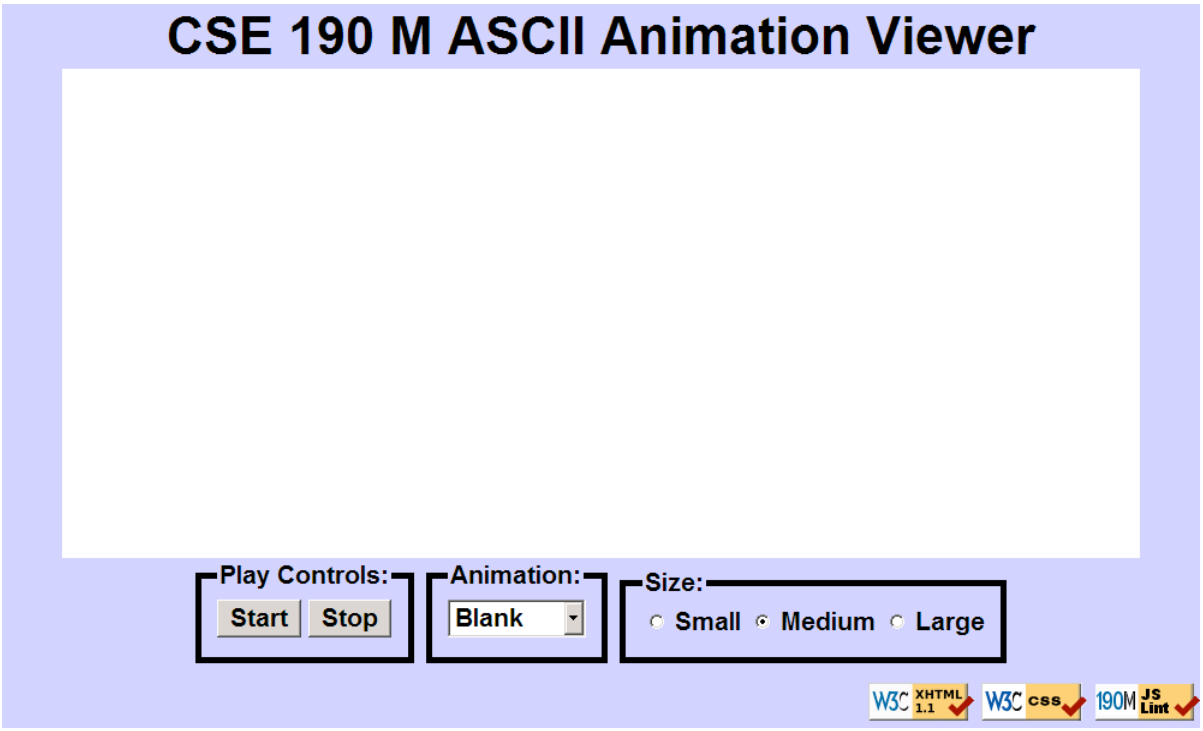
University of Washington, CSE 190 M
Homework Assignment 5: ASCIImation

Special thanks to Dave Reed of Creighton University for the original idea of this assignment.

This assignment tests your understanding of JavaScript and its interaction with HTML user interfaces. You must match the appearance and behavior of the following web page:



ASCII art is pictures that consist of text characters. ASCII art has a long history as a way to draw pictures for text-only monitors or printers. We will draw animated ASCII art, or "ASCIImation." Groups of nerds are working to recreate the entire movies Star Wars and The Matrix as ASCIImation.

The first task is to create a page **ascii.html** with a user interface (UI) for creating/viewing ASCIImations. The HTML page (skeleton provided) links to a provided CSS style sheet **provided.css** that you don't need to modify. It also links to a style sheet **ascii.css** that you should create to finish styling the page. After creating your page, you must make the UI interactive using JavaScript in **ascii.js** so that clicking the UI controls causes appropriate behavior.

The final part of your task is to create an ASCIImation of your own, stored in a file named **myanimation.txt** (and **myanimation.js**). Your ASCIImation must show non-trivial effort, must have multiple frames of animation, and must be entirely your own work. Be creative! We will put students' ASCIImations on the course web site for others to see.

In total you will turn in the following files:

- **ascii.html**, your web page
- **ascii.css**, the style sheet for your web page
- **ascii.js**, the JavaScript code for your web page
- **myanimation.txt**, your ASCII animation as a plain text file
- **myanimation.js**, your ASCII animation as JavaScript code (so it can be used on the page)

Our screenshots were taken on Windows XP in Firefox 3, which may differ from your system.

Appearance Details:

Under the page's heading is a text box with 80 columns and 20 rows, centered horizontally. Its width is 90% of the page size and height is 400px. It has no border and a 12pt monospace font. The CSS `width`/`height` properties will ultimately be responsible for its exact size on-screen, but you must include `rows`/`cols` attributes in your `textarea` HTML element for the page to validate.

Below the large text box is a set of controls grouped into field sets with 5px black borders. These field sets appear on the same line. *(Hint: To do this, see textbook section 4.4.4 about Element Visibility.)* The behavior of the controls inside them is described below.

Behavior Details:

The following are the groups of controls at the bottom of the page and the controls' behavior. (**NOTE:** Although we put controls inside a form in the last assignment, you **should NOT use a `form` tag** on your page this time.)

When the page is idle, all frames of the animation are visible. Frames are separated by 5 equals signs and a line break (`\n`) character.

Play Controls:

Start: When clicked, animation begins. When animation starts, whatever text is currently in the text box is broken apart to produce frames of animation. (This might be a pre-set animation, or text that the user has typed manually, or some modified version of a pre-set animation.) During animation, one frame is visible at any moment, starting with the first frame. The animation changes frames once every 200ms. When the animation reaches the last frame, it loops back around and repeats indefinitely. If the user clicks Start again while the animation is already running, it should not have any effect on the running animation, or break the animation behavior.

Stop: When clicked, halts any animation in progress. When animation is stopped, the text that was in the box before animation began is returned to the box. (Again, this may have been previously modified by the user.) The user should be able to click Stop multiple times in a row without breaking the animation behavior.

Font Size:

Contains three radio buttons. When one of these buttons (or the text next to it) is clicked, it immediately sets the font size in the main text area to small (7pt), medium (12pt), or large (24pt). Initially the medium button is checked and the text is 12pt in size. Only one of the font size buttons can be selected at a time. If the animation is playing when one of these buttons is clicked, the font size changes immediately while animation continues.

Animation:

A drop-down list of ASCII animations. When one of the animations is chosen (`onchange`), the main text area updates to display all text of the chosen animation. The choices available are: Blank, Exercise, Juggler, Bike, Dive, Custom. Initially the Blank animation is selected and no text is showing in the text entry box.

The **ascii.html** page links to a provided file **animations.js** that declares the ASCIImations String variables named `exercise`, `juggler`, `bike`, and `dive`. You shouldn't edit this file, but your **ascii.js** file can refer to these variables. For example, if you have a `textarea` on your page with an `id` of `mytextarea`:

```
$("mytextarea").value = juggler;
```

The **animations.js** file also defines a global associative array named `ANIMATIONS` whose keys ("indexes") are the names of the animations, and whose values are the entire text of the corresponding animation. So the `"Bike"` and `"Exercise"` keys map to those respective animations, and so on. Using this array can help you to avoid redundancy in your program. Here is a short example that uses the `ANIMATIONS` array:

```
var whichOne = "Juggler";
$("mytextarea").value = ANIMATIONS[whichOne];
```

The user may type new text in the text area after choosing a pre-set animation. The animation shown when Play is pressed should reflect any changes made by the user. (In other words, you should wait to capture the text to animate from the text area until the user has pressed the Start button.)

You may assume that the user will stop any running animation before changing animations. In other words, the user will not try to type in the text area or use the selection box to switch animations while animation is in progress.

Custom Animation:

You will submit your own custom animation in two separate files. The first, **myanimation.txt**, should be a **plain-text** version of your animation, such that it could simply be pasted into the text area on your ASCIImation page and be run as an animation. The second file, **myanimation.js**, should be a copy of the same animation as a **JavaScript string**. It should be used to populate the text area with your custom animation when the Custom choice in the Animation box is selected.

You can use the StringMaker tool linked on the web site to convert your plain-text **myanimation.txt** art into a JavaScript string you can put into **myanimation.js** and use in your program. Don't put any comments or other information in **myanimation.txt**; it should consist solely of your animation in plain text.

Development Strategy and Hints:

1. Edit the **XHTML** file to add the proper UI controls.
2. Write your **CSS** code to achieve the proper layout.
3. Write a small amount of **"starter" JS code** and make sure that it runs. (For example, make it so that when the Start button is clicked, an `alert` box appears.)
4. Implement code to change the **animation text and font sizes**. Make it so that when an option is chosen in the selection box, the proper text string appears in the text area. Get the font size radio buttons working.
5. Implement a **minimal Start behavior** so that when Start is clicked, a single frame of animation is shown. Clicking Start multiple times will show successive frames of animation.
6. Use a JavaScript **timer** to implement the proper animation based on your previous code.

We strongly recommend that you install and use the **Firebug** add-on for Firefox on this assignment. Inspect elements to fix your styles. It also shows **syntax errors** in your JavaScript code. You can use it as a debugger, set breakpoints, type expressions on its Console, and watch variables' values. Firebug is essential for JS programming!

Our **JSLint** tool can help you find common JavaScript bugs. Since this is your first JavaScript program, you will probably encounter frustrating problems. If so, paste your code into JSLint to look for errors or warnings.

Implementation and Grading:

Submit your assignment online from the course web site. Our **ascii.js** is around 61 lines long (35 "substantive").

Implement your page using **XHTML** 1.1 as taught in class. Your page must pass the W3C XHTML 1.1 **validator**. Choose appropriate tags to match the structure of the page content. Do not express style information in the XHTML page itself, such as inline styles or presentational XHTML tags such as `b` or `font`.

Express all stylistic information on the page in **CSS** using your style sheet file. For full credit, your style sheet must successfully pass the W3C CSS **validator**. You should not use HTML or CSS constructs that have not been discussed in lecture, slides, or textbook chapters during the first six weeks of the course.

Your HTML file should contain no JavaScript code whatsoever, a process called "**Unobtrusive JavaScript**." *(See textbook Chapter 8.1 or lecture slides.)* In other words, no `onclick` or other event handlers should be embedded in the HTML. Instead, attach all event handlers using `window.onload` and the Document Object Model (DOM).

Format your HTML, CSS, and JS to be readable, like to the examples in class. Place a comment header in each HTML/CSS/JS file. Your JavaScript should have more **comments**, on each function and complex sections of code. Wrap comments to no more than 100 characters per line.

For full credit, your JavaScript code should pass the provided **JSLint** tool with no errors reported. You should follow reasonable style guidelines similar to those of a CSE 14x programming assignment. In particular, minimize global variables, avoid redundant code, and use parameters and return values properly.

Format your code similarly to the examples from class. Properly use whitespace and indentation. Do not place more than one block element on a line or begin a block element past the 100th character.

Do not place a solution to this assignment on a public web site. Upload your files to the **Webster** server at:

- **https://webster.cs.washington.edu/*your_uwnetid*/hw5/ascii.html**