jQuery

André Restivo

Index

Introduction

jQuery

jQuery is a JavaScript library that simplifies the way developers interact with the DOM:

- HTML document **traversal** and **manipulation**
- **Event** handling
- **Animation**
- **Ajax**
- Browser **compatibility**

Core API

- A lot of jQuery's functionality is based on the usage of a function/object called **jQuery**.
- Because javascript is **awesome**, this function can also be called **$**.
- This function returns a collection of **matched** elements either:
  - found in the DOM.
  - based on passed arguments.
  - created by passing an HTML string.

jQuery() or $()

$(selector, [context])	Select elements from the DOM. Context can be a DOM Element, Document or jQuery.
$(element)	Wrap a DOM element inside a jQuery object.
$(jQuery)	Clone a jQuery object.
$(html)	Creates DOM elements on the fly from the provided string of raw HTML.
$(callback)	Executes the callback function when the document is ready.

Always returns a **jQuery** function/object containing the selected elements for easy chaining.

Resources

- Reference:
    - jQuery API
- Tutorials:
    - jQuery Learning Center
    - Try jQuery (interactive tutorial)

Using jQuery

Using jQuery

The easiest way to import the jQuery library into an HTML document is to use the jQuery CDN (Content Delivery Network).

```
<script src="http://code.jquery.com/jquery-1.11.3.min.js"></script>
```

Alternatively, you can download the library and include it locally.

Waiting

Before using jQuery, one must wait for the document to finish loading. There are several ways of doing this:

```
function setUp() { /* code goes here */ };
$(document).ready(setUp);
```

```
$(document).ready( function() { /* code goes here */ } )
```

```
function setUp() { /* code goes here */ };
$(setUp);
```

```
$( function() { /* code goes here */ } );
```

Selection, Traversal and Filtering

Selectors

The jQuery function is capable of selecting elements from the DOM using CSS selectors as well as some non-standard selectors.

```
$("p") // Selects all paragraphs
$("#foo");                  // Element with id foo
$(".bar");                  // All elements with class bar
$("input[name='username']"); // Based on attributes
$("#contents ul.people li"); // Multiple selectors
```

Learn more: API Documentation: Selectors

Traversing

As the **jQuery** function returns a **jQuery** object, it is possible to easily traverse the DOM tree.

Each one of this functions allow an optional selector to further filter the results.

```
$("article").children();       // All children of articles
$("article").children("p");    // All paragraphs that are children of articles
$("article").find("p");        // All paragraphs that are descendant of articles

$("article").siblings();       // All article siblings
$("#menu").next();             // The next sibling of the menu.
$("#menu").nextAll();          // The next siblings of the menu
$("#menu").nextAll("section"); // The next siblings of the menu that are sections
// Also see prev() and prevAll()

$("input").parent("article");  // All articles that have input children
$("input").parents("article"); // All articles that have input descendants
```

Filtering

It is also possible to reduce the list of selected elements.

```
$("article").first();                 // The first article
$("article").last();                  // The last article
$("article").eq(2);                   // The third article
$("article").children().filter("p");  // All article children that are paragraphs
$("article").has("p");                // Articles that have paragraph descendants
$("article").not(".important");       // All articles that are not important
$("article").slice(1, 3);             // Articles 2 to 3
```

Manipulation

Changing classes

After selecting elements, it is possible to manipulate them in several ways.

```
$("article").addClass("selected");    // Add the class selected to all articles
$("article").removeClass("selected"); // Remove the class from all articles
$("article").toggleClass("selected"); // Toggle the class on all articles
$("article").hasClass("selected");    // True if any articles have class selected
```

Changing attributes

The attributes of elements can also be inspected and changed.

When retrieving attributes, only the value for the first element is returned.

```
$("#menu").children("a").attr("href");   // Get the href of the menu's first link
$("form.secure").attr("method", "post"); // Change all secure forms to post
$("a").removeAttr("href");               // Remove the href from all links
```

For boolean attributes use the `prop()` function.

```
$("input:checkbox").prop("checked")       // Checked property of the 1st checkbox
$("input:checkbox").prop("checked", true) // Checks all checkboxes
```

Manipulating the CSS

CSS properties can be manipulated and inspected individually

When retrieving properties, only the value for the first element is returned.

```
$("#menu").css("background-color");        // The background color of the menu
$("#menu").css("background-color", "red"); // Change the background of the menu

// To set several properties at the same time, just use an object
$("#menu").css{"background-color": "red", "padding": "1em"}
```

Changing the DOM tree

The DOM tree can also be manipulated by insert, removing, copying and moving elements.

```
$("article").empty();                    // Remove all children from articles
$("article").remove();                   // Remove all articles from the tree
$("article").before("<p>Hello</p>");     // Insert paragraph before every article
$("article").prepend("<p>Hello</p>");    // Insert paragraph as first child
$("article").after("<p>Hello</p>");      // Insert paragraph after every article
$("article").append("<p>Hello</p>");     // Insert paragraph as last child
$("article").replaceWith("<p>Hello</p>");// Replace articles with paragraphs
```

Elements can be retrieved from the DOM tree.

To insert a copy of the element, use the `clone()` function.

```
$("article").after($("form#comment"));     // Insert the comment form after
                                           // each article
$("article").after($("#comment").clone()); // Insert a copy after each article
```

Changing the DOM tree

It is easy to manipulate the parents of an element.

```
// Replace the parent of all articles with themselves
$("article").unwrap();

// Wrap all articles with a new parent div
$("article").wrap('<div class="content"></div>');
```

Content

The content of the document can also be retrieved and changed.

The `val()` function get and set the values of **inputs**, **selects** and **textareas**.

```
$("input#username").val();          // Get the value of the username input
$("input#username").val("johndoe"); // Change the value of the username input
```

The `text()` and `html()` functions can be used to get and manipulate the text and html contents of elements.

```
$("p").text();                // Get the combined text of all paragraphs
$("p").text("Hello world!");  // Set the text of all paragraphs
$("#menu").html("<ul></ul>"); // Set the html content of the menu
$(".important").html();       // Get the html content of the first important element
```

Size and Position

```
$("#menu").height();         // Menu height in pixels
$("#menu").height("1em");    // Change menu height in css units
$("#menu").height(10);       // Change menu height in pixels
$("#menu").width();          // Menu width in pixels

$("#menu").offset();                     // Menu position relative to document
$("#menu").offset({ top: 10, left: 30 });// Change position relative to document
$("#menu").position();                   // Menu position relative to parent

$("document").scrollLeft(); // The horizontal position of the scroll bar
$("document").scrollTop();  // The vertical position of the scroll bar

$("document").scrollTop($("document").height);
```

Events

Events

These methods are used to register behaviors to take effect when the user interacts with the browser.

Normally, we define an event as:

```
$(<selector>).<event>( handler );

function handler ([Event event]) {
  // code goes here
}
```

With `<selector>` replaced by a jQuery selector expression and `<event>` replaced by an event name.

Anonymous functions can also be used:

```
$(<selector>).<event>( function ([Event event]) {
  // code goes here

});
```

Events

Events can also be triggered by calling the event function without an associated handler.

```
$(<selector>).<event>();
```

For example, triggering a click on the first menu link:

```
$("#menu li:first-child a").click();
```

The Event Object

The event object, received by the event handler, contains properties about the event. Each event has a different set of properties.

The following properties are normalized by jQuery for cross-browser compatibility:

`target`, `relatedTarget`, `pageX`, `pageY`, `which`, `metaKey`

These ones are copied from the original event:

`altKey`, `bubbles`, `button`, `cancelable`, `charCode`, `clientX`, `clientY`, `ctrlKey`, `currentTarget`, `data`, `detail`, `eventPhase`, `offsetX`, `offsetY`, `originalTarget`, `prevValue`, `screenX`, `screenY`, `shiftKey`, `view`

Other, more specific properties, can be accessed as properties of the `event.originalEvent`.

Learn more about the Event Object

Document Loading

The `ready()` event, is fired when the document finishes loading.

Differs from Javascript's `load` event as it is fired as soon as the DOM tree is complete while the `load` event waits for all assets to be completely received.

```
$(document).ready(setUp);
function setUp() {
  // code goes here
}
```

This event has a simpler alternative syntax:

```
$(setUp);
function setUp() {
  // code goes here
}
```

Form Events

Events that are related to form and inputs:

```
$("input#username").focus(h);    // Username input got focus
$("form#register").focusin(h);   // Element or any descendant gained focus
$("input#username").blur(h);     // Username input lost focus
$("form#register").focusout(h); // Element or any descendant lost focus
```

```
$("input#username").change(h);   // Username input changed value
$("input#username").select(h);   // User selected text inside the username input
$("form#register").submit(h);    // User submitted form
```

Keyboard Events

Events fired when the keyboard is used.

```
$("input#username").keydown(h);  // Key went down when username input had focus
$("input#username").keyup(h);    // Key went up when username input had focus
```

The keypress event is similar to the keydown event, except that modifier and non-printing keys (such as shift, Esc, and delete) trigger keydown events but not keypress.

```
$("input#username").keypress(h); // Key was pressed
```

Mouse Events

```
$("h1").click(h);               // H1 has been clicked
$("#menu a").dblclick(h);       // Link in the menu has been double clicked

$("#menu a").mousedown(h);      // Mouse button has been pressed
$("#menu a").mouseup(h);        // Mouse button has been released
$("#menu a").toggle(h1, h2);    // Call h1 and h2 alternatively on button pressed

$("#menu").hover(h1, h2);       // Pointer entered (h1) and left (h2) the menu
$("#menu").mouseenter(h);       // Mouse pointer entered the menu
$("#menu").mouseleave(h);       // Mouse pointer left the menu
$("#menu").mousemove(h);        // Mouse pointer moved inside the menu
```

On mouse button events, we can use event.which to know which button has been pressed.

Event Handlers

The `on()` function, allows binding a function to an event passed by name.

```
$("h1").on("click", h);
```

It is also possible to bind a function to more than one event simultaneously.

```
$("a").on("mouseenter mouseleave", h);
```

When a selector is provided, the event handler is referred to as delegated. The handler is not called when the event occurs directly on the bound element, but only for descendants (inner elements) that match the selector.

```
$("table").on("click", "tr", h); // Bind function h to all rows inside a table
```

In this last example, the event will call function `h` even for table rows added in the future. But not for table rows of tables added in the future.

See also .off() and .one().

Effects

Standard Effects

jQuery has some builtin effects.

In all of them, a handler can be added that is called when the effect finishes.

```
("#menu").fadeIn(400, handler)     // Menu fades in, in 400ms
("#menu").fadeOut(400, handler)    // Menu fades out, in 400ms
("#menu").fadeTo(400, 50, handler) // Menu fades to 50% opacity (400ms)
("#menu").fadeToggle(400, handler) // Menu fades in or out, alternatively
```

These three methods use the width, height and opacity to show and hide elements.

```
("#menu").hide(400, handler)       // Menu shows, in 400ms
("#menu").show(400, handler)       // Menu hides, in 400ms
("#menu").toggle(400, handler)     // Menu shows and hides, alternatively
```

```
("#menu").slideDown(400, handler)    // Menu slides down, in 400ms
("#menu").slideUp(400, handler)      // Menu slides up, in 400ms
("#menu").slideToggle(400, handler)  // Menu slides up and down, alternatively
```

Delay and Options

A delay can be set when chaining effects:

```
$("#menu").fadeIn(400).delay(200).fadeOut(400, handler)
```

An object containing extra **options** can also be used instead of sending the duration and handler of the effect. Read more on Effects.

Animate

The `animate()` function performs a custom animation of a set of CSS properties.

```
("#menu").animate({
    width: "70%",
    opacity: 0.4,
    marginLeft: "0.6in",
    fontSize: "3em",
    borderWidth: "10px"
});
```

Ajax

Ajax

There are several functions that make using Ajax in jQuery very easy.

The `ajax()` function is the most configurable one:

```
$.ajax({
  type: "get",
  url: "getdata.php",
  data: {"id": 1, "name": name}
}).done(function(data) {
  // Request completed
}).fail(function() {
  // Request failed
});
```

Read more on the ajax() function.

Get

The `get()` function is a fast way to perform a get.

```
$.get( "getdata.php", doneHandler)
  .done(function(data) { // Alternative done handler
    alert( "done" );
  })
  .fail(function() {
    alert( "error" );
  })
  .always(function() {
    alert( "finished" );
  });
```

The same as:

```
$.ajax({
  url: url,
  data: data,
  success: success,
  dataType: dataType
});
```

Post

The `post()` function is a fast way to perform a post.

```
$.post( "getdata.php", doneHandler)
  .done(function(data) { // Alternative done handler
    alert( "done" );
  })
  .fail(function() {
    alert( "error" );
  })
  .always(function() {
    alert( "finished" );
  });
```

The same as:

```
$.ajax({
  type: "POST",
  url: url,
  data: data,
  success: success,
  dataType: dataType
});
```

Get JSON

The `getJSON()` functions is similar to the `get()` function but the returned data is interpreted as JSON data.

```
$.getJSON( "getdata.php", doneHandler)
  .done(function(data) { // Alternative done handler
    alert( "done" );
  })
  .fail(function() {
    alert( "error" );
  })
  .always(function() {
    alert( "finished" );
  });
```

The same as:

```
$.ajax({
  dataType: "json",
  url: url,
  data: data,
  success: success
});
```

Utilities

Each

A generic iterator function, which can be used to seamlessly iterate over both objects and arrays.

```
$.each(array, handler);
```

```
$.each(object, handler);
```

```
function handler (key, value) {
  // handle each element of the array/object
}
```

Can be used when receiving data from the `getJSON` function.

Read more on other utility functions.