|  |  |
| --- | --- |
|  | **CE212 Lab 3: More JavaScript**  **Introduction**  The purpose of this lab is to introduce some JavaScript functions and techniques that will be needed for the first assignment.  The exercises are fairly short; after completing them you should start (or continue) working on the assignment.  We are going to use jQuery in this lab so you will need to obtain a copy of the [**jQuery javascript file**](https://moodle.essex.ac.uk/pluginfile.php/857857/mod_resource/content/3/lab3/jquery-3.3.1.min.js) and place it in the folder you are using for this lab.  **Using Drop-Down Menus**  This example demonstrates how to change the appearance of part of a document by using a drop-down menu.  Create or open an IntelliJ project and create a new HTML file with the following body.  <h3>Drop-down menu Test</h3>  <p id="para" style="color:red">Select a colour for this text</p>  <p>The colour of this paragraph will not change</p>  <select>  <option value = "red"> Red </option>  <option value = "green"> Green </option>  <option value = "blue"> Blue </option>  </select>  Preview the HTML display in Firefox - a drop-down menu should be visible and it should be possible to select an item but doing so will not cause any action.  Add inside the header of the HTML file the following JavaScript function declaration.  <script src="jquery-3.3.1.min.js"></script>  <script type="text/javascript">  function setCol(col) {  var par = $("#para");  par.css("color", col);  }  </script>  You may need to change the source to something like ../js/jquery-3.3.1.min.js if you are storing JavaScript files in a separate folder.  We wish to invoke this function to change the colour of the paragraph when the menu selection is changed. One way to do this is to add an onChange event attribute to the select tag:  <select onChange="setCol(value)">  Add this attribute to the tag and check that selecting from the menu does indeed now change the colour of the paragraph.  **Drop-down menus in Forms**  We now wish to see how to use a form with more than one drop-down menu, the actions being performed when a submit button is pressed.  Remove the onChange attribute that you have just added.  Place <form> and </form> tags around the <select>...</select> element and within the form element add another select element with three options whose values are left, right and center. Give the two select tags unique id attributes, e.g. <select id="col">.  Add to the form a submit button using  <input type="submit" value="Submit" onclick="setStyle();return false" />  Note the use of return false - this is to suppress any attempt to send the form contents to a server.  The setStyle function must now be written in the script in the document header. This will be similar to setCol but needs to set the CSS to something like color:red; text-align:left. To retrieve the selections from the menus we use the value attribute of the select element, which we can obtain by applying the val method to a jQuery object representing that HTML element. Hence you will need to use code such as  var col = $("#col").val();  **Loading Documents**  The next example we shall study is how to load another HTML document in a window and then use JavaScript to modify its display. We will be using the **Lists.html** file from lab 2 so you will need to open the IntelliJ project containing that file.  In the same folder as **Lists.html** create a new HTML file with the following body  <a href="javascript:openFile('Lists.html')">Open Lists.html in Popup</a>  <br/>  <a href="javascript:changeStyle('')">Default</a>  <br/>  <a href="javascript:changeStyle('navcontainer')">Inline</a>  Place the following script in the document header.  var w;  function openFile(url) {  w = window.open();  w.location = url;  }  function changeStyle(style) {  w.document.getElementById("myList").setAttribute("class", style);  }  Preview the document and click on the link.  You should see that a new popup window appears and **Lists.html** is loaded in this new window - this has been done by setting the location property of the window. You can change the style of the lists in the pop-up window by clicking on one of the options within that window but also by clicking on one of the links in the parent window.  **Retrieving and Modifying Text Elements**  It would be easy to load a different file instead of **Lists.html**, simply by asking the user to supply a URL and passing this as an argument to the call to openFile. (Note that this approach can be used only if the document to be loaded is on the same server as the document that contains the script.) However, when loading an arbitrary document we cannot use the getElementById function since we do not know what id attributes its tags might have. We could retrieve elements with particular tags using getElementsByTag but this will not help if we need to retrieve all text nodes from a document. To do this we will need to recurse over the entire body of the document.  Create a new HTML document with the following body  <p>Filename: <input id = "url" name="url" size=15 type="Text"/></p>  <a href="javascript:openFile(document.getElementById('url').value)">  Open document</a>  </br>  <a href="javascript:upperCase(w.document.body)">To upper case</a>  Copy the openFile script from above into the header of this document. (The changeStyle function is not needed this time.)  We need a file whose name we can supply as input, so copy the file [**Silly.html**](https://moodle.essex.ac.uk/pluginfile.php/857857/mod_resource/content/3/lab3/Silly.html) into the same folder as the current HTML file. Preview the current HTML document and type Silly.html into the text field then click the open document link; you should see a document with rather silly content in a popup window. [Clicking on the upper case link will not yet do anything - we need to provide the upperCase function.]  Add to the script the following function  function upperCase(node) {  if (node.nodeType==Node.ELEMENT\_NODE) {  for (var m = node.firstChild; m!=null; m = m.nextSibling)  upperCase(m);  }  else if (node.nodeType==Node.TEXT\_NODE) {  node.data = node.data.toUpperCase();  }  }  Note that this function is called with the body element of the document in the popup window as an argument. It recursively descends through all of the children of this element, and when it finds a text node it changes its contents to upper case. (There are other types of node as well as element nodes and text nodes but these cannot have text nodes as descendents so they can be ignored.)  Check that the function works correctly.  **Simple exercise**: Add another function to convert text to lower case and a link to invoke it.  We now wish to retrieve the contents of all text nodes and display them in the parent window. Add the following to to the end of the HTML body.  <a href="javascript:retrieveText()">Retrieve text</a><br/>  <pre id="count"> &nbsp; </pre><br/>  Add the following to the script.  var counter;  var pre;  function retrieveText() {  pre = document.getElementById("count");  counter = 0;  retrieve(w.document.body);  pre.textContent = "Found "+counter+ " text nodes";  }  We need to write the recursive function retrieve that is called by retrieveText - it should have a structure similar to upperCase but when it finds a text node it should create a new paragraph element (i.e. <p> tag) using document.createElement, append to that element a copy of the text node's data and insert this paragraph in the document immediately before pre.  You will need to use the function insertBefore - when applied to a node this inserts a new child (specified by its first argument) immediately before the child specified by its the second argument. The function will have to be applied to the parent of the <pre> tag, i.e. pre.parentNode.  **Exercise - Using Arrays**  An alternative approach for generation of the display of text nodes in the previous program is to store all of the nodes in an array of strings as they are retrieved and then display the array in the retrieveText function after returning from the call to the retrieve function.  Modify the script to use this approach. This time the array should be displayed as a list instead of using separate paragraphs.  **Hints:**  Arrays in JavaScript can change size dynamically. An empty array can be declared using a declaration such as var myArr = [];, and we can then add an extra element to the end using code such as myArr[myArr.length] = myString; .  To display the list place a <ul> tag with a unique id in the HTML, retrieve this using getElementById, and then for each element in the array add an li item with a textnode child to this tag using code similar to that previously seen for adding table data items to table rows. |