

## jQuery - Javascript's Best Friend

In this tutorial we will be taking your average everyday website and enhancing it with jQuery.

We will be adding ajax functionality so that the content loads into the relevant container instead of the user having to navigate to another page.

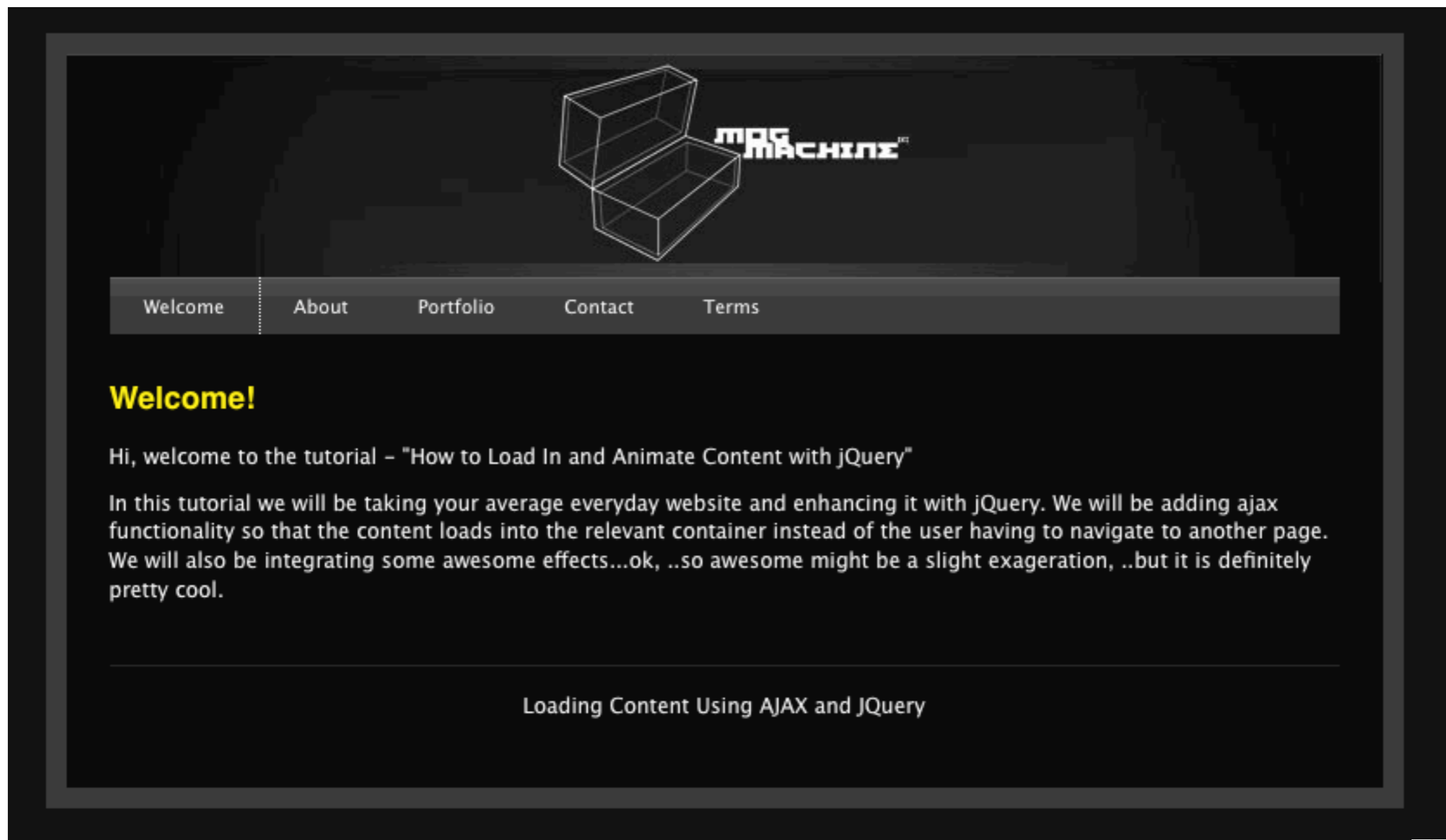
We will also be integrating some awesome animation effects.

The main purpose of this is to get you 100% comfortable with implementing tutorials you find online

# Animating Content with JQuery

## jQuery - Javascript's Best Friend

So first of all, I have put together a very simple layout for this example. Here's a screenshot and the HTML code we'll use.



## jQuery - Javascript's Best Friend

```
1. <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://
www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
2. <html xmlns="http://www.w3.org/1999/xhtml">
3. <head>
4. <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
5. <title>mmm... Ajax!</title>
6. <script type="text/javascript" src="jquery.js"></script>
7. <style type="text/css">
8. @import url(css.css);
9. </style>
10. <script type="text/javascript" src="js.js"></script>
11. </head>
12. <body>
13.     <div id="wrapper">
14.         <h1>ajax ... nettuts</h1>
15.         <ul id="nav">
16.             <li><a href="index.html">welcome</a></li>
17.             <li><a href="about.html">about</a></li>
18.             <li><a href="portfolio.html">portfolio</a></li>
19.             <li><a href="contact.html">contact</a></li>
20.             <li><a href="terms.html">terms</a></li>
21.         </ul>
22.         <div id="content">
23.             <h2>Welcome!</h2>
24.             <p>Text</p>
25.         </div>
26.         <div id="foot">Tutorial by James for NETTUTS</div>
27.     </div>
28. </body>
29. </html>
```

# Animating Content with JQuery

## Step 1

First thing's first, go and download the latest stable release of jQuery and link to it in your document:

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

First let's load the jQuery library and initiate a function when the document is ready (when the DOM is loaded).

```
1. $(document).ready(function() {  
2.     // Stuff here  
3. });
```

## Step 2

So what we want to do is make it so that when a user clicks on a link within the navigation menu on our page the browser does not navigate to the corresponding page, but instead loads the content of that page within the current page.

We want to target the links within the navigation menu and run a function when they are clicked:

```
1. $('#nav li a').click(function(){  
2.     // function here  
3. });
```

## Step 2

Let's summarize what we want this function to do in event order:

1. Remove current page content.
2. Get new page content and append to content DIV.

# Animating Content with JQuery

## Step 2

We need to define what page to get the data from when a link is clicked on.

All we have to do here is obtain the 'href' attribute of the clicked link and define that as the page to call the data from, plus we need to define whereabouts on the requested page to pull the data from - i.e. We don't want to pull ALL the data, just the data within the 'content' div, so:

```
1. var toLoad = $(this).attr('href')+' #content';
```

## Step 2

To illustrate what the above code does let's imagine the user clicks on the 'about' link which links to 'about.html'

- in this situation this variable would be: 'about.html #content'
- this is the variable which we'll request in the ajax call.

First though, we need to create a nice effect for the current page content. Instead of making it just disappear we're going to use jquery's 'hide' function like this:

```
1. $('#content').hide('fast',loadContent);
```

The above function 'hides' the #content div at a fast rate, and once that effect finished it then initiates the "loadContent" function (load the new content [via ajax]) - a function which we will define later (in step 4).

# Animating Content with JQuery

## Step 3

Once the old content disappears with an awesome effect, we don't want to just leave the user wondering before the new content arrives (especially if they have a slow internet connection) so we'll create a little "loading" graphic so they know something is happening in the background:



```
1. $('#wrapper').append('<span id="load">LOADING...</span>');
2. $('#load').fadeIn('normal');
```

Here is the CSS applied to the newly created #load div:

```
1. #load {
2.     display: none;
3.     position: absolute;
4.     right: 10px;
5.     top: 10px;
6.     background: url(images/ajax-loader.gif);
7.     width: 43px;
8.     height: 11px;
9.     text-indent: -9999em;
10. }
```

So, by default this 'load' span is set to display:none, but when the fadeIn function is initiated (in the code above) it will fade in to the top right hand corner of the site and show our animated GIF until it is faded out again.

## Step 4

So far, when a user clicks on one of the links the following will happen:

1. The current content disappears with a cool effect
2. A loading message appears

Now, let's write that loadContent function which we called earlier:

```
1. function loadContent() {  
2.     $('#content').load(toLoad, '', showNewContent)  
3. }
```

The loadContent function calls the requested page, then, once that is done, calls the 'showNewContent' function:

## Step 4

```
1. function showNewContent() {  
2.     $('#content').show('normal',hideLoader);  
3. }
```

This showNewContent function uses jQuery's show function (which is actually a very boring name for a very cool effect) to make the new (requested) content appear within the '#content' div. Once it has called the content it initiates the 'hideLoader' function:

## Step 4

```
1. function hideLoader() {  
2.     $('#load').fadeOut('normal');  
3. }
```

We have to remember to "return false" at the end of our click function - this is so the browser does not navigate to the page

It should work perfectly now.

# Animating Content with JQuery

## Step 4

Here is the code so far:

```
1. $(document).ready(function() {
2.
3.     $('#nav li a').click(function(){
4.
5.         var toLoad = $(this).attr('href')+' #content';
6.         $('#content').hide('fast',loadContent);
7.         $('#load').remove();
8.         $('#wrapper').append('<span id="load">LOADING...</span>');
9.         $('#load').fadeIn('normal');
10.        function loadContent() {
11.            $('#content').load(toLoad,'',showNewContent())
12.        }
13.        function showNewContent() {
14.            $('#content').show('normal',hideLoader());
15.        }
16.        function hideLoader() {
17.            $('#load').fadeOut('normal');
18.        }
19.        return false;
20.
21.    });
22. });
```

## Step 5

You could stop there but if you're concerned about usability (which you should be) it's important to do a little more work. The problem with our current solution is that it neglects the URL. What if a user wanted to link to one of the 'pages'? - There is no way for them to do it because the URL is always the same.

So, a better way to do this would be to use the 'hash' value in the URL to indicate what the user is viewing. So if the user is viewing the 'about' content then the URL could be: [www.website.com/#about](#). We only need to add one line of code to the 'click' function for the hash to be added to the URL whenever the user clicks on a navigation link:

```
1. window.location.hash = $(this).attr('href').substr(0,$(this).attr('href').length-5);
```

# Animating Content with JQuery

## Step 5

The code above basically changes the URL hash value to the value of the clicked link's 'href' attribute (minus the '.html' extension. So when a user clicks on the 'home' link (href=index.html) then the hash value will read '#index'.

Also, we want to make it possible for the user to type in the URL and get served the correct page. To do this we check the hash value when the page loads and change the content accordingly:

```
1. var hash = window.location.hash.substr(1);
2. var href = $('#nav li a').each(function(){
3.     var href = $(this).attr('href');
4.     if(hash==href.substr(0,href.length-5)){
5.         var toLoad = hash+'.html #content';
6.         $('#content').load(toLoad)
7.     }
8. });
```

# Animating Content with JQuery

With this included here is all the javascript code required: (plus the jQuery library)

```
1. $(document).ready(function() {
2.
3.     // Check for hash value in URL
4.     var hash = window.location.hash.substr(1);
5.     var href = $('#nav li a').each(function() {
6.         var href = $(this).attr('href');
7.         if(hash==href.substr(0,href.length-5)) {
8.             var toLoad = hash+'.html #content';
9.             $('#content').load(toLoad)
10.        }
11.    });
12.
13.    $('#nav li a').click(function() {
14.
15.        var toLoad = $(this).attr('href')+' #content';
16.        $('#content').hide('fast',loadContent);
17.        $('#load').remove();
18.        $('#wrapper').append('<span id="load">LOADING...</span>');
19.        $('#load').fadeIn('normal');
20.        window.location.hash = $(this).attr('href').substr(0,$
(this).attr('href').length-5);
21.        function loadContent() {
22.            $('#content').load(toLoad,'',showNewContent())
23.        }
24.        function showNewContent() {
25.            $('#content').show('normal',hideLoader());
26.        }
27.        function hideLoader() {
28.            $('#load').fadeOut('normal');
29.        }
30.        return false;
31.
32.    });
33. });
```

## Finished...

The great thing about this method is that's it's adaptable and can be applied to a site within minutes. It's fully unobtrusive and the website will work normally if the user has JS disabled.