6. Introducing jQuery Effects and Animations

Client-Side Web Programming

José Socuéllamos

Index

0 – Effects and Animation

- jQuery provides several techniques for adding animation to a web page.

- These include simple, standard animations that are frequently used, and the ability to craft sophisticated custom effects.

- These methods can take extra parameters:
  - Speed: time that takes the animation in ms or predefined strings like slow or fast.
  - Callback: a function to run some code when the animation has finished.

1 – Showing and Hiding Elements

- We can show or hide elements suddenly or at a certain speed:

Method	Description	Example
show(*speed,callback*)	Shows the selected elements	$("div").show("slow");
hide(*speed,callback*)	Hides the selected elements	$("p").hide(1000);
toggle(*speed,callback*)	Toggles between the hide() and show() methods	$(“ul").toggle();

2 - Fading Elements

- We can fade in or out elements completely or until a specific grade of opacity.

Method	Description	Example
fadeIn(*speed,callback*)	Fades in a hidden element (shows gradually)	$("#div1").fadeIn(2000);
fadeOut(*speed,callback*)	Fades out a visible element (hides gradually)	$("#div2").fadeOut("fast");
fadeToggle(*speed,callback*)	Toggles between the fadeIn() and fadeOut() methods	$("#div3").fadeToggle();
fadeTo(*speed*,*opacity*,*callback*)	Allows fading to a given opacity (between 0 and 1)	$("#div4").fadeTo("slow", 0.6);

3 – Sliding Elements

- Sliding is another way to show or hide DOM elements. It slides elements up and down.

Method	Description	Example
slideDown(*speed,callback*)	Slides down an element	$("#div1"). slideDown("slow");
slideUp(*speed,callback*)	Slides up an element	$("#div2"). slideUp(3000);
slideToggle(*speed,callback*)	Toggles between the slideDown() and slideUp() methods	$("#div3").slideToggle();

4 – Create your own animations

- jQuery allows us to perform a custom animation of a set of CSS properties.

- The animate() method changes an element from one state to another with CSS styles.

- The CSS property value is changed gradually, to create an animated effect.

4 – Create your own animations

- animate({parameters}, speed, easing, callback)
  - Parameters: CSS properties of the elements to animate. The names must be camel-cased (marginRight, borderLeft…)
  - Speed: duration in ms, slow or fast
  - Easing: speed at which the animation progresses at different points of itself.
    - Linear: moves slower at the beginning/end, but faster in the middle
    - Swing: moves in a constant speed (default value)
  - Callback: function to be executed when the animation is complete.

4 – Create your own animations

- animate() requirements and conditions:
  - Only numeric values can be animated (like " width: '400px' ").
  - String values cannot be animated (like "background-color:blue")
  - Only the strings "show", "hide" and "toggle" are allowed.
  - It's possible to define relative values, by putting += or –= before the value.
  - To manipulate the position of an element first set the CSS position property of the element to relative, fixed, or absolute!
  - Color animation is not included in jQuery!!!

4 – Create your own animations

- animate() examples:

```
$("div").animate({
    left: '250px'
});
```

One property

```
$("div").animate({
    height: 'toggle'
});
```

Using "show", "hide", or "toggle" as value

```
$("div").animate({
      bottom: '150px',
      height: '+=200px',
      width: '+=200px',
      opacity: '0.25'
    }, "slow", "linear");
```

Multiple properties
Relative values
Speed & Easing

```
$("div").animate({fontSize:'2em'}, 2000, function() {
    alert("The Font size has changed");
});
```

Callback function

```
$("div").animate({fontSize:'2em'}, 2000);
alert("The Font size will change when you close this");
```

Without Callback function

5 – Queuing animations

- What would you expect to happen if you were executing the following code?

```
$(".test").animate({left: '+=256px'}, "slow");
$(".test").animate({left: '+=256px'}, "slow");
```

The .test element will move…

A. 256px to the right and stop
B. 256px to the right, stop and then 256px to the right again and stop
C. to the end of the screen and fall off the browser

5 – Queuing animations

- In jQuery we can use queue functionality for animations.
- If we write multiple animate() calls after each other, an "internal" queue is created, running the animate calls ONE by ONE.

```
var div = $("div");
startAnimation();
function startAnimation(){
    div.animate({height: 300px}, "fast");
    div.animate({width: 300px}, "slow");
    div.animate({height: 100px}, "fast");
    div.animate({width: 100px}, "slow", startAnimation);
```

Callback function
Queue functionality

- This div will increase its height and then its width.
- After that, it will decrease its height and then its width.
- To finish, the animation will start over and over.

5 – Queuing animations

- Run the following code…

```
console.log(1);
$(".test").animate({left: '+=256px'}, "slow");
console.log(2);
$(".test").animate({left: '+=256px'}, "slow");
console.log(3);
```

- Does it run as expected? Why?
- As we said before, jQuery creates an "internal" queue, but continues to run the rest of the code without waiting for the animations to finish.

6 – Stopping animations

- stop(stopAll, goToEnd) – Stops the currently running animation for the selected elements.
  - stopAll: boolean to specify whether or not to stop the queued animations as well (optional, default is false)
  - goToEnd: boolean to specify whether or not to complete all animations immediately (optional, default is false)

```
var div = $("div");
div.animate({height: 300px}, "fast");
div.animate({width: 300px}, "fast");
div.animate({height: 100px}, "fast");
div.animate({width: 100px}, "fast");
```

```
div.stop(true);
```

Stop all queued animations

```
div.stop(true, true);
```

Stop but complete all animations immediately

7 – Chaining animations

- We have learned to write jQuery statements one after the other.
- However, the Chaining technique allows us to run multiple jQuery commands, one after the other, on the same element(s).
- To do that, we simply append one action to the previous action.

```
$("#div1").css("background-color","green").fadeOut(3000).fadeIn(3000).animate({width:
"300px"}).animate({opacity: 0.5});
```

The background goes green, fades out then fades in, grows in width and turns transparent

```
$("#div1").css("background-color","green")
  .fadeOut(3000)
  .fadeIn(3000)
  .animate({width: "300px"})
  .animate({opacity: 0.5});
```

We can bend the syntax and write each statement in a new line for a better reading

8 – Start creating your own animations

http://api.jquery.com/animate/