Forum:Hack Forums

Thread: [Tutorial] jQuery Selectors & Basic jQuery Animation

Keeper - *August 2, 2014, 8:23 pm*

I intended to tutorialize quite a different topic but since my last tutorial was on SASS/SCSS, decided, I'd better carry on with this field and do a brief write-up on jQuery. As probably most of you already know, jQuery is one of the top-notch libraries for Javascript. Not only is Javascript a useful language but it is also pretty much easier to cope with compared to other programming languages out there. So in short, jQuery makes it even more elegant and handy. Let's now take a simple example:

Code
```
<!DOCTYPE html>
<html>
    <head>
        <meta http-equiv='Content-Type' content='text/html: charset=utf-8' />
        <title>jQuery</title>
        <script type='text/javascript' src='jquery.js'></script>
    </head>
    <body>
        <input type='button' value='Don\'t touch' id='btn' />
    </body>
</html>
```

The above HTML includes just a single input field. Say we want something to happen once the button has been pressed. And note that we need this done client-sided. The old way which is still also applicable (but highly not recommended) is using an in-line Javascript event listener like so:

Code
```
<input type='button' value='Don\'t touch' id='btn' onClick='alert(0);' />
```

Issue with this type of approach is that we need to exclusively specify in the HTML to listen for a particular event and what to do once the event has taken place. Which in programming is real bad practice and in HTML - even worse. So the better alternative is for something different to be listening for any event on the page (like one BigBrother, monitoring everything). In this scenario there is no need to alter the HTML at all and on the plus side, we can add or remove such stuff (listen or not) in real time execution. So let's see how we can achieve this using jQuery.

Code
```
$('#btn').click(function(){
    alert(0);
});
```

The dollar sign is called a selector (I'll narrate those later on). We select the id which is pretty much 'btn' and apply a click action on to it using a lambda (anonymous) function. For those of you who don't know what an anonymous function is, this is a function that has not been declared before and this is the first time we use it without calling a name for it. Basically, this might work or might not work. That depends on whether how the DOM has been loaded. If the button has been loaded first for some reason, then this would work. But since the script is defined above the input field, it can't really select it because it simply does not still exist, hence the action is not going to take place. There are cases in which things would happen the other way round but they are rare in existence.

To solve this problem, we could easily write the script below the HTML, however, this is not a good idea and should not be put into practice. You can also use the onLoad event for the body tag but that bears the same logic. For that occasion, jQuery has a special function called ready(). We use it the following way:

Code
```
$(document).ready(function(){
    $('#btn').click(function(){
        alert(0);
    });
});
```
Which means that once the document has been loaded, the code within the scope of this function is gonna be executed only then. To see if the id has been selected you can check the console but before that do the following:

Code
```
var a = $('#btn');
console.log(a);
```
and if selected properly you'll get [div#content]. The purpose of selecting an element is to manipulate it. Once we have it obtained, we are entitled to a great amount of in-built functions. So suppose we want to modify the whole of the HTML page. We can do this like that:

Code
```
a.html('random text');
```
So what would be the difference between document.getElementById() and selecting the id with jQuery? The main difference is that with pure Javascript if you want to select, let's say all elements with the class 'name', you would have to write loops, checks etc. While with jQuery we can do this in one line.

Code
```
$('.name');
```
A few other sample selectors are the following:

Code
```
$('*')
```
Selects it all. The whole document content.

Code
```
$('code, #id_example, .class_example')
```
Combining selectors. If we want to get all elements that have the id - id_example and all classes that have the name class_example, we can separate them with a comma.

Let's take the following HTML snippet to better visualize what's going on. As we are going to be selecting list elements, we'll write the following:

Code
```
<ul>
    <li>Example1</li>
    <li class='two'>Example2</li>
    <li>Example3</li>
    <li>Example4</li>
    <li><a href='#'>Example5</a></li>
</ul>

<h1>Text</h1>
<h5>Another</h5>
```
Code
```
$('li:first')
```
Selects the first list element from a group of ordered/unordered lists.

Code
```
$('li:last')
```
Respectevely selects the last element from a group of ordered/unordered lists.

Code
```
$('li:not(li:first)')
```
Selects all list elements from a group of ordered/unordered lists, except for the first one.

Code $('li:even')
Selects all list elements from a group of ordered/unordered lists whose position is an even number.

Code $('li:odd')
Selects all list elements from a group of ordered/unordered lists whose position is an odd number.

Code $('li:eq(1)')
Selects all list elements from a group of ordered/unordered lists whose position is equal to one.

Code $('li:gt(2)')
Selects all list elements from a group of ordered/unordered lists whose position is greater than two.

Code $('li:lt(2)')
Selects all list elements from a group of ordered/unordered lists whose position is lesser than two.

Code $(':header')
Selects all heading tags. These include <h1>, <h2>, <h3>, <h4>, <h5> and <h6>.

Code $('li:has(a)')
Selects all list elements from a group of ordered/unordered lists that have the <a> tag within them.

Code $('p:parent')
Selects the parent element of the paragraph.

Now so as a tutorial to be complete, we need to put these into practice. The idea is to make an effective animation using jQuery, so let's first assume that this is our HTML document including two images that are within several divs.

Code
```
<body>
    <div id='menu'>

      <a href='#'><img src='img/basketball.png' align='left' class='b'></a>

      <div id='abs'>
        <a href='#'><img src='img/football.png' align='left' class='c'></a>
      </div>

    </div>
  </body>
```
Make sure you have these in your head tag as well:

Code
```
<script type='text/javascript' src='js/jquery.js'></script>
<script type='text/javascript' src='js/jquery.easing.1.3.js'></script>
```
The first file is the jQuery library itself and the second one is a plugin for transition effects which I've decided to use. You can download it from here. And let's not forget our stylesheets which are in fact gonna align the elements and give them a better look.

Code
```
body {
  background-color: #000;
}

.b, .c {
```

```
    width: 70px;
    heigh: 70px;
    border: none;
    opacity: 0.4;
    margin-top: 50px;
    padding-left: 40px;
  }

  #menu {
    margin: 100px auto auto auto;
    width: 500px;
    height: 200px;
  }

  #abs {
    width: 200px;
    height: 100px;
    position: absolute;
    left: 50px;
    top: 200px;
  }
```

What we're going to do with the images is apply an effect which would make them bounce up and down, move up and right and so on and so forth. In order to achieve this, we ought to rely on the animate() function which is pretty much the core of jQuery animation effects. There are custom functions like fadeOut(), fadeIn() etc. but they are limited to their arguments and functionality. First, we need to check the document availability as I've pointed out earlier in this tutorial by using the ready() function.

Code `$(document).ready(function(){});`

Now here we need to perform our animation for the relevant element. Suppose we want to end up with the following animation:

One of the events will be happening upon a hover state and the other one upon a click. For the basketball ball to bounce let's assume the following: we will first select the class and append it to a variable so as to work with it easier.

Code `var button = $('.b');`

Now we need to actually animate it (keep in mind the 'b' class holds our basketball image).

Code
```
button.hover(function(){
      $(this).stop().animate({
        opacity: 1,
        marginTop: '0px'
      }, 800);
```

As we've already specified a margin-top of 20 pixels, we now take it back so the image to go up. Same goes for the opacity property. The function to apply whenever the button is hovered specifies the two properties and allows this to happen for a time schedule of 800 miliseconds. Which is 80% of a second (so in other words 1000 miliseconds would be a whole second).

The stop() function prevents multiple hovers to force the images to enter a number of animate() iterations. In short, if we omit the stop() function, the image will constantly be going up and down

once hovered multiple times (which is kinda annoying). So far, we've applied an animation only for the first effect which means that we still need to get the ball down once the mouse is no longer over the image.

Code
```
function(){
      $(this).stop().animate({
        opacity: 0.6,
        marginTop: '50px'
      }, 800, "easeOutBounce");
    });
  });
```
Normally as we did in the above example, we use the stop() function to bypass any unnecessary repetitions and changed the margin-top property to 50 pixels which will get the ball to its original position in the document. Now the 'easeOutBounce' is functioning as a parameter alongside the timing for the animation to take place. There is a number of effects for transitions that go with the easing plugin but I've chosen this one so as to achieve the bounce effect upon returning to the initial state.

Combining all of the above like so:

Code
```
$(document).ready(function(){
    var button = $('.b');

    button.hover(function(){
      $(this).stop().animate({
        opacity: 1,
        marginTop: '0px'
      }, 800);
    }, function(){
      $(this).stop().animate({
        opacity: 0.6,
        marginTop: '50px'
      }, 800, "easeOutBounce");
    });
  });
```
We end up with a nice effect for the basketball ball, just as we wanted it to be. So now for the football image, we would like to, let's say, move it up and left as if it was kicked. So let's get our ready() function in place:

Code
```
$(document).ready(function(){
    $('#abs').click(move);

  });
```
and animate the image as shown below:

Code
```
function move(){
      $('#abs').animate({
        left:'+=50px',
        top:'-=30px'
      }, 300);
    }
```
The function move() is selecting the abs ID for the div element and applies the animate() function to it along with a few specifications. The += means that the image will move with 50 pixels to the left

every time and will also keep its so far passed distance. Same goes for the top property. Since a bottom property won't work for this, we'll use a negative value for the top one.

Hope this helped any novice developers that are about to meet with jQuery. There is a lot more to talk about but it wouldn't all fit in a single tutorial. In the future, I'll cover as many aspects as I am aware of (these would include events, attributes, (cross-site) AJAX, JSON etc.

JumpWar - *August 3, 2014, 8:03 am*

Very good tutorial on the basics of selection and animation. Great you took the time to write this down.

If anyone is interested in doing selections without jQuery, I wrote a tutorial about that.