

# Lesson #9:

# Introduction to JavaScript and jQuery

# What is JavaScript?

- JavaScript is a client-side scripting language. This means the web surfer's browser will be running the script.
- The main benefit of Javascript is to add additional interaction between the website and its visitors.
- JavaScript, despite the name, is unrelated to Java, although both have the common C syntax, and JavaScript copies many Java names and naming conventions.

# A basic script

- JavaScripts are embedded in the HTML page.  
No need for special extensions as all processing is done client-side.

```
<html>
```

```
<body>
```

```
<script type="text/JavaScript">
```

```
<!--
```

```
document.write("Hello World!");
```

```
//-->
```

```
</script>
```

```
</body>
```

```
</html>
```

optional  
;

COMMENTS

HTML  
JavaScript

# Simple redirection

- JavaScript being client-side, the webmaster is dependent to the client's preferences. It is possible to disable JS in the browser settings. The following script will be inoperable if JS is not enabled.

```
<script type="text/JavaScript">  
window.location =  
    "http://www.mydomain.com/mysite.html"  
</script>
```

# Scripts in <head> part

- Basic scripts are usually placed in the body part where you need them.
- If you want to have a script run on some event, such as when a user clicks somewhere, then you will place that script in the head section.
- Many times function definitions will take place in the head section, and function calls in the body section.

# Function definition

```
<head>
```

```
<script type="text/JavaScript">
```

```
<!--
```

```
function popup() {
```

```
    alert("Hello World")
```

```
}
```

```
//-->
```

```
</script>
```

```
</head>
```

Used to hide code  
from older browsers  
(not really necessary)

The alert function  
brings up a popup box  
with the text provided in it.

# Function call

```
<body>
```

```
<input type="button" onclick="popup()" value="popup">
```

```
</body>
```

```
</html>
```



Event



# External file

- A script can also be placed in an external file bearing the .js extension. In that case the script in that file is not embedded in HTML, it is pure JavaScript.
- Here is an example of the contents of a file called *pop.js*.

```
function popup()  
{  
    alert("Hello World");  
}
```



# External file

- Here are the head and body sections of a page using that external file.

```
<head>
```

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

```
</head>
```

```
<body>
```

```
<input type="button" onclick="popup()" value="Click  
Me!">
```

```
</body>
```

# Operators and Variables

- The operators are the same as those you can find in the C/Java/Perl/PHP family.
- + - \* / %
- < <= > >= == !=
- You don't have to declare variables before you use them.
- Sometimes we use the keyword `var` when using a variable the first time in a program. It is not required, just some sort of convention. Naming rules are similar to Perl's or PHP.

```
var greeting = "Good morning!";
```

# Events

- The building blocks of an interactive web page is the JavaScript event system. An event in JavaScript is something that happens with or on the web page.
- JavaScript has predefined names that cover numerous events that can occur.
- To capture an event and make something happen when that event occurs, you must specify the event, the HTML element that will be waiting for the event, and the function(s) that will be run when the event occurs.

# Common Events

- `onload` / `onUnload`: triggered when the user enters or leaves the page.
- `onFocus` / `onBlur` / `onChange`: triggered when an object is activated (`onFocus`), loses focus (`onBlur`) and when it changes value (`onChange`). Usually goes with form objects.
- `onSubmit`: triggered when a form is submitted. Used to validate a form.
- `onClick`: triggered when an object is clicked, usually a link or a button.
- `onMouseOver` / `onMouseOut`: triggered when moved over an element or off the element.

More info:

[ihypress.net/programming/javascript/events.html](http://ihypress.net/programming/javascript/events.html)

# Statements

- Statements in JavaScript are ended by a semicolon or a new line.

```
<script type="text/javascript">
```

```
<!--
```

```
var number = 7;
```

```
if(number == 7){
```

```
    document.write("Lucky 7!");
```

```
}
```

```
//-->
```

```
</script>
```

# If statement

```
<script type="text/javascript">
```

```
<!--
```

```
var number = 10;
```

```
if(number == 7){
```

```
    document.write("Lucky 7!");
```

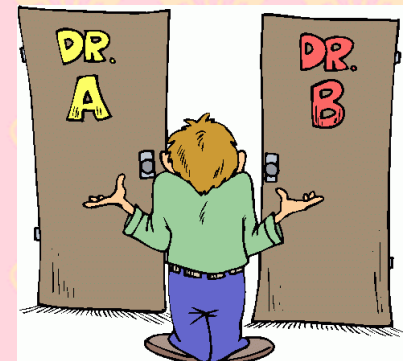
```
}else{
```

```
    document.write("You're not very lucky today...");
```

```
}
```

```
//-->
```

```
</script>
```



# Nested if statements

```
<script type="text/javascript">
```

```
<!--
```

```
var visitor = "principal";
```

```
if(visitor == "teacher"){
```

```
    document.write("My dog ate my homework...");
```

```
}else if(visitor == "principal"){
```

```
    document.write("It was the other guy's fault!");
```

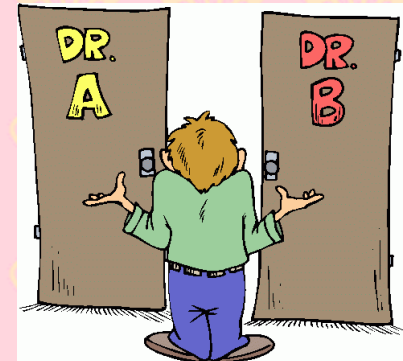
```
} else {
```

```
    document.write("How do you do?");
```

```
}
```

```
//-->
```

```
</script>
```





# While loops

```
<script type="text/javascript">
```

```
var i = 1;
```

```
while(i < 200){
```

```
    document.write(i + " ");
```

```
    ++i;
```

```
}
```

```
</script>
```



# For loops

```
<script type="text/javascript">
```

```
for(i = 100; i >=0; --i)
{
    document.write(i + " ");
}
```

```
</script>
```



# Arrays

- You have seen arrays in other languages, and they aren't that different in JavaScript. You have to use a special function to create a new array.

```
<script type="text/javascript">
```

```
var myArray = new Array();
```

```
myArray[0] = "Orange";
```

```
myArray[1] = "Apple";
```

```
myArray[2] = "Pumpkin";
```

```
document.write(myArray[0] + myArray[1] +  
    myArray[2]);
```

```
</script>
```

OrangeApplePumpkin

# Alerts

- Alerts should be very, very rarely used and even then these following guidelines should be considered when using them. JavaScript alerts are ideal for the following situations:
- If you want to be absolutely sure they see a message before doing anything on the website.
- You would like to warn the user about something.
- An error has occurred and you want to inform the user of the problem.
- When asking users for confirmation of some action.

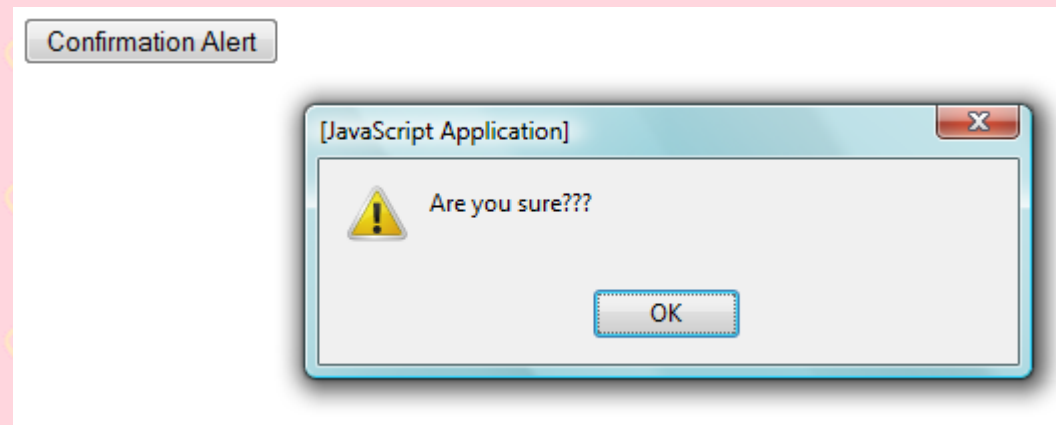
# Alerts

- Here is an alert example to be sure your visitors are fully aware before submitting a form.

<form>

<input type="button" onClick="alert('Are you sure???');" value="Confirmation Alert"></input>

</form>



# Confirm

- Because *alert* does not give you the choice of accepting or refusing (besides exiting the browser), there can be a better choice: *confirm*.
- A small dialog box pops up and appears in front of the web page currently in focus. The confirm box is different from the alert box. It supplies the user with a choice; they can either press OK to confirm the popup's message or they can press cancel and not agree to the popup's request.

# Confirm

```
<head><script type="text/javascript">
```

```
function confirmation() {
```

```
    var answer = confirm("Really want to leave this page?")
```

```
    if (answer){
```

```
        alert("Bye bye!")
```

```
        window.location = "http://www.ryerson.ca/";}
```

```
    else{
```

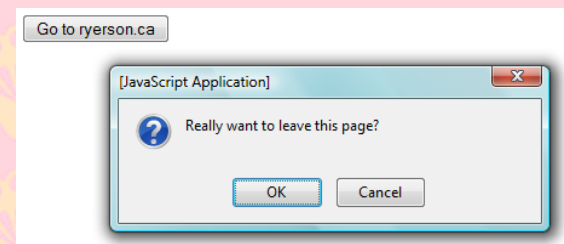
```
        alert("Thanks for sticking around!")}}
```

```
//-->
```

```
</script></head><body><form>
```

```
<input type="button" onclick="confirmation()" value="Go to  
ryerson.ca">
```

```
</form></body>
```





# Timed Redirection

- We have seen redirection early on with the `window.location` method. Here is another example but this time using a timing delay.

```
<head><script type="text/javascript">  
function delayer(){  
    window.location = "http://www.toronto.ca"  
</script></head>  
<body onLoad="setTimeout('delayer()', 5000)">  
<h2>Prepare to be redirected!</h2>  
<p>The city of Toronto website will appear in a few  
seconds</p>  
</body></html>
```



milliseconds

**Prepare to be redirected!**

The city of Toronto website will appear in a few seconds

# Popup windows

- One of the most flagrant misuses of JavaScript are popup windows.
- The `window.open()` function creates a new browser window, customized to your specifications, without the use of an HTML anchor tag.
- There are three arguments that the `window.open` function takes: The relative or absolute URL of the web page to be opened, the text name for the window and a long string that contains all the different properties of the window (key=value separated by commas).

```
window.open( "http://www.cbc.ca/", "pop", "status  
= 1, height = 300, width = 300, resizable = 0")
```

# Window properties

- dependent: subwindow closes if the parent window (the window that opened it) closes.
- fullscreen: display browser in fullscreen mode.
- height: the height of the new window, in pixels.
- width: the width of the new window, in pixels.
- left: pixel offset from the left side of the screen.
- top: pixel offset from the top of the screen.
- resizable: allow the user to resize the window or prevent the user from resizing. (not functional in Firefox)
- status: Display or don't display the status bar.

# Popup Example

```
<head>
```

```
<script type="text/javascript">
```

```
function pop1() {
```

```
    window.open( "http://www.cs.ryerson.ca/",  
        "ryerson", "status=0, height=400, width=400,  
        left=500, top=10, resizable=0, dependent=1" );
```

```
</script>
```

```
</head>
```

```
<body onLoad = "pop1();">
```

```
    Main window
```

```
</body>
```

# Date and time



- The Date object is useful when you want to display a date or use a time stamp in some sort of calculation. You can either make a Date object by supplying the date of your choice, or you can let JavaScript create a Date object based on your **visitor's** system clock (not the server's).
- It is important to note that if someone's clock is off by a few hours or they are in a different time zone, then the Date object will create a different time from the one created on your own computer.



# Date and time



```
var currentTime = new Date()
```

- The Date object has been created, and now we have a variable that holds the current date. To get the information we need to print out, we have to utilize some or all of the following functions:
- `getTime()` : Number of milliseconds since 1/1/1970 at 12:00 AM
- `getSeconds()` : Number of seconds (0-59)
- `getMinutes()` : Number of minutes (0-59)
- `getHours()` : Number of hours (0-23)
- `getDay()` : Day of the week(0-6). 0 = Sunday, ... , 6 = Saturday

# Date and time



- getDate() : Day of the month (0-31)
- getMonth() : Number of month (0-11)
- getFullYear() : The four digit year (1970-9999)

```
<script type="text/javascript">
```

```
var currentTime = new Date()
```

```
var month = currentTime.getMonth() + 1
```

```
var day = currentTime.getDate()
```

```
var year = currentTime.getFullYear()
```

```
document.write(month + "/" + day + "/" + year)
```

```
</script>
```



# Form validation

- There's nothing more troublesome than receiving orders, guest book entries, or other form submitted data that are incomplete in some way. You can avoid these headaches once and for all with JavaScript's form validation.
- The idea behind JavaScript form validation is to provide a method to check the user entered information before they can even submit it. JavaScript also lets you display helpful alerts to inform the user what information they have entered incorrectly and how they can fix it.



# Form Validation: Checking for Non-Empty

- This has to be the most common type of form validation. You want to be sure that your visitors enter data into the HTML fields you have "required" for a valid submission.

```
<script type='text/javascript'>
```

function  
parameters

```
function notEmpty(elem, helperMsg){
```

```
    if(elem.value.length == 0){
```

```
        alert(helperMsg);
```

```
        elem.focus();
```

```
        return false;    }
```

```
        return true;}
```

```
</script>
```



# Form Validation: Checking for Non-Empty

<form>

Enter your name: <input type="text" id="req1" />

<input type="button"

onclick="notEmpty(document.getElementById('req1'), 'You must enter your name!')"

value="Check Field" />

</form>

- As long as elem.value.length isn't 0 then it's not empty and we return true, otherwise we send an alert to the user with a helperMsg to inform them of their error and return false.



# Form Validation: Checking for all numbers

```
function isNumeric(elem, helperMsg){  
    var numericExpression = /^[0-9]+$/;  
    if(elem.value.match(numericExpression)){  
        return true;  
    }else{  
        alert(helperMsg);  
        elem.focus();  
        return false;  
    }  
}
```

See  
[evolt.org/regexp\\_in\\_javascript](http://evolt.org/regexp_in_javascript)  
for regular expressions  
in JavaScript.



# Form Validation: Length restriction

- Being able to restrict the number of characters a user can enter into a field is one of the best ways to prevent bad data. For example, if you know that the postal code field should only be 7 characters you know that 2 characters is not sufficient.

```
<script type="text/javascript">  
function lengthRestriction(elem, min, max){  
    var uInput = elem.value;  
    if(uInput.length >= min && uInput.length <= max){  
        return true;  
    }else{  
        alert("Please enter between " +min+ " and " +max+ " characters");  
        elem.focus();  
        return false;}}  
</script>
```



# Form Validation: Length restriction

- Being able to restrict the number of characters a user can enter into a field is one of the best ways to prevent bad data. For example, if you know that the postal code field should only be 7 characters you know that 2 characters is not sufficient.

```
<form>
```

```
Postal Code? <input type="text" id="pc" />
```

```
<input type="button"
```

```
  onclick="lengthRestriction(document.getElementById('pc'  
    ), 6, 7)"
```

```
  value="Validate Postal Code" />
```

```
</form>
```





# Form Validation: Length restriction

- Every email is made up for 5 parts: A combination of letters, numbers, periods, hyphens, plus signs, and/or underscores, the @ symbol, a combination of letters, numbers, hyphens, and/or periods, a period and the top level domain (com, net, org, ca, in, ...).

```
function emailValidator(elem, helperMsg){  
    var emailExp = /^[\\w\\-\\.\\+]+\\@[a-zA-Z0-9\\.\\-]+\\.?[a-zA-Z0-9]{2,63}$/;  
    if(elem.value.match(emailExp)){  
        return true;  
    }else{  
        alert(helperMsg);  
        elem.focus();  
        return false;}}
```





# Form Validation: FINAL VALIDATION

- The final step is to be able to perform all of these validation steps when the user is ready to submit their data.
- Each form has a JavaScript event called `onSubmit` that is triggered when its submit button is clicked. If this even returns 0 or false then a form cannot be submitted, and if it returns 1 or true it will always be submitted.
- `<form onSubmit="return formValidator()" >`
- A bunch of nested ifs validating each form element and calling other functions (including the ones we saw already) will constitute the `formValidator` function.
- Try to write one to validate a phone number (all digits), a postal code and make sure no field is empty!



# Strings: length property

- length: the length property returns the number of characters that are in a string, using an integer

```
<script type="text/javascript">
```

```
var myString = "the quick brown fox";
```

```
document.write("The string is this long: " +  
    myString.length);
```

```
myString = myString + " jumped over the lazy dog.";
```

```
document.write("<br />The string is now this long: " +  
    myString.length);
```

```
</script>
```

```
The string is this long: 19  
The string is now this long: 45
```



# Strings: split function

- The ability to split up a string into separate chunks is available in JavaScript as well. If you have a long string like "Toronto Ottawa London Hamilton Windsor" and want to store each city name separately, you can specify the space character " " and have the split function create a new chunk (an array cell) every time it sees a space.

```
<script type="text/javascript">
```

```
var cities = "Toronto Ottawa London Hamilton  
Windsor";
```

```
var cityArray = cities.split(" ");
```

```
for(i = 0; i < cityArray.length; ++i){
```

```
    document.write("<br /> City #" + i + " = " +  
    cityArray[i]); }
```

```
</script>
```

City #0 =	Toronto
City #1 =	Ottawa
City #2 =	London
City #3 =	Hamilton
City #4 =	Windsor



# Strings: search function

- This string function takes a regular expression and then examines that string to see if there are any matches for that expression. If there is a match, it will return the position in the string where the match was found. If there isn't a match, it will return -1.

```
<script type="text/javascript">
```

```
var cities = "Toronto Ottawa London Hamilton Windsor";
```

```
var regexp = /on|to/;
```

```
var found = cities.search(regexp);
```

```
if(found != -1)
```

```
    document.write("Found at position " + found);
```

```
else
```

```
    document.write("No match");
```

```
</script>
```

Found at position 3



# Strings: replace function

- The string replace function has two arguments: what word is going to be replaced (this can be a string or a regular expression) and what the word will be replaced with (this needs to be a string).
- *replace* returns the new string with the replaces, but if there weren't any words to replace, then the original string is returned.

```
<script type="text/javascript">
```

```
var cities = "Toronto Ottawa London Hamilton Windsor";
```

```
var modcities = cities.replace(/on/g, "??");
```

```
document.write ("Cities: " + cities);
```

```
document.write ("<br />Modified cities: " + modcities);
```

```
</script>
```

```
Cities: Toronto Ottawa London Hamilton Windsor  
Modified cities: Tor??to Ottawa L??d?? Hamilt?? Windsor
```





# Changing HTML

- Each HTML element has an *innerHTML* property that defines both the HTML code and the text that occurs between that element's opening and closing tag. By changing an element's *innerHTML* after some user interaction, you can make much more interactive pages.
- However, using *innerHTML* requires some preparation if you want to be able to use it easily and reliably. First, you must give the element you wish to change an id. With that id in place you will be able to use the *getElementById* function, which works on all browsers.
- After you have that set up you can now manipulate the text of an element. To start off, let's try changing the text inside a bold tag.

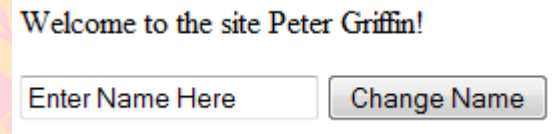
# Changing HTML

```
<script type="text/javascript">  
function changeText(){  
    document.getElementById('visitor').innerHTML =  
    "Homer Simpson";  
}  
</script>  
<p>Welcome to the site <span id="visitor">Peter  
Griffin</span>! </p>  
<input type="button" onclick="changeText()" value="Change Name" />
```



# Changing HTML

```
<script type="text/javascript">  
function changeText(){  
    var name = document.getElementById('name').value;  
    document.getElementById('visitor').innerHTML = name;  
}  
</script>  
<p>Welcome to the site <span id="visitor">Peter Griffin</span>!  
</p>  
<input type="text" id="name" value="Enter Name Here" />  
<input type="button" onclick="changeText()" value="Change  
Name" />
```



Welcome to the site Peter Griffin!

Enter Name Here

# Changing HTML

```
<script type="text/javascript">
```

```
function changeCol(){
```

```
    var oldHTML =
```

```
    document.getElementById('para').innerHTML;
```

```
    var newHTML = "<span style='color:#cc0000'>" +  
    oldHTML + "</span>";
```

```
    document.getElementById('para').innerHTML =  
    newHTML;
```

```
}
```

```
</script>
```

```
<p id='para'>Welcome to the site my friend!</p>
```

```
<input type="button" onclick="changeCol()"  
    value="Change Text" />
```

# jQuery

- The purpose of jQuery is to make it much easier to use JavaScript on your website.
- jQuery takes a lot of common tasks that require many lines of JavaScript code to accomplish, and wraps them into methods that you can call with a single line of code.
- jQuery also simplifies a lot of the complicated things from JavaScript, like AJAX calls and DOM manipulation.

# jQuery

- There are lots of other JavaScript frameworks out there, but jQuery seems to be the most popular, and also the most extendable.
- Many of the biggest companies on the Web use jQuery.
- The jQuery library is a single JavaScript file, and you reference it with the HTML `<script>` tag (notice that the `<script>` tag should be inside the `<head>` section)

```
<script  
src="https://ajax.googleapis.com/ajax/libs/jquery/3.3.1/jquery.min.  
js"></script>
```

or

```
<script src="https://ajax.aspnetcdn.com/ajax/jquery/jquery-  
3.3.1.js"></script>
```

# jQuery

- With jQuery you select (query) HTML elements and perform "actions" on them.
- The jQuery syntax is tailor made for selecting HTML elements and performing some action on the element(s).
- Basic syntax is: `$(selector).action()`
- A `$` sign to define/access jQuery
- A (selector) to "query (or find)" HTML elements
- A jQuery `action()` to be performed on the element(s)
- Examples:
  - `$(this).hide()` - hides the current element.
  - `$("p").hide()` - hides all `<p>` elements.
  - `$(".test").hide()` - hides all elements with `class="test"`.
  - `$("#test").hide()` - hides the element with `id="test"`.

# jQuery: The document ready event

- To prevent any jQuery code from running before the document is finished loading (is ready).
- Here are some examples of actions that can fail if methods are run before the document is fully loaded:
- *Trying to hide an element that is not created yet*
- *Trying to get the size of an image that is not loaded yet*
- `$(document).ready(function() {`
- `// jQuery methods go here...`
- `});`



# Jquery Selectors

- jQuery selectors are one of the most important parts of the jQuery library.
- jQuery selectors allow you to select and manipulate HTML element(s).
- jQuery selectors are used to "find" (or select) HTML elements based on their id, classes, types, attributes, values of attributes and much more. It's based on the existing CSS Selectors, and in addition, it has some own custom selectors.

[www.w3schools.com/jquery/  
jquery\\_selectors.asp](http://www.w3schools.com/jquery/jquery_selectors.asp)

# Jquery Events

- All the different visitor's actions that a web page can respond to are called events.

Mouse Events	Keyboard Events	Form Events	Document/Window Events
click	keypress	submit	load
dblclick	keydown	change	resize
mouseenter	keyup	focus	scroll
mouseleave		blur	unload

- To assign a click event to all paragraphs on a page, you can do this: `$("p").click();`

[www.w3schools.com/jquery/jquery\\_events.asp](http://www.w3schools.com/jquery/jquery_events.asp)

# Jquery Effects - Hide and Show

```
$("#hide").click(function(){  
    $("p").hide();  
});
```

Hides and shows  
all paragraphs  
at the click of  
a button.

```
$("#show").click(function(){  
    $("p").show();  
});
```

More options  
here:

[www.w3schools.com/  
jquery/jquery\\_  
hide\\_show.asp](http://www.w3schools.com/jquery/jquery_hide_show.asp)

```
<button id="hide">Hide</button>  
<button id="show">Show</button>
```

# Jquery Effects - Fade

```
$("#button").click(function(){  
    $("#div1").fadeIn();  
    $("#div2").fadeIn("slow");  
    $("#div3").fadeIn(3000);  
});
```

```
<button>Click to fade</button>
```

```
<br><br>
```

```
<div id="div1" style="width:80px;height:80px;display:none;background-color:red;"></div><br>
```

```
<div id="div2" style="width:80px;height:80px;display:none;background-color:green;"></div><br>
```

```
<div id="div3" style="width:80px;height:80px;display:none;background-color:blue;"></div>
```

Fade in divs at different speeds.

More options here:

[www.w3schools.com/jquery/jquery\\_fade.asp](http://www.w3schools.com/jquery/jquery_fade.asp)

# Jquery Effects - Animation

The jQuery animate() method is used to create custom animations.

The required params parameter defines the CSS properties to be animated.

The optional speed parameter specifies the duration of the effect. It can take the following values: "slow", "fast", or milliseconds.

The optional callback parameter is a function to be executed after the animation completes.

More options here:  
[www.w3schools.com/  
jquery/  
jquery\\_animate.asp](http://www.w3schools.com/jquery/jquery_animate.asp)



End of lesson