DOM + AJAX + jQuery Exercises:

This example shows how you might structure an ajx js file. This example works with the weather api at Yahoo.

ajaxWeather.html:

```
<!DOCTYPE html>
<html lang="en">
<head>
   <meta charset="UTF-8">
   <title>Ajax Weather Worked example getting JSON data </title>
</head>
<body>
<p id="WeatherPlaceHolder"></p>
<script src="ajaxWeather.js"></script>
</body>
</html>
```

ajaxWeather,js:

```
   // create the XMLHTTPRequest object
   var xhr = ( typeof window.XMLHttpRequest == "function" )
         ? new XMLHttpRequest( )
         : new ActiveXObject("Microsoft.XMLDOM") ;

   // This is our function to process the returned data
   // NB in this example the data is returned in JSON format
   function processWeatherData( jsonResults ) {
      var jsonData  = JSON.parse(jsonResults);
      var forecastData = jsonData.query.results.channel.item.forecast;

      var ulListHtml = "<ul>";
      forecastData.forEach( function( item ) {
                  ulListHtml += "<li>" + item.date + " " + item.text + "</li>";
      } );
      ulListHtml += "</ul>";
      document.getElementById( 'WeatherPlaceHolder' ).innerHTML = ulListHtml;
   }

   // This is our callback function to process the returned data
   function getWeatherCallBack() {
      if (xhr.readyState == 4) {
         if (xhr.status == 200) {
            processWeatherData( xhr.responseText );
         } else {
            document.getElementById( 'WeatherPlaceHolder' ).innerHTML = "XHR failed:" +
xhr.statusText + " status:" + xhr.status;
         }
      }
   }
```

```
   function getWeather() {
      var url = "https://query.yahooapis.com/v1/public/yql?q=select * from weather.forecast
where woeid in (select woeid from geo.places(1) where text='glasgow, uk')&format=json"
      xhr.open("GET", url, true);
      xhr.setRequestHeader("Accept","application/json; charset=utf-8");
      xhr.onreadystatechange = getWeatherCallBack;
      xhr.send(null);
   }

   document.addEventListener( 'DOMContentLoaded', function() {
      if ( typeof( document.getElementsByTagName ) == "undefined" ) {
         document.write( "getElementsByTagName is not supported" );
         exit;
      }
      // getting here means everything is ok - so
      getWeather();
   }, false );
</script>
```

ajaxWeather,css:

create appropriate styling for the ul/li tags in the output html and remember to add code to you html file to link in the css file.

Q 1. Take a look at the following HTML document:

```
<!DOCTYPE html >
<html lang=”en”>
<head>
<title>AJAX Tutorial: JavaScript Events and DOM</title>
</head>
<body>
<p>
Hello World! Here's a cool list of colors for you:
</p>
<ul>
<li>Black</li>
<li>Orange</li>
<li>Pink</li>
</ul>
</body>
</html>
```

Listing ex1a.html

Suppose that you want to have the <ul> element and its children generated dynamically using JavaScript and DOM. The first step is to create a placeholder in their place. This placeholder must have an **id**, so that it can be then identified (found) by your JavaScript code. If the element is used as a placeholder, then your page could look like this:

```
<!DOCTYPE html >
<html>
  <head>
```

```
    <title>AJAX Tutorial: JavaScript Events and DOM</title>
  </head>
  <body>
    <p>
      Hello World! Here's a cool list of colors for you:
    </p>
    <div id="myUnorderedList"></div>
  </body>
</html>
```

Listing ex1b.html

Here's an exercise for you to try using lb.html given above:

1.a) Access the named element programmatically from a JavaScript function. Have the JavaScript code execute after the HTML page loads, so that it can access the element to add the list of colours – e.g. red, green & blue

1.b) OK that was straightforward. Use your existing code as the basis of getting the colours from a file called colours.txt using AJAX. The format of the file is just text eg red green blue etc. You should again get a list of colours displayed in the browser.

1.c) Using AJAX let's get the colours from file colours.phtml where the format of the file is an unordered list of colours in html.

1.d) You can try this one if you're ok with XML file structures. Using AJAX let's get the colours from file colours.xml where the format of the file is an list of colours in an appropriate xml format e.g.

```
<colours>
   <colour>black</colour>
   <colour>blue</colour>
   <colour>yellow</colour>
   <colour>green</colour>
</colours>
```

Again, display the colours as an unordered list.

1.e) You can try this one if you're ok with JSON file structures. Finally using AJAX let's get the colours from file colours.json where the format of the file is an list of colours in an appropriate json format e.g. ["black", "blue", "yellow", "red"]. Again, display the colours as an unordered list.

N.B. Remember these techniques for accessing data via AJAX as they are fundamental in being able to process and display data retrieved in this way.

Q 2. This simple exercise demonstrates a few techniques of using CSS with JavaScript, which is the most relevant scenario in the context of AJAX development. For this exercise you need a table and a couple of buttons named 'Set Style 1' and 'Set Style 2' that change the table's colors and appearance by just switching the current styles. The code that executes when the first button is clicked is as follows:

```
// Change table style to ‘style 1’
function setStyle1( )  {
  // obtain references to HTML elements
  var oTable = document.getElementById("table");
  var oTableHead = document.getElementById("tableHead");
  var oTableFirstLine = document.getElementById("tableFirstLine");
  var oTableSecondLine = document.getElementById("tableSecondLine");

  // set styles
  oTable.className = "Table1";
  oTableHead.className = "TableHead1";
  oTableFirstLine.className = "TableContent1";
  oTableSecondLine.className = "TableContent1";
}
```

The TableStyles.css file contains two sets of styles that can be applied to the table in ex2CSS.html. When the user clicks one of the Style buttons, the JavaScript DOM is used to assign those styles to the elements of the table. You need to write a similar function setStyle2 which sets the second style when the second button is pushed.

Q 3. Build a web page which displays the contents of a file in an alert box when :

- a button is pushed
- the page is loaded

Q 4. Write a routine called getData which has 2 parameters. The first parameter is the name of the file to be retrieved and the second parameter is the name of the DIV tag where the contents of the file are to be displayed. A typical call could be as follows:

```
getData( ‘test.html’, ‘content’ );
```

Here’s the HTML file that you can use to test your new routine:

```
<!DOCTYPE html>
<html lang="en">
 <head>
  <title> Dom AJAX Test</title>
  <script type="text/javascript">  <!-- type not needed in modern browser -->
   function getData( dataSource, divID )  {
     // place all your code here for the new function
   }
  </script>
 </head>
<body>
<h1>Fetching data with Ajax</h1>
<form>
        <input type="button" value="Display Message"
        onclick="getData('test.html', 'targetDiv')">
</form>
<div id="targetDiv">
```

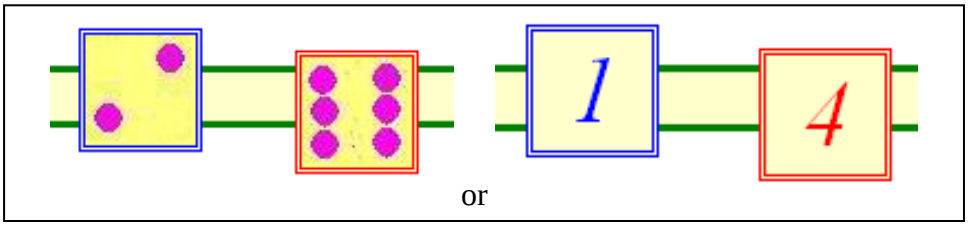
```
<p>The fetched data will go here.</p>
</div>
</body>
</html>
```

Q 5. Use Listing above as the basis for getting the contents of an XML file called text.xml (see below) displayed on the page ie your page should show Hello World!

```
<?xml version="1.0" encoding="utf-8" ?>
<message>
        Hello World!
</message>
```

Also use a css file to style your message!

Q 6. Build a web dice viewer application showing output of the dice pair eg



Q 7. Another worked example using jQuery this time:

```
Using jQuery to modify the CSS. We have 3 files: ex7.html, ex7.css & ex7.js

<html>
    <head>
        <title>jQuery Examples</title>
        <link rel="stylesheet" href="ex7.css" />
        <script src="http://***any CDN you fancy for jquery></script>
        <script src="ex7.js"></script>
    </head>

    <body>
        <div class="coloured">
        </div>
        <form>
            <input type="button" id="red" value="red"/>
            <input type="button" id="green" value="green"/>
            <input type="button" id="blue" value="blue"/>
            <br />
            <input type="button" id="round" value="round" />
            <input type="button" id="unround" value="unround" />
            <br />
            <input type="button" id="toggle" value="toggle" />

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

Listing ex7.html

```
.coloured {
    width : 200px;
    height: 200px;
    position: absolute;
```

```
    top: 40%;
    left: 40%;
    background-color: red;
}
```

Listing ex7.css

```
$('document').ready(function() {

    var div = $('div.coloured');

    $('#red').click( function() {
        div.css('background-color', 'red');
    });

    $('#blue').click( function() {
        div.css('background-color', 'blue');
    });

    $('#green').click( function() {
       div.css('background-color', 'green');
    });

    $('#round').click( function() {
       div.css('border-radius', 10);
    });

    $('#unround').click( function() {
       div.css('border-radius', 0);
    });

    $('#toggle').click( function() {
       div.fadeToggle();
    });

});
```

Listing ex7.js

That's All Folks!