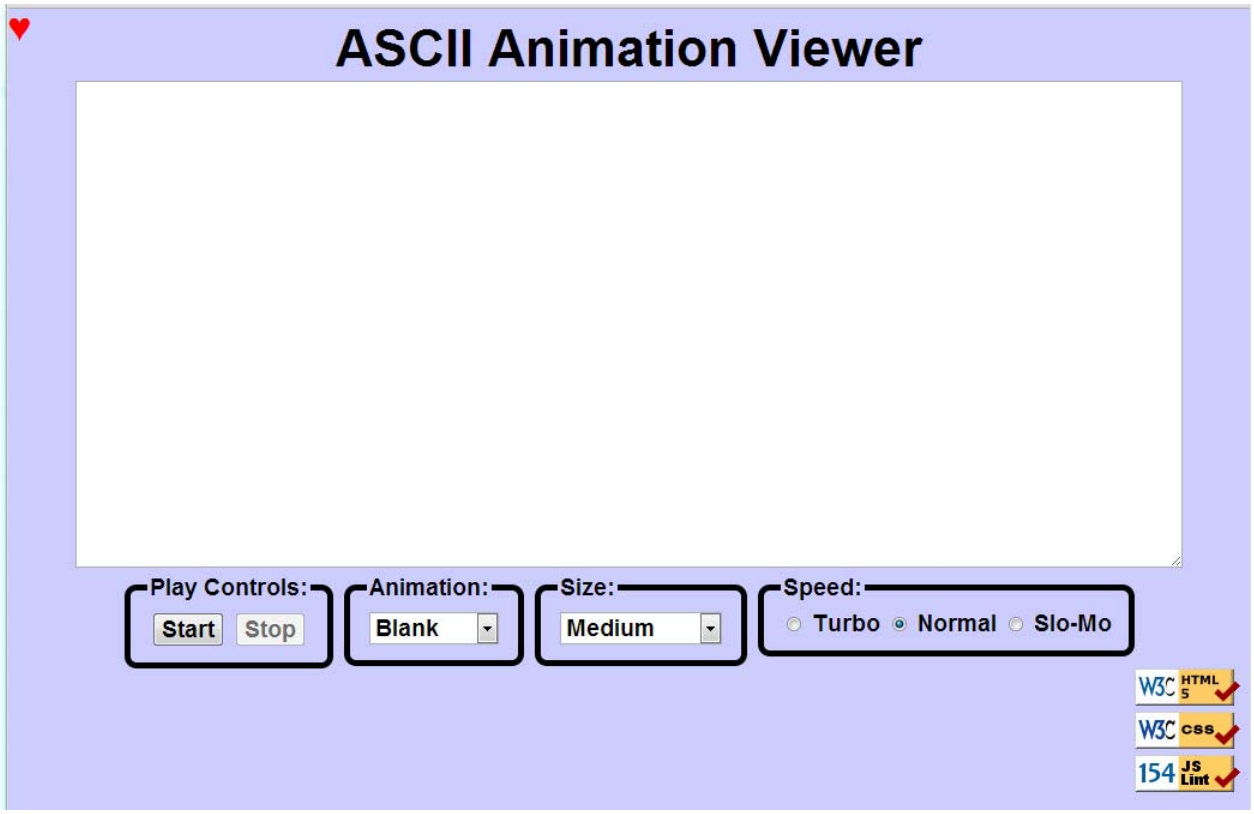
CSSE 290 Web Programming
Homework Assignment 5: ASCIImation

This assignment should increase 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 involves pictures made from 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.

Your first task is to create a page **ascii.html** with a user interface (UI) for creating/viewing ASCIImations. Your page should link to a style sheet you'll write named **ascii.css**. After creating your page, you must make the UI interactive by writing JavaScript code in **ascii.js** so that manipulating the UI controls causes appropriate behavior. Your HTML page should link to your JS file with a `script` tag.

You should also create an ASCIImation of your own, stored in a file named **myanimation.txt**. Your ASCIImation must show non-trivial effort, must have multiple frames of animation, and must be entirely your own work. Be creative!

Summary of the files you need to create:

- **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 custom ASCII animation as a plain text file
- **myanimation.js**, your custom ASCII animation as JavaScript code (so it can be used on the page)

Appearance Details:

The page should have a title of **ASCIImation**. Your html page must link to the following JavaScript and CSS resources.

- **animations.js** *(provided)*
- **myanimation.js** *(you will write this file)*
- **ascii.js** *(you will write this file)*
- **ascii.css** *(you will write this file)*

Ignore the red heart in the sample picture.

The overall page has a background color of #CCCCFF. The preferred font for all text on the page is the default sans-serif font available on the system, in size 14pt, in bold.

The top of the page contains a heading in 32pt bold text, centered horizontally within the page. There is no margin between the heading content area and other neighboring content on the page.

Under the page's heading is a text box with 80 columns and 20 rows, centered horizontally. Its width is 90% of the page width, and height is 400px. It uses a 12pt bold monospace font initially. CSS `width`/`height` properties will set the text box's size.

Below the text box is a set of controls grouped into several field sets, each with a 5px black border around it and a label on top. Their behavior is described below. To get the field sets to appear in a row horizontally, see textbook Chapter 4's section about Element Visibility and the `display` property. You should make sure that the tops of the field sets line up by setting their vertical alignment. The text area and control field sets are centered horizontally.

Below the controls is a right-aligned section with images that are links to the W3C validators and the JSLint tool.

Behavior Details:

The following are the groups of controls at the bottom of the page and each control's behavior. (**NOTE:** Although we put controls in a form in past assignments, **do not use a `form` tag** on your page this time.)

Play Controls:

Start: When clicked, animation begins. 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.

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. During animation, one frame is visible at any moment, starting with the first frame. By default, the animation changes frames once every **250ms**. When the animation reaches the last frame, it loops back around and repeats indefinitely.

(You must implement your animation using a JavaScript timer with the `setInterval` function.)

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.

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.

Your **ascii.html** page should link to the provided **animations.js** file that declares the ASCIImations as global string variables named `EXERCISE`, `JUGGLER`, `BIKE`, and `DIVE`. You shouldn't edit this file; your **ascii.js** file can refer to these variables. For example, if you have a `textarea` on your page with an `id` of `mytextarea`, the following code is legal:

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

The provided **animations.js** file also defines a global associative array named `ANIMATIONS` that maps from indexes (keys) that are strings equal to the names of the animations, such as `"Bike"` or `"Exercise"`, to values that are long

strings representing the entire animation text for that image. Using this array well can help you avoid redundancy.

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 field after choosing a pre-set animation. The animation shown when Play is pressed should reflect these changes. (i.e., Don't capture the text to animate until the user presses the Start button.)

You may assume that the user will not try to type into the text area while animation is in progress. You may also assume that the user will not use the selection box to change to a new animation while animation is occurring; assume that the user will stop any existing animation before changing to a new one.

Custom Animation:

The Custom choice in the Animation box should show an animation that you have created. The http://www.rose-hulman.edu/class/csse/csse290-WebProgramming/201330/SupportCode/StringMaker/stringmaker.html link on the web site can convert your animation to a string that you can put into **myanimation.js**. Don't put comments or headings in **myanimation.txt**; those should NOT be encoded by StringMaker. The text file contains your animation in **plain text**, so that if someone did Select All, Copy, and Paste into your running ASCIImation page, it would animate properly.

Note: You are turning in your custom animation in two ways: first as a plain **.txt** file, and also in a **.js** file as an encoded string.

Font Size:

A drop-down list of font sizes. When one of the font sizes is chosen, it immediately sets the font size in the main text area. The font sizes listed in the drop-down list, and the corresponding font size to set, are:

- Tiny (7pt), Small (10pt), Medium (12pt), Large (16pt), Extra Large (24pt), XXL (32pt)

Initially Medium is selected and the text is 12pt in size. If the animation is playing and one of these buttons is clicked, the font size changes immediately.

Note that when you write the code for changing the font sizes, it is easy to introduce redundancy. By setting a `value` attribute on each of the options in the drop-down list, you can avoid a long series of `if/else` statements.

Speed:

There are three radio buttons labeled "Turbo", "Normal", and "Slo-Mo". Turbo sets the speed of animation to use a **50ms** delay instead of 250ms. Slo-mo uses a 1000ms delay. Initially Normal is selected and the delay is 250ms.

If the animation is already playing and the user selects different speed button, the speed change should take effect immediately (the user shouldn't have to stop and restart the animation to see the change). Changing the speed should not cause the animation to start if it wasn't already started. It also shouldn't reset which frame is showing; it should just change the delay.

Control Enabling/Disabling:

Modify your GUI to disable elements that the user shouldn't be able to click at a given time. Initially and whenever animation is not in progress, the Stop button should be disabled. When animation is in progress, Start and the selection of a different animation should be disabled. The Size box and speed buttons should always be enabled.

Enable or disable a control with its `disabled` property. For example, to disable a control with `id` of `customerlist`:

```
document.getElementById("customerlist").disabled = true;
```

Development Strategy and Hints:

1. Write the basic **HTML** content including the proper UI controls. *(Don't use the* `form` *tag.)*
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 options working.
5. Implement a **minimal Start behavior** so that when Start is clicked, a single frame of animation is shown. Clicking Start multiple times would 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, or use the similar tool built into other browsers such as Chrome. Firebug 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 serious JavaScript programming.

The University of Washington **JSLint** tool can help you find common JavaScript bugs. If you encounter tricky bugs. If so, paste your JavaScript code into JSLint to look for possible errors or warnings.

For full credit, your .js file must be written in **JavaScript "strict" mode** by putting this exact line of code at the top:

```
"use strict";
```

Submission:

Submit your entire `HW5` folder to your `webProgramming` folder on the `wwwusers.csse.rose-hulman.edu` server.