|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **CE212 Lab 1: Getting Started**  **Introduction**  The purpose of this lab is to introduce simple examples of the main client-side parts of a web application, namely:   * HTML * CSS * JavaScript   During the lab you will create an IntelliJ project to manage and edit your files, link HTML with CSS and JavaScript files, and write JavaScript code to dynamically add new HTML elements to a document.  You should aim to finish the lab before the start of next week's lab session if you don't complete it within the allotted lab time.  The version of IntelliJ used to generate the screenshots is older than the one in the labs so what you see will not correspond precisely to the screenshots.  Some of the terms used in the lab will inevitably precede their lecture coverage; don't worry about this; learning them in the lab as you go along will make them more familiar when covered in lectures later in the course.  We'll begin with a walkthrough of the project structure, and a simple example of a dynamically created Celsius / Fahrenheit conversion table.  **IntelliJ**  This is a very advanced IDE, and during this course you need use only a small fraction of its features.  Here are some shortcuts Simon Lucas (who wrote the original version of this lab) finds very useful:   |  |  | | --- | --- | | **Shortcut** | **Description** | | <CTRL> F4 | Close the Current Pane | | <CTRL> B | Go to definition of item under cursor (useful for navigation) | | <CTRL><ALT><LeftArrow> | Go back to the previous place (e.g. before you hit <CTRL>B) | | <CTRL><ALT>L | Perform automatic layout of content in current pane | |  |  |   *What if I'm happier using TextPad?*  In the long run, you're better off with a good IDE.  An IDE understands the structure of your application and will warn you about any broken cross references.  The main disadvantage of IntelliJ is that it is not freely available. You are also free to use other IDEs on this course.  NetBeans and Eclipse are free, and both good.  Eclipse is probably the better IDE overall, but NetBeans makes it especially easy to start web applications..  IntelliJ gives code completions for HTML, Java, JavaScript and CSS, plus some others.  This is very helpful when exploring a new language e.g. see picture below:  **Directory Structure**  When building web applications it helps to layout the code in an organised way.  The structure shown here is enough for simple applications (of the type covered in this course).  The src folder would normally contain the Java source tree used for a project; this is currently empty since we are not writing any server-side code in this lab.  **Sample Application**  Create a new folder for CE212 on your M: drive.  We'll now look at a simple example of JavaScript code.  The chosen example is the construction of a temperature conversion table, showing conversions of Celsius to Fahrenheit.  This will create output in a table looking something like the following:  This will be a simple application with no interactivity, but it at least creates some content programmatically.  The steps are as follows:   1. Start Intellij Idea 2. Select **Create New Project**. 3. Accept all the defaults and advance through the screens using **Next**; choose any appropriate project name; the project location should be a new sub-folder of the CE212 folder on your M: drive (this sub-folder will be created automatically) 4. Right-click on the project folder in the project view pane (in the project view diagram above the project had been given the name start - yours will probably have a different name) and select **New - Directory** to create a new folder which should be called resources. Right-click on the resources folder icon three times to create sub-directories called css, html and js. 5. We are now ready to create the files.   **HTML file**  Create a file Converter.html in the html folder (by right-clicking and selecting **New - HTML File**; when supplying the name you don't need to supply the.html extension - it will be added automatically).  The HTML to be placed in this file is provided here for you; you can cut and paste the various bits into the template generated by IntelliJ.  <!DOCTYPE html>  <html lang="en">  <head>  <link rel="stylesheet" type="text/css" href="../css/converter.css"/>  <title>Celsius to Fahrenheit Converter</title>  <script language="JavaScript" src="../js/c2f.js" type="text/javascript">  </script>  </head>  <body onload="conversionTable('conversion', 0, 30);">  <h2>Celsius to Fahrenheit Converter</h2>  <div id="conversion">  </div>  </body>  </html>  The HTML will be automatically formatted, and you should see something like  Note that IntelliJ highlights the yet undefined JavaScript and CSS files.  Create these two files in the appropriate folders (using **New - Stylesheet** for the CSS file) and the highlighting should vanish.  A common idiom for adding dynamic content to a web page is to catch the onload  event which is triggered when the browser has finished loading and parsing the web page.  In our file this will call the function conversionTable() which will be defined in c2f.js.  A division element (<div id="conversion">) is used as a placeholder for where the dynamically created HTML will be placed.  In the above HTML, note the following:   * the links to the external CSS and JavaScript files * the onload attribute of the <body> tag used to invoke the conversionTable function * single quotes are used to wrap the string ('conversion') passed as an argument as we are inside a double-quoted HTML attribute string. (JavaScript, like Python, allows strings to be written using single or double quotes.) * the fact that 'conversion' matches the id attribute of the div tag   **CSS file**  The following is an example of a CSS file.    Type some of these CSS declarations into the file converter.css. (The h2, body, pre, th and td ones are the only ones we will need for this lab); IntelliJ will check the legality of your CSS - try mis-spelling center as centre and see what happens.  We now wish to view the page in Firefox. Right-click on Converter.html in the project view pane and select **Open in Browser** and choose Chrome or Firefox. A preview of the web page should appear in your browser; since the JavaScript has not yet been written all that appears will be the header.  Experiment with some changes to the style for h2 in the CSS file and see how the appearance changes when the browser page is refreshed.  **JavaScript**  Before looking at how to dynamically create the HTML for the conversion data, let's first consider the nature of the data that we're aiming to create.  Our first solution will put everything in a <pre> tag, with lines separated by <br> elements.  Therefore, we need to create HTML like the following and place it inside the <div> tag.  <div id="conversion">  <pre>  0 : 32  1 : 33.8  2 : 35.6  3 : 37.4  4 : 39.2  </pre>  </div>  Temporarily copy and paste this into the Converter.html file and use the preview pane at the bottom of this file to see what the page will look like. Experiment with the CSS styles for pre if you wish.  Remove the temporary code; we will be using JavaScript to generate the data; the JavaScript is shown below.  Observe the use of:   * document.getElementById(tagId) - used to retrieve a uniquely-identified node from the current document (the document that loaded this external JavaScript file) * document.createElement(tagName) - creates a new element to add to a document * node.appendChild - where node is a reference to an HTML element, adds a child element to the end of its list of children (e.g. pre.appendChild will append a new element inside the <pre> tag.)     Spend some time studying each of these files.  Type this JavaScript into the c2f.js file. Note how IntelliJ handles completions as you type.  Check that the table is now displayed correctly in the browser; you may need to refresh the page. If the table is not displayed there must be an error in the JavaScript file - check it carefully. The display should look something like the following screen shot.    **JavaScript Exercise**  The above application produces rather poor looking output.  We want to rewrite the JavaScript in order to produce a table, rather than the current lazy method of putting all the output data in a <pre> element.  The aim is to produce a table that looks something like this:    Hint: define CSS for even and odd table rows to get the alternative background effect (.odd and .even).  For example, possible CSS for even rows could be:  tr.even {background: #EEFFEE}  The HTML that you're trying to create could look like this:  <table>  <thead>  <tr>  <th>Celsius</th><th>Fahrenheit</th>  </tr>  </thead>  <tbody>  <tr class="even"><td>0</td><td>32</td></tr>  <tr class="odd"><td>1</td><td>33.8</td></tr>  <tr class="even"><td>2</td><td>35.6</td></tr>  <tr class="odd"><td>3</td><td>37.4</td></tr>  <tr class="even"><td>4</td><td>39.2</td></tr>  </tbody>  </table>  To set an HTML attribute from JavaScript, use setAttribute e.g.:  var r = document.createElement("tr");  if (i%2==1) r.setAttribute("class", "odd");  The browser output should now look something like  **Summary**  In this lab we covered setting up an IntelliJ project and adding files to it. IntelliJ has a good understanding of the syntax plus cross references used within your web application.  This makes it much easier to write legal code and should enhance your learning experience.  During the exercise we studied how to add dynamically created content to a web page, using a uniquely identified <div> tag, catching the onload event of <body>, and using JavaScript to manipulate the content of the tag.  We also linked to an external CSS file to style the HTML.  end of page |