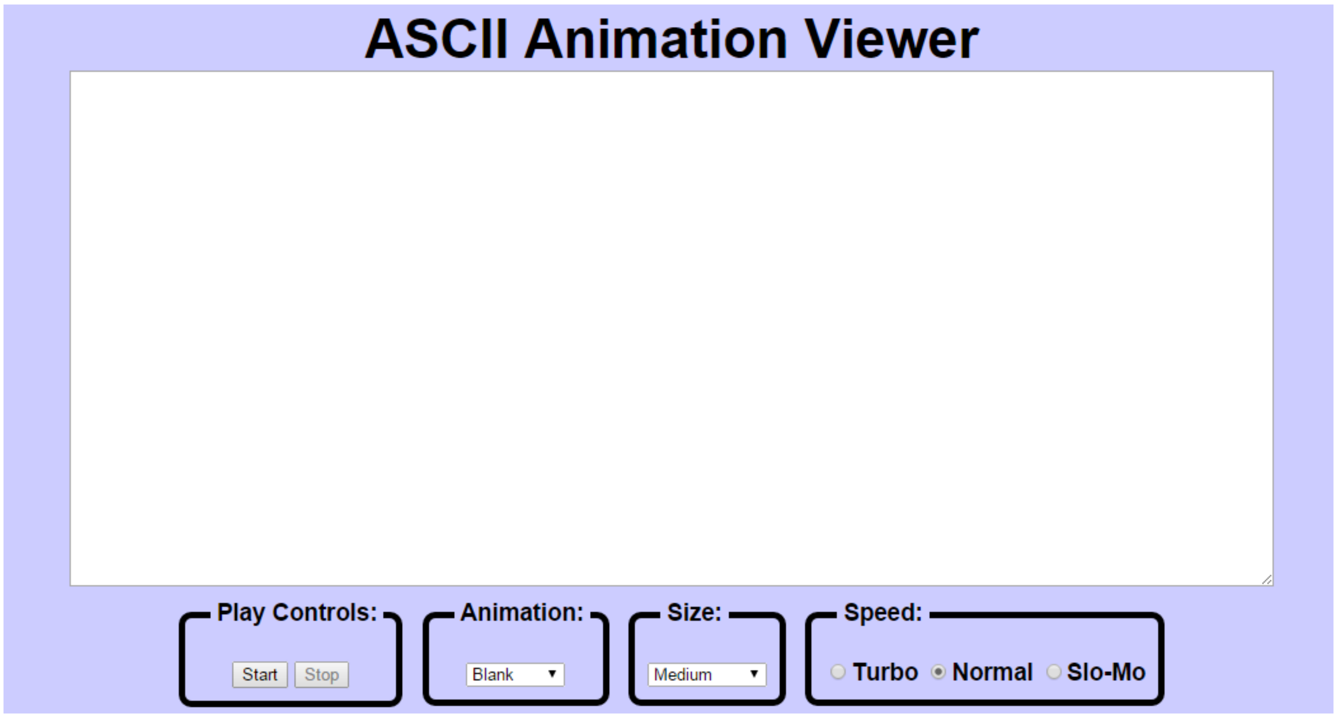
CSSE 290 Web Programming
Homework Assignment 4: ASCIImation

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 animation involves rapidly displaying a sequence of 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 movie buffs are working to recreate in their entirety Star Wars and The Matrix as ASCIImation. Example: http://www.asciimation.co.nz/

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)*

The overall page has a background color of #CCCCFF.

Under the page's heading is a text box with centered horizontally. Its width is 90% of the page width, and height is 400px (set using CSS properties). It uses a 12pt bold monospace font initially. You can expect an animation frame to contain no more than 80 columns and 20 rows.

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 the section of Chapter 4 that is about Element Visibility and the `display` property. You should make the tops of the field sets line up by setting their vertical alignment. The text area and the collection of control field sets are centered horizontally.

Control Details and corresponding behaviors:

(**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, the animation begins. Before the animation begins, all frames of the animation are visible (may require scrolling to see them all). Frames are separated a line that consists of 5 equals signs and a line break (`\n`) character.

When the Start button is clicked, JavaScript code breaks apart whatever text is currently in the text box to produce frames of animation. This could be one of the pre-set animations from a file, or it could be text that the user has typed manually or pasted into the text box. During animation, only one frame of the animation is displayed at any moment. When the animation reaches the last frame, it loops back to the first frame and repeats the animation indefinitely. By default, the delay between frame changes is 250ms.

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

Stop: When clicked, halts the 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 animation names. 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:

```
document.getElementById("mytextarea").value = JUGGLER;
```

The provided **animations.js** file also defines a global associative array named `ANIMATIONS` that maps from the names of the animations (e.g., `"Bike"` or `"Exercise"`) to the long strings that represent the entire animation text for that image. Good use of this array can help you avoid redundancy.

Here is a short example that uses the `ANIMATIONS` array:

```
var whichOne = "Juggler";
document.getElementById ("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 that will be animated until the user presses the Start button.)

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

Custom Animation:

The Custom choice in the Animation box should display an animation that you have created. The http://www.rose-hulman.edu/class/csse/csse290-WebProgramming/201520/SupportCode/StringMaker/stringmaker.html page 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: So that both of the above can be tested, you should turn 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 (even if an animation is running) sets the font size in the main text area. The font size names that should be in the drop-down list, and the corresponding font sizes are:

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

Initially Medium is selected and the text is 12pt.

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 a 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 length of the delay between frames.

Enabling/Disabling various Controls:

Include JavaScript code that modifies your GUI to disable any elements that the user shouldn't be able to click at a given time. Initially and whenever animation is not running, 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.

To enable or disable a control, use 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.

Submission:

Submit your entire `HW4` folder to your `webProgramming` folder on the `wwwuser.csse.rose-hulman.edu` server. This time I did not pre-create the folder.