

# Appendix

## Showing and Hiding Elements

- To set a duration and a callback function
  - `show(duration, callback)`
    - duration is the amount of time taken (in milliseconds), and callback is a callback function jQuery will call when the transition is complete.
- The corresponding version of `hide( )`
  - `hide(duration, callback)`
- To toggle an element from visible to invisible or the other way around with a specific speed and a callback function, use this form of `toggle( )`
  - `toggle(duration, callback)`

## jQuery Sliding Effects

- The jQuery slide methods gradually change the height for selected elements.
- jQuery has the following slide methods:
  - `$(selector).slideDown(speed,callback)`
  - `$(selector).slideUp(speed,callback)`
  - `$(selector).slideToggle(speed,callback)`
- The speed parameter can take the following values: "slow", "fast", "normal", or milliseconds.
- The callback parameter is the name of a function to be executed after the function completes.

## jQuery Fading Effects

- The jQuery fade methods gradually change the opacity for selected elements.
- jQuery has the following fade methods:
  - `$(selector).fadeIn(speed,callback)`
  - `$(selector).fadeOut(speed,callback)`
  - `$(selector).fadeTo(speed,opacity,callback)`
- The speed parameter can take the following values: "slow", "fast", "normal", or milliseconds.
- The opacity parameter in the `fadeTo()` method allows fading to a given opacity.
- The callback parameter is the name of a function to be executed after the function completes.

## Creating Custom Animation

- Custom animation can be created in jQuery with the `animate()` function
  - `animate(params, duration, callback)`
  - `params` contains the properties of the object you're animating, such as CSS properties, `duration` is the optional time in milliseconds that the animation should take and `callback` is an optional callback function.

## jQuery Ajax features

- Allows part of a page updated
- Cross-Browser support
- Simple API
- GET and POST supported
- Load JSON,XML, HTML ...

## jQuery Ajax functions

- jQuery provides several functions that can be used to send and receive data
  - `$(selector).load()` : Loads HTML data from the server
  - `$.get()` and `$.post()` : Get raw data from the server
  - `$.getJSON()` : Get / Post and return JSON data
  - `$.ajax()` : Provides core functionality
- jQuery Ajax functions works with REST APIs, Webservices and more

## Loading HTML content from server

- `$(selector).load(url,data,callback)` allows HTML content to be loaded from a server and added into DOM object.
  - `$("#targetDiv").load('GetContents.html');`
- A selector can be added after the URL to filter the content that is returned from the calling `load()`.
  - `$("#targetDiv").load('GetContents.html #Main');`
- Data can be passed to the server using `load(url,data)`
  - `$("#targetDiv").load('Add.aspx',{firstNumber:5,secondNumber:10})`
- `load ()` can be passed a callback function

```
$("#targetDiv").load('Notfound.html', function (res,status,xhr) {  
    If (status == "error") { alert(xhr.statusText); }  
});
```



## Using get(), getJSON() & post()

- \$.get(url,data,callback,datatype) can retrieve data from a server.

```
$.get('GetContents.html',function(data){  
    $('#targetDiv').html(data);  
},'html');
```

- datatype can be html, xml, json
- \$.getJSON(url,data,callback) can retrieve data from a server.

```
$.getJSON('GetContents.aspx',{id:5},function(data){  
    $('#targetDiv').html(data);  
});
```

- \$.post(url,data,callback,datatype) can post data to a server and retrieve results.

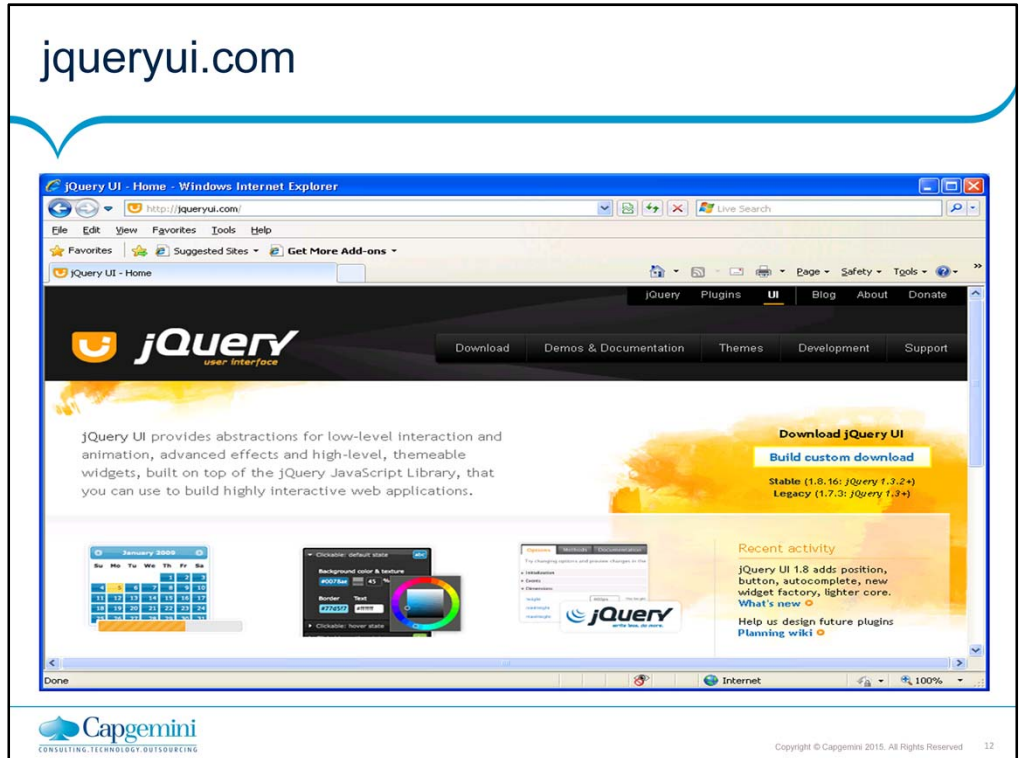
## Using ajax() function

- ajax() function is configured by assigning values to JSON properties

```
$.ajax({  
  url: "employee.aspx/GetEmployees",  
  data : null,  
  contentType: "application/json; charset=utf-8",  
  datatype: 'json',  
  success: function(data,status,xhr){  
    //Perform success operation  
  },  
  error: function(xhr,status,error) {  
    //show error details  
  }  
});
```

## jQuery UI

- jQuery offers a plug-in architecture that allows web developers to extend the core jQuery library (jqueryui.com).
- It has following plug-ins
  - Effects plug-ins
    - It make elements bounce, explode, pulsate, or shake.
  - Interaction plug-ins
    - It enable users to interact with elements by making those elements draggable, droppable, or sortable etc.
  - Widget plug-ins
    - Widgets saves tons of coding time and complexity while creating usable and responsive user interface elements.



## jQuery Plug-in Repository

- <http://plugins.jquery.com> is official jQuery plug-in repository, where jQuery plug-in developers submit their plug-ins.
- We can look for the best plug-ins from this repository.
- We can browse plug-ins by category, by name, date or by search for a string.
- We can get a valuable and versatile final product very easily and eventually we can tweak it to fit our needs from this repository.



## jQuery Plug-in naming convention

- `jquery.PLUGINNAME.js` is the recommended file name convention for jQuery plug-ins.
- For instance, if we create a plug-in to highlight elements, we need to name it as `jquery.highlight.js`
- Place the `index.html` in the same location, which shows the demo for using the plug-in.



## Closures

- A closure is used when a function is declared inside another function.
- A closure is the local variables for a function - kept alive after the function has returned.
- A closure is a stack-frame which is not de-allocated when the function returns.
- A closure in JavaScript is like keeping a copy of the all the local variables, just as they were when a function exited.



## jQuery Plug-in Types

- jQuery plug-ins can be broadly classified into two types.
  - Function Plug-in
  - Method Plug-in
- To make sure the plug-in doesn't collide with other libraries that might use the dollar sign, it's a best practice to pass jQuery to a self executing function (closure) that maps it to the dollar sign so it can't be overwritten by another library in the scope of its execution.

```
(function($) {  
    //jQuery Plug-in implementation  
})(jQuery);
```

## Function Plug-in

- Standalone function (\$.xxx) which does not return a jQuery object.
- All new functions are attached to the jQuery object
- It doesn't support method chaining.
- Function plugins directly extends the jQuery object.

```
(function($) {  
    $.PLUGINNAME = function() {  
        // Plugin Code  
    }  
})(jQuery);
```

## Method Plug-in

- The method (\$.fn.xxx) must return the jQuery object.
- All new methods are attached to the jQuery.fn object
- It supports Method chaining.
- Method plugins extends the jQuery.fn object.

```
(function($) {  
    $.fn.PLUGINNAME = function() {  
        return this.each(function() {  
            // code  
        });  
    }  
})(jQuery);
```

## Working with Plug-in Options

- Options were used to customize the behavior of the plugin
- Plug-in packed with options is quite flexible and easily extensible.
- To use options function definition would include an argument.
- The options object must be formatted following the inline object { option1: value1, option2: value2 } and so on.
- If the options were not provided then the plug-in will use it's default values

## Creating Private functions

- In order to make our functions to use only by plug-in only, we need to make them private, so that others cannot access and misuse them.
- Closures make it possible for the developer to avoid creating functions in the main namespace and keep them private to avoid problems with naming, backward compatibility issues.