jQuery Reference

Targeting elements using jQuery

$("#idName") - targets an id

$(".className") - targets a class

$("htmlTag") - targets an HTML tag

These methods can be mixed and matched. For example, to target all the paragraphs inside a DIV (box) with an ID of "container" you could use:

$("#container p")

jQuery FX

jQuery comes packaged with an array of FX:

- show()
- hide()

The show() and hide() FX work by essentially toggling the value of CSS property display between "block" and "none".

- fadeIn()
- fadeOut()

The fadeIn() and fadeOut() FX work by scaling the CSS opacity property. These two FX can also accept a duration setting inside the brackets. Accepted values are "slow", "fast" or a numerical value measured in milliseconds.

Manipulating CSS

jQuery's power comes from its ability to manipulate CSS properties. All CSS properties can be accessed and their values set using the following syntax:

```
$("#idName").css({
        backgroundColor: "#FF0000",
        fontSize: "15px",
        opacity: 0.6
})
```

Note: if you have multiple CSS properties then a comma is used to separate them. The last item is NOT followed by a comma.

Note: CSS properties such as background-color are camel-cased in jQuery, e.g. backgroundColor.

Note: The opacity property accepts values from 0 to 1, where 0 is fully transparent and 1 is fully opaque.

Animation

jQuery can animate elements (IDs, classes and tags) using the animate() function. For example:

```
$("#idName").animate({
	left: 200,
	top: 300
});
```

The above example will animate the left and top CSS properties to 200px and 300px respectively.

The animate() function can also accept a duration inserted BEFORE the closing bracket. For example to set a duration of 6 seconds we would use the following:

```
$("#idName").animate({
	left: 200,
	top: 300
}, 6000);
```

Note: The CSS property values in the above example will move an element to that fixed point. If you just want to move an element 100px from its current position you need to use the following syntax to add 100 onto the current position. Note: This value is written as a string:

```
$("#idName").animate({
	left: "+=100"
});
```

If the element is currently 200px from the left, then the above example will move it to 300px from the left.

Chaining FX and Animations

jQuery's dot syntax makes it easy to chain FX together. For example:

```
$("#idName").fadeIn(3000).animate({
	left: 400
}, 3000);
```

In this example, the ID will fade in over 3 seconds and then take another 3 seconds to move right by animating the left property.

Delaying FX and Animations

The delay() function makes it easy to organise the execution of FX:

```
$("#idName").fadeIn(3000).delay(5000).animate({
        left: 400
});
```

In this example the ID takes 3 seconds to fade in and then waits for 5 seconds before performing the animation.

Callback Functions

Although we can chain FX and animations there are times when we want to execute something at the end of an effect or animation that can't be simply chained. In these instances we use a callback function.

Callback functions are executed when the previous function has finished. For example, if we want to automatically move the user to a new page once a paragraph has faded out we would use:

```
$("#idName").fadeOut(5000, function() {
        window.location = "theNextPage.html";
});
```

Callback functions can be used to trigger any other event once the previous function has finished.

Using Functions

Functions can be used to create discrete modules of code that can be reused over and over again.

For example, if in your interactive story you have a NEXT >> button that the user clicks to move onto the next page you could write a single function to handle this and simply pass through the URL (page name) that the user will visit next. This is particularly handy if you are giving your user a choice, e.g. "go left or right" where choosing "left" will go to "page4.html" and choosing right will go to "page9.html".

In the below example we have a function called moveOn() which can be coded into its own javascript (.js) file called something like moveOn.js and included on every page where it is needed using <script src="moveOn.js"></script>. All we then need to do is call that function and pass the URL (page name) through to it:

```
function moveOn(url) {
        window.location = url;
};
```

We can then call this function when an element is clicked:

```
$("#leftBtn").click(function() {
	moveOn("page4.html");
});
$("#rightBtn").click(function() {
	moveOn("page9.html");
});
```

Triggering Functions from Hyperlinks

Functions can be triggered from hyperlinks. For example:

```
function someFunction() {
	do something here
}
```

To call this function from a hyperlink you must bind the click handler to an element. If your HTML hyperlink is Click Me then your JavaScript would be:

```
$("#trigger").click(function() {
	someFunction();
});
```

There is a shorter way to achieve the same thing. In the above example, we are using an anonymous function to call a named function. We could just do away with the anonymous function altogether and use:

```
$("#trigger").click( someFunction() );
```