

CSC309, Summer 2007, Assignment 1

In this assignment you will familiarize yourself with several client-side web programming technologies by implementing a simple but very useful web-based XHTML editor. The central idea is to edit the XHTML code in one frame and see the formatted results in another frame. Then, extend this basic editor with some extra functionality to manipulate individual DOM elements, support browser-specific behavior, and load another file into the editor.

Make sure that:

- Your code works on the CDF lab machines under Firefox and Konqueror browsers. (No other browsers need to be supported.)
- Your code is kept out of sight at all times. Do NOT deploy your code on a web server (not even the web server in your CDF account) since that would make the source viewable by classmates and therefore it would be a breach of academic integrity. When using the CDF machines to test your code, restrict permissions to your assignment files and also the directory where you put your files, so only you can read and execute the files. These restrictions must be in effect until at least 3 full days after the assignment submission time.
- Submit all your source files named as follows: index.html, editor.html, display.html (XHTML files), editor.css (CSS file), and editor.js (if you decide to use a separate JavaScript file rather than embedding JavaScript into your page) for Questions 1, 2, and 4; Fetch.java (a Java source file) and editor.html (XHTML file) for Question 3.
- Submit a text file question0.txt with the answer for Question 0.
- Submit XHTML and not HTML. There are good XHTML validators available (e.g. <http://validator.w3.org/>).

Question 0.

Indicate your favorite references for the syntax of XHTML, CSS, JavaScript, and Applets (could be any reference including book, article, online, etc.). [1%]

Question 1.

1. Create a framed XHTML page with two frames: call them 'editor' and 'display'. The editor frame takes the left half of the page and the display frame takes the right half of the page. For the editor frame, use a file editor.html that you are going to create in the subsequent questions. [10%]
2. Design the page using a CSS file. Using the CSS, give all text areas some dark grey background and white text color. Set the background of the editor.html page light blue. In editor.html, create a text area for editing. [10%]

3. Create a JavaScript function `copyTo(ID)` that copies the content of the editing text area into an element of the display frame specified by the respective ID, and, conversely, a function `copyFrom(ID)` that copies the content of the specified element of the display frame into the editing text area. When the ID is 'body', refer to the body of the display frame. (Hint: use the `innerHTML` attribute of the element.) [10%]
4. Create a Load button on the editor page that loads the contents of the display frame into the editing text area by invoking the `copyFrom` function and a Display button that copies the contents of the editing text area into the display frame by invoking the `copyTo` function. Use JavaScript events and the `copyFrom` function to prepopulate the editing text area with the contents of the display frame when the page is loaded. [10%]

Question 2.

1. Implement a function `getElementIDs()` that retrieves all the defined element IDs in the XHTML of the display frame. Create a pulldown menu in the editor frame that is filled with all available element names when invoking `getElementIDs`. [10%]
2. Invoke `getElementIDs` when the pulldown is pulled. Invoke `copyFrom` when selecting an element ID in the pulldown. Refine your implementation of Question 1.4 as follows: Invoke `copyTo` on the selected element ID when the space or enter key is pressed. [10%]

Question 3.

In this task you will invoke a function implemented in a Java applet from JavaScript and act upon the return value. This will effectively allow loading text files (e.g. HTML) into the editor without reloading the page.

1. Create a Java applet that implements a function `getPage(String URL)`, which returns the content of the file referred by the URL. Embed the applet into the page. [10%]
2. In the editor frame, add a text input field where the user can enter a URL and a link that invokes the `getPage` function on the entered URL. Replace the content of the editing text area by the content of the retrieved file. Then invoke `copyTo` to also update the display frame. [10%]

Question 4.

1. Create three text input fields in the editor frame and label them 'element', 'attribute', and 'value'. Create functions `getDOM(ID, Attr)` and `setDOM(ID, Attr, Value)` that get, respectively set, the attribute 'Attr' of the element 'ID' in the display frame. Add two links to the editor frame to invoke these functions. [10%]
2. Since certain browsers require special treatment for the style attribute, create three text input fields 'element', 'style-attribute', and 'value' and two functions get and set invoked

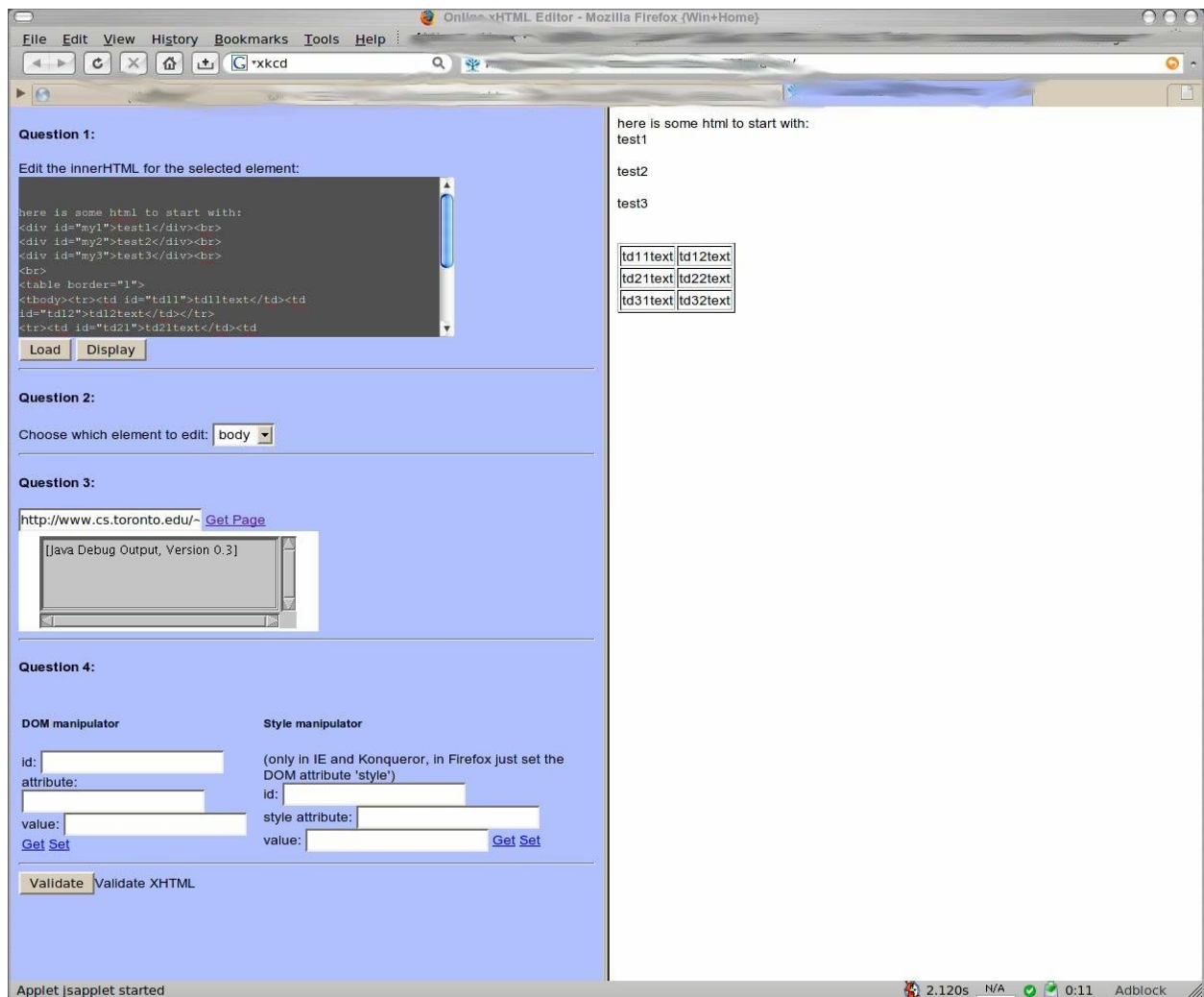
by corresponding links. (This part of your solution only needs to be demonstrated on the Konqueror browser in the CDF lab.) [10%]

3. Create a button Validate on the editor frame that submits the content of the textarea of the editor frame for validation at <http://validator.w3.org/>. Display the results of the validation (HTML reply) in a new window. Hint: Use the action of the "Validate by direct input" section of the online W3 validator as action for your form of the editor frame to send the content of your editor frame textarea for validation by the W3.org validator. [10%]

Notes:

The assignment is marked out of 100%, i.e. up to 11% bonus is available.

Here is a sample of the screen for the completed solution.



Good luck and have fun!