jQuery Effects

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

- These include simple, standard animations that are frequently used, and the ability to create custom effects.
Custom Animation

- The jQuery animate() method lets you create custom animations.

- The method changes an element from one state to another with CSS styles - the CSS property value is changed gradually, to create an animated effect.

- Only numeric values can be animated (like "margin:30px").

- String values cannot be animated (like "background-color:red").
Animation

- Shorthand CSS properties (e.g. font, background, border) are not fully supported.

- For example, if you want to animate the rendered border width, at least a border style and border width other than "auto" must be set in advance.

- Or, if you want to animate font size, you would use fontSize or the CSS equivalent 'font-size' rather than simply 'font'.
Animation Syntax

- animate() syntax:
  
  $\text{(selector).animate({properties),speed, easing, callback});}

- The properties parameter (required) defines the CSS properties to be animated.

- The speed parameter (optional) specifies the duration of the effect. Possible values: "slow", "fast", or milliseconds.

- The easing parameter (optional) specifies the speed of the element in different points of the animation. Possible values:
  
  - "swing" - moves slower at the beginning/end, but faster in the middle (default)
  - "linear" - moves at a constant speed

- The callback parameter (optional) is a function to be executed after the animation completes.
jQuery Callback Functions – an aside

- JavaScript statements are executed line by line. However, with effects, the next line of code can be run even though the effect is not finished. This can create errors.

- Creating a callback function prevents this from happening.

- A callback function is executed after the current effect is 100% finished.

- Typical syntax:
  \$\texttt{(selector).hide(speed,\texttt{callback};}
jQuery Callback Functions – an aside

The example below has a callback parameter that is a function that will be executed after the hide effect is completed:

```
$("button").click(function(){
    $("p").hide("slow",function(){
        alert("The paragraph is now hidden");
    });
});
```
Animation

- By default, all HTML elements have a static position, and cannot be moved.

- To manipulate the position, you must first set the CSS position property of the element to relative, fixed, or absolute in the stylesheet.
animate() Manipulate Multiple Properties

- Multiple properties can be animated at the same time:

```javascript
$("button").click(function(){
  $("div").animate(
    {
      left:'250px',
      opacity:'0.5',
      height:'150px',
      width:'150px'
    }
  );
});
```
animate() Point to Note

- When manipulating CSS properties with the `animate()` method all property names must be camel-cased.

- This means you will need to write `paddingLeft` instead of `padding-left`, `marginRight` instead of `margin-right`, and so on.
animate() Using Relative Values

- It is possible to define relative values (the value is then relative to the element's current value).

- This is done by putting += or -= in front of the value:

```javascript
$("button").click(function(){
    $("div").animate({
        left:'250px',
        height:'+=150px',
        width:'+=150px'
    });
});
```
**animate() - Using Pre-defined Values**

- You can also specify a property's animation value as "show", "hide", or "toggle":

```javascript
$("button").click(function(){
  $("div").animate({
    height: 'toggle'
  });
});
```
animate() - Uses Queue Functionality

- By default, jQuery comes with queue functionality for animations.

- This means that if you write multiple `animate()` calls after each other, jQuery creates an "internal" queue with these method calls. Then it runs the animate calls ONE by ONE.

- So, if you want to perform different animations after each other, we take advantage of the queue functionality:
animate() - Uses Queue Functionality

```javascript
$("button").click(function(){
    var div=$('#div');
    div.animate({height:'300px',opacity:'0.4'},"slow");
    div.animate({width:'300px',opacity:'0.8'},"slow");
    div.animate({height:'100px',opacity:'0.4'},"slow");
    div.animate({width:'100px',opacity:'0.8'},"slow");
});

$("button").click(function(){
    var div=$('#div');
    div.animate({left:'100px'},"slow");
    div.animate({fontSize:'3em'},"slow");
});
```
Basic Animation: `hide()` and `show()`

- You can hide and show HTML elements with the `hide()` and `show()` methods:

- Syntax:
  
  ```javascript
  $(selector).hide(speed, callback);
  $(selector).show(speed, callback);
  ```
Fading

- You can fade elements in and out...

- jQuery has the following fade methods:
  - fadeIn()
  - fadeOut()
  - fadeToggle()
  - fadeTo()
Fading

- **fadeOut()** syntax:
  
  $$\$(\text{selector}) . \text{fadeOut}(\text{speed, callback}) ;$$

- **fadeIn()** syntax:
  
  $$\$(\text{selector}) . \text{fadeIn}(\text{speed, callback}) ;$$

- The **speed** parameter (optional) specifies the duration of the effect. Possible values: "slow", "fast", or milliseconds.

- The **callback** parameter (optional) is a function to be executed after the fading completes.
Fading

- **fadeToggle() syntax:**
  - `$(selector).fadeToggle(speed, callback);`

- This toggles between the **fadeIn()** and **fadeOut()** methods.
  - If the elements are faded out, **fadeToggle()** will fade them in.
  - If the elements are faded in, **fadeToggle()** will fade them out.
Fading

- The `fadeOut()` method allows fading to a given opacity (value between 0 and 1).

- `fadeOut()` syntax:
  ```javascript
  $(selector).fadeOut(speed, opacity, callback);
  ```
Sliding

- The slide methods slides elements up and down.

- jQuery has the following slide methods:
  - `slideDown()`
  - `slideUp()`
  - `slideToggle()`
Sliding

- **slideDown()** syntax:
  - `$(selector).slideDown(speed, callback);`

- **slideUp()** syntax:
  - `$(selector).slideUp(speed, callback);`

- The **speed** parameter (optional) specifies the duration of the effect. Possible values: "slow", "fast", or milliseconds.

- The **callback** parameter (optional) is a function to be executed after the fading completes.
Sliding

- `slideToggle()` syntax:
  - `$(selector).slideToggle(speed, callback);`

- This toggles between the `slideDown()` and `slideUp()` methods.
  - If the elements have been slid down, `slideToggle()` will slide them up.
  - If the elements have been slid up, `slideToggle()` will slide them down.
Stop Animations

- The `stop()` method is used to stop animations or effects before they are finished.

- It works for all effect functions, including sliding, fading and custom animations.

- `stop()` syntax:
  
  ```javascript
  $(selector).stop(stopAll, goToEnd);
  ```
Stop Animations

- The **stopAll** parameter (optional) specifies whether the animation queue should also be cleared or not.

- The default is false, which means that only the active animation will be stopped, allowing any queued animations to be performed afterwards.

- The optional **goToEnd** parameter (optional) specifies whether or not to complete the current animation immediately. The default is false.

- So, by default, the **stop()** method stops the current animation being performed on the selected element.
jQuery Method Chaining

- With jQuery, you can chain together actions/methods.

- This allows you to run multiple jQuery methods (on the same element) within a single statement.

- This way, browsers do not have to select the same element(s) more than once.

- To chain an action, you simply append the action to the previous action.
jQuery Method Chaining

- This chains together the `css()`, `slideUp()` and `slideDown()` methods.

```javascript
$('#p1').css('color','yellow').slideUp(2000).slideDown(2000);
```

- The element with the id "p1" first changes to red, then it slides up, and then it slides down:
jQuery Method Chaining

- When chaining, the line of code could become quite long. However, jQuery is not very strict on the syntax; you can format it like you want, including line breaks and indentations.

- So this also works just fine:

```
$('#p1').css('color', 'red').slideUp(2000).slideDown(2000);
```

jQuery just throws away extra whitespace and executes the lines as one long line of code.
CS7026

jQuery DOM Manipulation
jQuery Manipulators

- All of the methods that we are going to look at manipulate the DOM in some manner.

- You can:
  - change one of the attributes of an element
  - set an element’s style properties
  - modify entire elements (or groups of elements) themselves - inserting, copying, removing, and so on.

- All of these methods are referred to as “setters,” as they change the values of properties.

- A few of these methods—such as `.text()`, `.html()`, and `.val()` - also act as “getters”, retrieving information from DOM elements for later use.
text(), html(), and val()

- Three simple, but useful, jQuery methods for DOM manipulation are:
  - `text()` - Sets or returns the text content of selected elements
  - `html()` - Sets or returns the content of selected elements (including HTML markup)
  - `val()` - Sets or returns the value of form fields

Today we will look at the first two.
text()

- **Gets** the combined text contents of each element in the set of matched elements, including their descendants.

- The result of the `.text()` method is a string containing the combined text of all matched elements.

- Note, due to variations in the HTML parsers in different browsers, the text returned may vary in new lines and other white space.

- This method does not accept any arguments.
text()

```html
<div class="demo-container">
  <div class="demo-box">Demonstration Box</div>
  <ul>
    <li>list item 1</li>
    <li>list <strong>item</strong> 2</li>
  </ul>
</div>
```

- The code `$('div.demo-container').text()` would produce the following result:
  Demonstration Box list item 1 list item 2
.text(textString)

- **Sets** the content of each element in the set of matched elements to the specified text.

- `textString`: A string of text to set as the content of each matched element.

- We need to be aware that this method escapes the string provided as necessary so that it will render correctly in HTML. It does not interpret the string as HTML.
.text(textString)

Consider the following HTML:

```html
<div class="demo-container">
  <div class="demo-box">Demonstration Box</div>
  <ul>
    <li>list item 1</li>
    <li>list <strong>item</strong> 2</li>
  </ul>
</div>
```

The code

```javascript
$('div.demo-container').text('<p>This is a test.</p>');
```

will produce the following DOM output:

```html
<div class="demo-container">
  <p>This is a test.</p>
</div>
```
.text(textString)

- It will appear on a rendered page as though the tags were exposed, like this:
  `<p>This is a test</p>`

- The .text() method cannot be used on input elements.

- For input field text, use the .val() method.
**html()**

- In a HTML document, `.html()` can be used to get the contents of any element.

- If the selector expression matches more than one element, only the first match will have its HTML content returned.
Consider this code:

```javascript
$('div.demo-container').html();
```

In order for the following `<div>`'s content to be retrieved, it would have to be the **first** one with class="demo-container" in the document:

```html
<div class="demo-container">
  <div class="demo-box">Demonstration Box</div>
</div>
```

The result would look like this:

```html
<div class="demo-box">Demonstration Box</div>
```
Some browsers may not return HTML that exactly replicates the HTML source in an original document.

For example, Internet Explorer sometimes leaves off the quotes around attribute values if they contain only alphanumeric characters.
html (htmlString)

- **Sets** the HTML contents of each element in the set of matched elements.

- `htmlString`: A string of HTML to set as the content of each matched element.

- When `.html(htmlString)` is used to set an element's content, any content that was in that element is **completely replaced** by the new content.

- Additionally, jQuery removes other constructs such as data and event handlers from child elements before replacing those elements with the new content.
Consider the following HTML:
```html
<div class="demo-container">
  <div class="demo-box">Demonstration Box</div>
</div>
```

The content of `<div class="demo-container">` can be set like this:
```javascript
$('div.demo-container')
  .html('<p>All new content. <em>You bet!</em></p>');</p>
```

That line of code will replace everything inside `<div class="demo-container">`:
```html
<div class="demo-container">
  <p>All new content. <em>You bet!</em></p>
</div>
```
Get Attributes - `attr()`

- The `attr()` method is used to get attribute values.

- E.g. this will get the value of the `href` attribute in a link:
  ```javascript
  $\("button\"\)\ .\ click(function(){
    alert($\(\"#tcd\"\)\ .\ attr(\"href\")\);
  })
  ```
The `attr()` method is also used to set/change attribute values.

This will set the value of the `href` attribute in a link:

```javascript
$("button").click(function()
    $('#tcd').attr("href","http://www.scss.tcd.ie");
});
```
Set Attributes - `attr()`

- The `attr()` method also allows you to set multiple attributes at the same time.

- This sets both the `href` and `title` attributes at the same time:

```javascript
$('button').click(function()
    $('#tcd').attr({
        'href': 'http://www.scss.tcd.ie',
        'title': 'School of Computer Science & Statistics'
    });
});
```
Add Elements and Content

- We will look at four jQuery methods that are used to add new HTML content:
  - `append()` - Inserts content at the end of the selected elements
  - `prepend()` - Inserts content at the beginning of the selected elements
  - `after()` - Inserts content after the selected elements
  - `before()` - Inserts content before the selected elements

- The difference between `.append()` and `.after()` is that `.append()` adds elements into the end of an element's contents, while `.after()` adds them directly after its closing tag.
append() and prepend()

- The `append()` method inserts content **at the end** of the selected HTML elements:
  `$("p") . append("Some appended text.");`

- The jQuery `prepend()` method inserts content **at the beginning** of the selected HTML elements.
  `$("p") . prepend("Some prepended text.");`
Add Several New Elements

- both the `append()` and `prepend()` methods can take an infinite number of new elements as parameters.

- The new elements can be generated with text/HTML, with jQuery, or with JavaScript code and DOM elements.
Add Several New Elements

Here the elements are created with text/HTML, jQuery, and JavaScript/DOM. Then we append the new elements to the text with the `append()` method:

```javascript
function appendText() {
    var txt1 = "<p>Text.</p>
    // Create element with HTML
    txt2 = $("<p></p>").text("Text.");
    // Create with jQuery
    var txt3 = document.createElement("p");
    txt3.innerHTML = "Text.";
    // Create with Javascript/DOM
    $("p").append(txt1, txt2, txt3);
    // Append the new elements
}
```
### after() and before()

- The **after()** method inserts content **after** the selected HTML elements.

$$\$("img").after("Some text after");$$

- The **before()** method inserts content **before** the selected HTML elements.

$$\$("img").before("Some text before");$$

- Like **append()** and **prepend()**, both the **after()** and **before()** methods can take an infinite number of new elements as parameters. The new elements can be generated with text/HTML, with jQuery, or with JavaScript code and DOM elements.
Removing Elements & Content

- To remove elements and content, there are mainly two jQuery methods:

  - `remove()` - Removes the selected element (and its child elements)
    ```javascript
    $('#div1').remove();
    ```

  - `empty()` - Removes the child elements from the selected element
    ```javascript
    $('#div1').empty();
    ```
Filter the Elements to be Removed

- The `remove()` method also accepts one parameter, which allows you to filter the elements to be removed.

- The parameter can be any of the jQuery selector syntaxes.

- The following example removes all `<div>` elements with `class="hello"`:
  ```javascript
  $('div').remove('.hello');
  ```
We will look at the following methods for CSS manipulation:

- **addClass()** - Adds one or more classes to the selected elements
- **removeClass()** - Removes one or more classes from the selected elements
- **toggleClass()** - Toggles between adding/removing classes from the selected elements
- **css()** - Sets or returns the style attribute
addClass()

- You can select multiple elements, when adding classes:

  ```javascript
  $("button").click(function(){
    $("h1,h2,p").addClass("blue");
    $("div").addClass("important");
  });
  ```

- You can also specify multiple classes within the `addClass()` method:

  ```javascript
  $("button").click(function(){
    $("#div1").addClass("important blue");
  });
  ```
removeClass()

- Works similarly to `addClass()`:

```javascript
$("button").click(function(){
    $("h1,h2,p").removeClass("blue");
});
```
toggleClass()

- This method toggles between adding/removing classes from the selected elements:

```javascript
$("button").click(function(){
    $("h1,h2,p").toggleClass("blue");
});
```
The `css()` method sets or returns one or more style properties for the selected elements.

To return a CSS property:

```javascript
css("propertyname");
```

E.g., the following will return the background-color value of the first matched element:

```javascript
$("p").css("background-color");
```
To set a CSS property:

```javascript
css("propertyname","value");
```

The following will set the background-color value for **all** matched elements:

```javascript
$("p").css("background-color","yellow");
```
Set Multiple CSS Properties

To set multiple CSS properties, use the following syntax:

```javascript
css({"propertyname":"value","propertyname":"value",...});
```

The following will set a background-color and a font-size for all matched elements:

```javascript
$("p").css({"background-color":"yellow","font-size":"200%"});
```