip
  - Animación a utilizar para mostrar el tip
  - Animación a utilizar para ocultar el tip
  - Clase CSS para la capa del tip

```
var configuracion = {
        velocidad: 500,
        animacionMuestra: {width: "show"},
        animacionOculta: {opacity: "hide"},
        claseTip: "tip"
}
```

Alias personal y ocultar código en plugins

- El $ se utiliza en otros frameworks y la web donde se usa el plugin lo utiliza, pueden ocurrir conflictos, por eso hay que utilizar un alias.
- En los plugins se pueden utilizar las variables o funciones con el nombre deseado. Pero esos nombres pueden ser usados en la página, generando conflictos también. Por eso no es mala idea ocultar el código y hacerlo local.
- Esto se consigue colocando el código dentro de una función que se invoca según se declara. A esa función se le puede enviar la variable "jQuery" con la funcionalidad del framework, Se recibirá como parámetro con cualquier alias:

```
(function($) {
    //Código del plugin las variables o funciones sólo serán visibles aquí
    var loquesea;
    function algo(){
    }
}) (jQuery); //la función se ejecuta instantáneamente
```

- En este caso se esta recibiendo la variable jQuery con el nombre $, pero se podría utilizar cualquier otro nombre para el alias a jQuery.