.andSelf()
Add the previous set of elements on the stack to the current set.

Also in: Deprecated > Deprecated 3.0 | Events > Event Handler Attachment
.bind()
Attach a handler to an event for the elements.

Also in: Deprecated > Deprecated 1.10 | Internals | Properties > Properties of jQuery Object Instances | Removed
.context
The DOM node context originally passed to jQuery(); if none was passed then context will likely be the document.

Also in: Deferred Object | Deprecated > Deprecated 1.7 | Removed
deferred.isRejected()
Determine whether a Deferred object has been rejected.

Also in: Deferred Object | Deprecated > Deprecated 1.7 | Removed
deferred.isResolved()
Determine whether a Deferred object has been resolved.

Also in: Deferred Object | Deprecated > Deprecated 1.8
deferred.pipe()
Utility method to filter and/or chain Deferreds.

Also in: Deprecated > Deprecated 3.0 | Events > Event Handler Attachment
.delegate()
Attach a handler to one or more events for all elements that match the selector, now or in the future, based on a specific set of root elements.

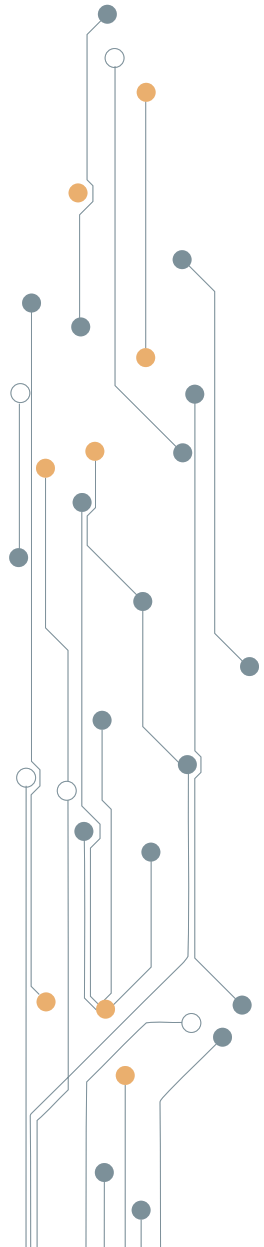
Also in: Deprecated > Deprecated 1.7 | Events > Event Handler Attachment | Removed
.die()
Remove event handlers previously attached using .live() from the elements.

Also in: Events > Browser Events | Deprecated > Deprecated 1.8 | Removed
.error()
Bind an event handler to the “error” JavaScript event.

Also in: Deprecated > Deprecated 1.3 | Removed | Utilities
jQuery.boxModel
States if the current page, in the user’s browser, is being rendered using the W3C CSS Box Model. This property was removed in jQuery 1.8. Please try to use feature detection instead.

Also in: Deprecated > Deprecated 1.3 | Properties > Properties of the Global jQuery Object | Removed | Utilities
jQuery.browser
Contains flags for the useragent, read from navigator.userAgent. This property was removed in jQuery 1.9 and is available only through the jQuery.migrate plugin. Please try to use feature detection instead.

Índice



1. Descarga y utilización

JQuery es una librería *JavaScript* que nos permite acceder a los objetos del *DOM* de manera simple creada por John Resig (Mozilla).

Como los desarrollos web son cada vez más complejos, con funcionalidades de autocompletar, drag & drop, validaciones y un largo etcétera; se necesitan algunas herramientas que no nos las hagan desarrollar desde cero. Esto son las librerías de terceros, entre ellas JQuery, que están hechas para simplificar y aclarar nuestro código JavaScript. Adicionalmente nos suelen resolver los problemas de códigos que funcionen de manera diferente en distintos navegadores.

Para utilizarla primeramente debemos descargar la librería del sitio oficial (https://jquery.org/)

Después, debemos incluirla en nuestras páginas web mediante esta línea:

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

Del sitio oficial de JQuery podemos descargar la versión descomprimida que ocupa alrededor de 60 Kb que es el archivo jquery.js que vemos en el atributo "src", sin embargo cuando la incluyamos en nuestra aplicación (y por motivos de peso) deberíamos descargar la versión comprimida que pesa 20 Kb.

1.1 | Ventajas de usar JQuery

- **Ahorra** muchas líneas de código.
- Transparenta el **soporte** de nuestra aplicación para los principales navegadores.
- Provee un mecanismo para **capturar los eventos.**
- Provee un conglomerado de **funciones para hacer animaciones** en la página de una manera muy simple.
- Integra funcionalidades para trabajar con **AJAX.**

1.2 | Maneras de desarrollar con JQuery

En las librerías de terceros (como en muchas otras materias) debemos adaptarnos a sus mecanismos de uso ya que intentar desarrollar como lo hacemos habitualmente podría hacer que nuestro código no tuviera el nivel que fuera de esperar. Lo adecuado sería ir viendo cual es la mecánica al trabajar con dicha librería mediante problemas sencillos e ir complicándolo a medida que entendemos la propia librería.

En las librerías de terceros (como en muchas otras materias) debemos adaptarnos a sus mecanismos de uso ya que intentar desarrollar como lo hacemos habitualmente podría hacer que nuestro código no tuviera el nivel que fuera de esperar. Lo adecuado sería ir viendo cual es la mecánica al trabajar con dicha librería mediante problemas sencillos e ir complicándolo a medida que entendemos la propia librería.

La función principal de JQuery se llama $.

A la función $ le podemos pasar distintos valores:

```
x=$(document);
...
x=$("#boton1");
```

En el primer caso le estamos pasando todo el documento (nuestra página) y en el segundo caso le estamos pasando el identificador (id) de un componente de nuestra página mediante un selector (como veremos en el apartado siguiente).

La función $ nos devuelve un objeto de tipo JQuery.

Vamos a ver este código de ejemplo que nos servirá de base para los siguientes:

```
var x;
x=$(document);
x.ready(iniEventos);
```

función

Inicialmente nos creamos una variable en JavaScript a la que llamamos x. La inicializamos una línea más tarde dándole el valor de todo el documento y, por tanto, la variable x contiene como valor nuestro documento. En la tercera línea llamamos al método ready de JQuery y le pasamos como parámetro el nombre de la función a la que queremos llamar. El método ready tiene una funcionalidad: cuando todos los elementos de nuestro documento se carguen en el navegador llamará a la función que le hemos pasado mediante parámetro.

El código de esta función:

```
function iniEventos (){var x; x=$("#b1");
x.click(presionB1); }
```

Usamos de nuevo la función $ para crear un objeto de la clase JQuery pero ahora asociándolo a un botón que se encuentra en el documento y que tiene el identificador (id) de b1. Por último llamamos al método click pasando como parámetro el nombre de la función que se ejecutará al presionar dicho botón.

Este ha sido un ejemplo simplificado de la utilización de JQuery.

2. Selectores

2.1 | Maneras de desarrollar con JQuery

SELECTOR POR ID

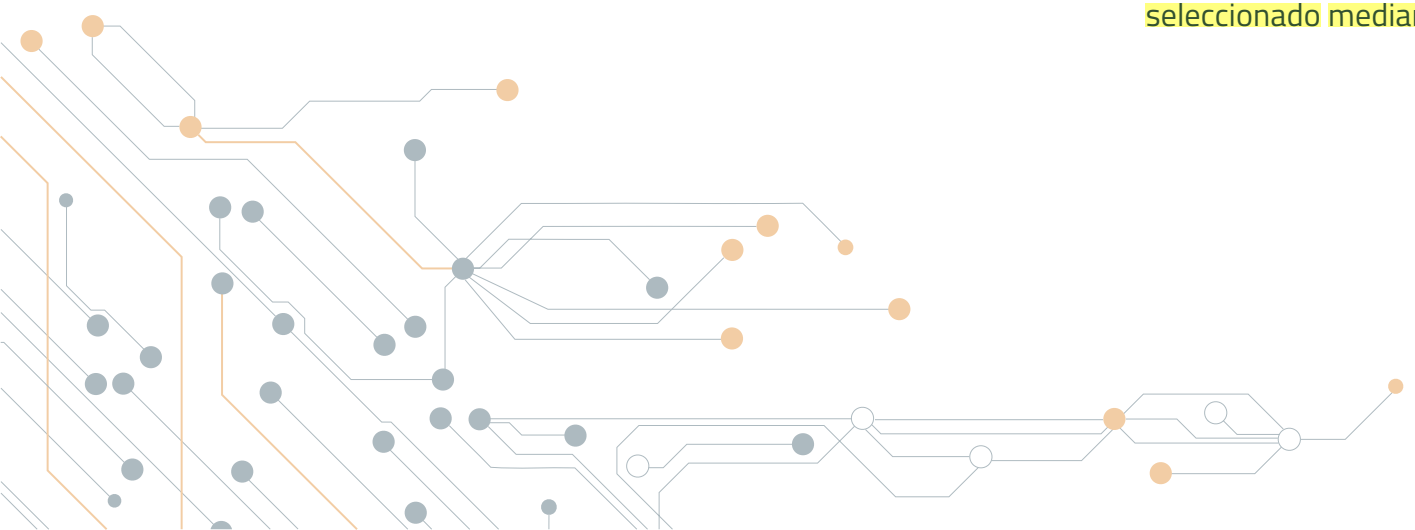
El primer selector que nos encontramos es aquel que selecciona un componente dentro de nuestro mapa documental mediante su atributo "id".

Como sabemos, el atributo "id" que tiene un componente debería ser único dentro del documento (esto es que no hubiera otro componente con el mismo valor de id)

Para seleccionar unívocamente un componente se usa este tipo de selector y su sintaxis es la que sigue:

```
var a = $("#nombre_del_id");
```

Como se ve, hay que preceder al id del signo '#' y entrecomillar. Dentro de la variable a, por tanto, estará nuestro componente seleccionado mediante su id.



2.2 | Selector por tipo Elemento o etiqueta

El selector por tipo es aquel que selecciona por el nombre de la etiqueta (o tag), es decir, por el tipo de los componentes que se encuentran dentro de nuestro mapa documental.

En contra del selector visto anteriormente, este selector puede seleccionar más de un elemento por lo que devuelve un array de valores con todos los elementos seleccionados.

Su sintaxis es la que sigue:

```
var a = $("tipo");
```

Únicamente debemos ingresar en el string entrecomillado el tipo de los elementos que queremos seleccionar, como:

```
var a = $("div");
```

En este ejemplo, la variable a contendría un array con todos los div que hubiera en nuestro documento.

2.3 | Selector por clase

El selector por clase es un tipo de selector que selecciona a través del atributo class que tengan los componentes.

Así, con un mismo selector, podemos seleccionar componentes de diferentes tipos solo con que compartieran la misma clase.

Como ocurre con el selector por tipo, este tipo de selector puede seleccionar más de un componente a la vez con lo que nos devolverá un array de elementos.

Su sintaxis es la que sigue:

```
x=$(".nombre_de_clase");
```

Usamos el signo "." al inicio del nombre de nuestro estilo para indicar a la librería que estamos seleccionando por clase. Como hemos dicho diferentes tipos de componentes pueden tener la misma clase así que:

```
x=$(".redondeado");
```

Esta línea seleccionaría todos los elementos que tuvieran la clase redondeado.

```
<input class="redondeado" ... />
<div class="redondeado" ... />
```

Así que x contendría al final un array de dos posiciones, una para el input y otra para el div.

2.4 | Selectores anidados

Podemos anidar selectores con una separación mediante de esta manera:

```
var a = $("div .redondeado");
```

Con esta línea lo que seleccionaría nuestro selector sería un div (o varios) y dentro de ellos aquellos componentes que tuvieran la clase redondeado. La selección se lee de izquierda a derecha siendo las siguientes selecciones discriminantes de las anteriores. Se pueden poner tantas como si quieran mientras tengan sentido:

```
var a = $("#uno div .redondeado");
```

Como anotación podemos comentar que no tiene sentido poner el selector por id en el medio de una selección o al final porque se entiende que este tipo de selector es único y, por tanto, no se necesitaría discriminar las selecciones anteriores para llegar a él.

2.5 | Agrupación de selectores

Quizás queramos unir en una misma variable diferentes selecciones a la vez. JQuery nos ofrece una sintaxis simple para ello:

```
var a = $("#uno, div, .redondeado");
```

unico elemento

a = almacena un array

La separación por comas dentro de nuestra selección se entiende cuando, siguiendo el ejemplo, queremos seleccionar el componente cuyo id es uno, todos los componentes de tipo div y todos los componentes que tengan la clase redondeado. Todos ellos quedarían en el mismo array guardado como valor en la variable a.

3. Funciones principales

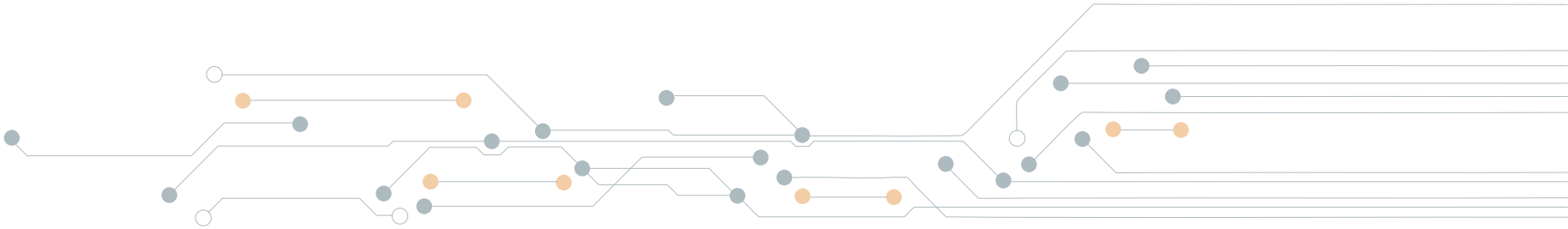
- **text(), text(valor)**

  La primera extrae el texto plano contenido en los elementos seleccionados.

  La segunda reemplaza ese texto plano con lo que hayamos pasado como valor.

- **attr(nombre de propiedad), attr(nombre de propiedad,valor) y removeAttr(nombre de propiedad)**

  La primera obtiene el valor de un atributo para nuestra selección, la segunda modifica el atributo con el nuevo valor dentro de nuestra selección y la tercera elimina el atributo de nuestra selección.



- **addClass(valor) y removeClass(valor)**

  El primero añade la clase que hemos pasado en valor a nuestra selección. El segundo, en cambio, la elimina.

- **html() y html(valor)**

  El primero obtiene el html contenido dentro de nuestra selección. El segundo modifica el html contenido en nuestra selección por el nuevo html que pasamos por valor.

- **show() y hide()**

  El primero muestra nuestra selección por el navegador, el segundo la oculta. Ambas admiten medidas de tiempo para que se realicen estas funciones (medidas de tiempo en milisegundos o mediante palabras reservadas como "slow" o "fast").

- **fadeIn() y fadeOut()**

  Estos dos métodos son muy parecidos a los dos anteriores. El primero muestra nuestra selección paulatinamente, el segundo la oculta. Producen un efecto mucho más bonito y suelen vincularse antes o después de un show o hide. Ambas admiten medidas de tiempo para que se realicen estas funciones (medidas de tiempo en milisegundos o mediante palabras reservadas como "slow" o "fast"). También existe el faceto que admite como parámetro un número de 0 a 1 con decimales siendo 1 nuestra selección completamente visible y 0 completamente invisible. Por lo que los números decimales comprendidos entre 0 y 1 darían a nuestra selección una apariencia transparente o fantasmal.

- **toggle**

  Este método, cada vez que se ejecuta, cambia nuestra selección alternativamente de visible a invisible, y de invisible a visible.

- **each**

  Es la iteración de los elementos de un array que nos implementa JQuery.

```
var x;
x=$([elementos]);
x.each([nombre de funcion])
```

La función se ejecutaría para cada uno de los elementos de nuestro array guardado en la variable x.

3.1 | Manipulación de los elementos del DOM

Hay algunas funciones en JQuery preparadas para manipular los elementos del DOM. Pasamos a listar algunas de ellas:

- **empty()**

  Vacía de contenido nuestra selección.

- **append(valor)**

  Añade un componente o un html que pasamos por parámetro al final de nuestra selección (justo antes de la etiqueta de cierre).

- **prepend(valor)**

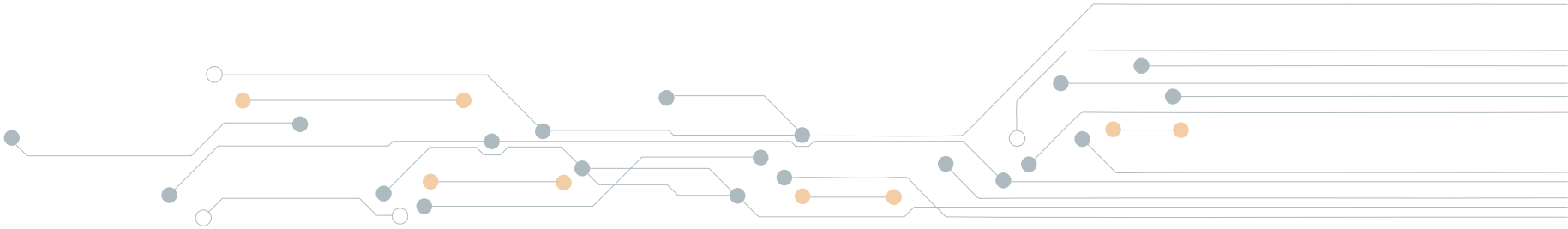
  Lo mismo que la anterior salvo que el componente o el html se añaden justo después del cierre de la etiqueta de apertura.

- **remove()**

  Elimina nuestra selección de nuestro mapa documental.

- **eq(valor)**

  Selecciona un componente de nuestra selección pasándole como valor la posición que ocupa en el array.



3.2 | Tratamiento de eventos con JQuery

Existen funciones que sirven para añadir escuchadores de eventos a nuestras selecciones, es decir, agregar funcionalidad a nuestras selecciones en el momento en el que se escuchen alguno de los eventos que hemos descrito.

- **click()**

  Escucha el evento click del ratón y se le puede pasar por parámetro la función que queramos ejecutar cuando se escuche ese evento.

- **dblclick()**

  Escucha el evento dobleclick del ratón y se le puede pasar por parámetro la función que queramos ejecutar cuando se escuche ese evento.

- **mouseover() y mouseout()**

  El primero se lanza cuando pasamos por encima de nuestra selección con el ratón y el segundo cuando salimos de ella, aparte se le puede pasar por parámetro a ambas la función que queramos ejecutar cuando se escuche ese evento.

- **hover(funcion1, funcion2)**

  Es la fusión de mouseover y mouseout. La funcion1 se ejecutará cuando se escuche el mouseover y la funcion2 cuando se escuche el mouseout.

- **mousemove()**

  Se ejecuta con cada movimiento del ratón en nuestra selección. Se le puede pasar por parámetro la función que queramos ejecutar cuando se escuche ese evento.

- **Mousedown() y mouseup()**

  El primero escucha cuando presionamos el botón del ratón (no en un click, sino en presión) y dura lo que la presión dura. El segundo es cuando liberamos la presión en el botón del ratón. A los dos se le puede pasar por parámetro la función que queramos ejecutar cuando se escuche ese evento.

- **focus() y blur()**

  El primero actúa cuando se gana el foco mientras el segundo actúa cuando se pierde. Se les puede pasar por parámetro la función que queramos ejecutar cuando se escuchen estos eventos.

4. AJAX con JQuery

Las utilidades y las funciones que están relacionadas con *AJAX* son parte principal de la librería *JQuery*. Tenemos un método para hacer peticiones *AJAX* que es *$.ajax()* (no hay que olvidar el punto entre *$ y ajax*).

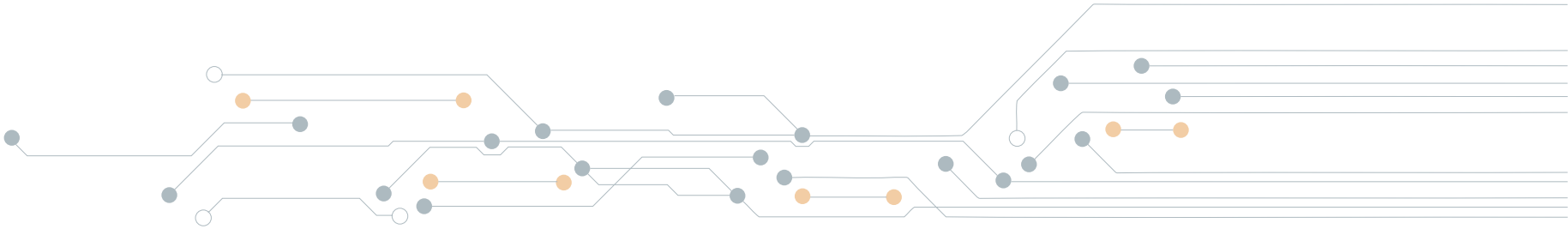
Aparte, se han descrito varias funciones que han partido de este método inicial que son: *$.get()*, *$.post()*, *$.load()*, etc. Estas están preparadas para tareas predeterminadas.

Sintaxis:

```
$.ajax(opciones);
```

La URL que se trata se incluye dentro del array de opciones.

```
$.ajax({
  url: '/ruta/hasta/pagina.php',
  type: 'POST',
  async: true,
  data: 'parametro1=valor1&parametro2=valor2',
  success: procesaRespuesta,
  error: muestraError
});
```



LA SIGUIENTE TABLA MUESTRA TODAS LAS OPCIONES QUE SE PUEDEN DEFINIR PARA EL MÉTODO $.AJAX():

Opción	Descripción
async	Indica si la petición es asíncrona. Su valor por defecto es true, el habitual para las peticiones AJAX
beforeSend	Permite indicar una función que modifique el objeto XMLHttpRequest antes de realizar la petición. El propio objeto XMLHttpRequest se pasa como único argumento de la función
complete	Permite establecer la función que se ejecuta cuando una petición se ha completado (y después de ejecutar, si se han establecido, las funciones de success o error). La función recibe el objeto XMLHttpRequest como primer parámetro y el resultado de la petición como segundo argumento
contentType	Indica el valor de la cabecera Content-Type utilizada para realizar la petición. Su valor por defecto es application/x-www-form-urlencoded
data	Información que se incluye en la petición. Se utiliza para enviar parámetros al servidor. Si es una cadena de texto, se envía tal cual, por lo que su formato debería ser parametro1=valor1&parametro2=valor2. También se puede indicar un array asociativo de pares clave/valor que se convierten automáticamente en una cadena tipo query string
dataType	El tipo de dato que se espera como respuesta. Si no se indica ningún valor, jQuery lo deduce a partir de las cabeceras de la respuesta. Los posibles valores son: xml (se devuelve un documento XML correspondiente al valor responseXML), html (devuelve directamente la respuesta del servidor mediante el valor responseText), script (se evalúa la respuesta como si fuera JavaScript y se devuelve el resultado) y json (se evalúa la respuesta como si fuera JSON y se devuelve el objeto JavaScript generado)
error	Indica la función que se ejecuta cuando se produce un error durante la petición. Esta función recibe el objeto XMLHttpRequest como primer parámetro, una cadena de texto indicando el error como segundo parámetro y un objeto con la excepción producida como tercer parámetro
ifModified	Permite considerar como correcta la petición solamente si la respuesta recibida es diferente de la anterior respuesta. Por defecto su valor es false
processData	Indica si se transforman los datos de la opción data para convertirlos en una cadena de texto. Si se indica un valor de false, no se realiza esta transformación automática
success	Permite establecer la función que se ejecuta cuando una petición se ha completado de forma correcta. La función recibe como primer parámetro los datos recibidos del servidor, previamente formateados según se especifique en la opción dataType
timeout	Indica el tiempo máximo, en milisegundos, que la petición espera la respuesta del servidor antes de anular la petición
type	El tipo de petición que se realiza. Su valor por defecto es GET, aunque también se puede utilizar el método POST
url	La URL del servidor a la que se realiza la petición

Aparte, existen varias funciones relacionadas que son versiones especializadas y simplificadas de esa función. Por ejemplo, *$.get()* y *$.post()* se usan para tratar peticiones *GET* y *POST:*

```
$.get('/ruta/hasta/pagina.php'); // Petición GET simple
// Petición GET con envío de parámetros y función que
procesa la respuesta
$.get('/ruta/pagina.php',  { producto: '34' },
  function(envio) { alert('Respuesta = '+envio);});
```

Las POST se escriben de la misma manera, únicamente cambiando *$.get()* por *$.post()*. Sintaxis:

```
$.post(url, datos, funcionManejadora);
```

El único parámetro que es obligatorio es el de 'url' donde indicamos a que dirección vamos a ir para realizar la comunicación. El segundo parámetro (datos) es lo que queramos enviar a esa dirección (habitualmente un objeto *JSON*) y el tercer parámetro (funcionManejadora) es una función JavaScript para el tratamiento de la devolución de los datos mediante la llamada de Ajax.

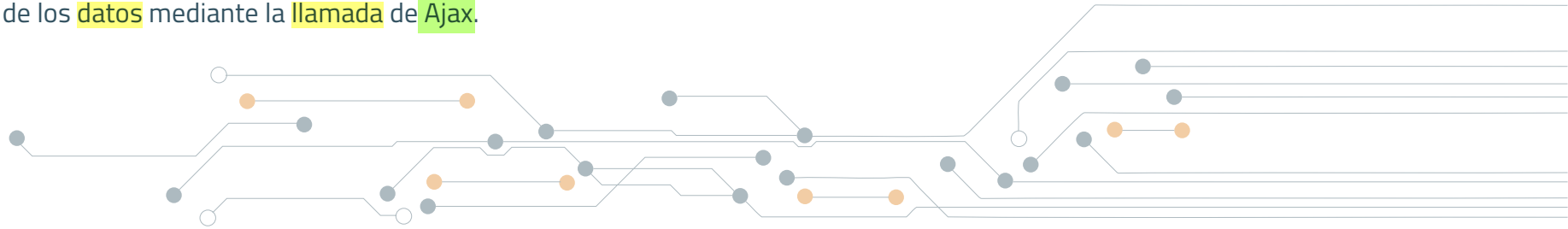
Existe una versión modificada del *$.get()* que se escribe *$.getIfModified()*, que mide si la respuesta es diferente a la anteriormente recibida.

La función *$.load()* agrega el contenido de la respuesta del servidor en el componente del documento que se indica.

```
<div id="info"></div>
$('#info').load('/ruta/hasta/pagina.php');
```

También dispone de una versión específica denominada *$.loadIfModified()* que carga la respuesta del servidor en el elemento sólo si esa respuesta es diferente a la última recibida.

Para finalizar, existen las funciones *$.getJSON()* y *$.getScript()* que cargan y ejecutan respectivamente una respuesta de tipo *JSON* y una respuesta con código *JavaScript.*



5. Funciones para CSS

La librería contiene funciones para manipular las propiedades CSS de los componentes de nuestro documento.

Una vez seleccionados un único elemento o diferentes elementos como nuestra selección podemos variar sus propiedades CSS de una manera simple con JQuery.

```
$('div').css('background');
$('div').css('color', '#000000');
$('div').css({ padding: '3px', color: '#CC0000' });// Establece varias
propiedades CSS
$('div').height();// Obtiene la altura en píxel del primer 'div' de la página
$('div').height('150px'); // Establece la altura en píxel de todos los 'div'
de la página
$('div').width();// Obtiene la anchura en píxel del primer 'div' de la página
$('div').width('300px');// Establece la anchura en píxel de todos los 'div'
de la página
```

Also in: Deprecated > Deprecated 1.7 | Internals | Properties > Properties of jQuery Object Instances | Removed

.selector

A selector representing selector passed to jQuery(), if any, when creating the original set.

Also in: Deprecated > Deprecated 1.8 | Miscellaneous > DOM Element Methods | Removed

.size()

Return the number of elements in the jQuery object.

Also in: Deprecated > Deprecated 1.8 | Events > Mouse Events | Removed

.toggle()

Bind two or more handlers to the matched elements, to be executed on alternate clicks.

Also in: Deprecated > Deprecated 3.0 | Events > Event Handler Attachment

.unbind()

Remove a previously-attached event handler from the elements.

Also in: Deprecated > Deprecated 3.0 | Events > Event Handler Attachment

.undelegate()

Remove a handler from the event for all elements which match the current selector, based upon a specific set of root elements.

Also in: Deprecated > Deprecated 1.8 | Events > Document Loading | Removed

.unload()

Bind an event handler to the “unload” JavaScript event.