# Journal Entry for Unit 5

# Weblinks

Site location: <http://student.athabascau.ca/~trevorgi4/Unit5/>

Animation 1: <http://student.athabascau.ca/~trevorgi4/Unit5/projectControls.htm>

Animation 2: <http://student.athabascau.ca/~trevorgi4/Unit5/processProcedures.htm>

Animation 3: <http://student.athabascau.ca/~trevorgi4/Unit5/softwareDevelopment.htm> (and main page).

# Introduction

To better understand JavaScript I apoted to do some Animations, although these are mainly for ‘flair’ they all me to have a deeper understanding of how JavaScript works.

To do this work, I continued to do the Lynda.com ‘Web Developer’ learning path.

<https://www.lynda.com/learning-paths/Web/become-a-web-developer>

Specifically the JavaScript Essential Training by Morten Rand-Herdiksen

<https://www.lynda.com/JavaScript-tutorials/JavaScript-Essential-Training/574716-2.html>

This allowed me to explore the many features and functions that JavaScript has, and how it ties into the overall development of a website.

For my task specifically, which was to create animations (both canvas and SVG), I also researched the following:

* Learning the HTML Canvas
* <https://www.lynda.com/JavaScript-tutorials/Learning-HTML-Canvas/765406-2.html>
* Learning SVG
* <https://www.lynda.com/SVG-tutorials/Learning-SVG/622089-2.html>
* MDN web docs Canvas Tutorial
* <https://developer.mozilla.org/en-US/docs/Web/API/Canvas_API/Tutorial>
* As well as other supporting websites, like stackoverflow, javascript.info, and more.

I began by creating simple SVGs to work from. To do this is originally used InkScape to create the SVGs and then various SVG optimization tools to get simplified versions of the animations for the final. There are many optimization tools out there and some are described in the section of this document that is related.

In the end however I ended up just creating the very specific coordinates for my animations on graph paper and converting these into SVGs.

There are 2 SVGs that I use in my final site. Because these are both animated in a similar function, they share a common script. This script is somewhat versatile, meaning it is somewhere between custom for this assignment and a framework. However to be an actual framework, there would need to be a complete re-build. It does however allow for other animations, that are similar, to be built with relative ease.

The second part was to use the HTML 5 Canvas object. For this I made a modified version of my unit 4 submission “MatixRain” where random streams of text go across the screen. Once a stream has passed then the final text of the image appears.

In this script a json object can be taken in and converted to a similar animation. It takes into account the size of the canvas and converts the text into the appropriate size to fit and final colors that should be rendered.

This script is also somewhere between custom for this assignment and a more generally usable script. In future iterations a text editor can be created that converts text entered into the json object that will be rendered. For this assignment however the json object is part of the script.

By creating both of these scripts I was able to get a good understanding of the following JavaScript principles:

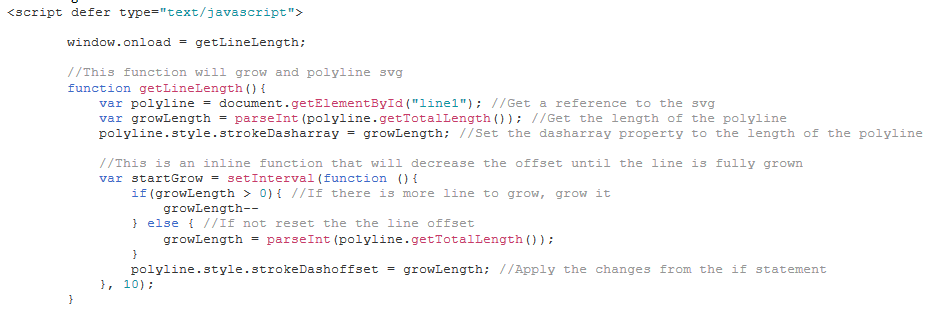
* Classes (for newer JavaScript implementation), callbacks, timeouts, the animation frame, getting properties from an HTML element and using them, general flow control, de-bugging, anonymous/inline functions, maintaining references (including binding), and more.

What follows are the approaches that I took and the challenged that I faced to finish this assignment.

# Starting out – Creating an animation

I began with the SVG animations, first is the “Bar Chart” SVG Animation, found on the “Project Controls” section of my website. Here are the notes that were created during completion of the task.

* Start by creatine a polyline. To animate, get the total length of the line and use the dasharray-offset to show the line 'growing'. This idea came from the Lynda course: <https://www.lynda.com/SVG-tutorials/Learning-SVG/622089-2.html>

The starting code to animate one line: 

Some interesting methods from the SVG:

* getTotalLength() returns the total length of the line,
* getPointAtLength() gets the x,y from a spot on the line
* Items that I have researched to get this get this to work
  + Closures (Lynda and google)
  + Callbacks (MDN, CodingTrain, and others)
  + Promises (and async/await)

Thoughts on the way I am going about this:

* When done just with JavaScript the functions look clunky because they do not tween, etc. After one animation is complete, I will look to try controlling css styles through its built animations. These will include items like tweening. And with the knowledge of how manipulate them, this may work out better.
* I have done the complete tutorial on 'canvas' from MDN and would also like to recreate these animations without SVG. This way will have tried a complete compliment of ways to animate before moving on libraries,
* I have restructured the original code to have the animated objects into a class (with sub-classes). By going about this way I am able to consolidate the timing into a requestAnimationFrame. The challenge now is to have the multiple objects attempt to get the same animation frame.

Questions include:

* should the function that is requesting the frame be in charge of calculations?
* Or should this be up to the requester. How do the right frames get to the right callers?
* If I just return does this stop the recursion on the animation?
* Should I have class variable that holds a list of information that the animator function can access and the class memebers?
* If no one is requesting an amination how can I start/stop the recursion on the animation function? Can/Should callbacks or promises be used?

I am having trouble getting the "Animation" to handle all of the amination functions. This is mainly because the requestAnimationFrame is scoped to the 'window' object, so I lose currently scoped referenced. After several hours of tying I have now turned to putting the animation separate from the class. To do this I am having a function in global scope responsible for the rAF (requestAnimationFrame).

Once the frame callback is executed the function will iterate over a list of objects that need the frame information and calls their animation function (i.e. - they will animate themselves). This is done with the use of 'static' list (which is not supported well in js and I am starting with a global variable) and a bool on the object (whether it wants to stay on the list). If the list is empty, the request frame function will stop.

Having restructured the class and the animation callback the whole script is functioning well. I do realize that what I am trying to do has been done very well by other already (some notable examples include:

Google charts <https://developers.google.com/chart/>

Vivsus <https://maxwellito.github.io/vivus/>

javascript.info (where I got some of the math from) <https://javascript.info/>

This code is not the best but creating it did take me through the gambit of javascript, such as: normal conventions in js, like: anonymous functions, storing functions in a variable, etc. It also showed me how it differs from other languages I have used, like: no enums, little class support, many public variables, and a looser structure to the code.

I am learning to embrace this less structure, although I am still writing more lines than needed (by far), this is way for me to better understand how the code is functioning.

To complete this first module if learned (and in many cases incorporated) the following:

* anonymous functions, functions as parameters, functions as variables.
* setInveral and timeouts, as well as callbacks and promises
* Implemented classes and sub-classes
* Kept references to 'this' through various levels of scope with various means, including: bind, 'window[functionName]call (to replace text), sending objects through parameters, and having objects placed on a list for later callback
* Used various mean of de-bugging, including: the chrome developer editor, console logging, adding members to objects (like request for timeout flag), and created printable logs (and using console.table), some other good ideas here: <https://raygun.com/javascript-debugging-tips>

Some items that have not yet tackled that I will focus on in upcoming work, are:

* Events, listeners and triggers
* Async and promises
* API requests (to be combined with promises)
* The canvas
* Manipulation of DOM elements and CSS

Overall I am happy with the way this animation class module turned out. Going forward I will lean more on other developer's code to complete my own. I feel like I have a good idea on how to leverages other's work and know how to put it to good use.

The next step is to extend the framework to make another animation that includes animating over a spline. This will use the line animation and return a 'point at' where I will set the transform of another object.

# Step 2 – Extending the script for another animation

First, what am I trying to do:

* Take the box images and animate them over a spline.
* To do this the line object will need to provide a 'pointat' reference and the transform of the box will set to this element.
* This should also include 'stops' in the animation where elements what for the animation 'step'.
* To do this my code can be extended to include a 'pointat' function line the line and transitions
* pointat should be straight forward, this is already part of a line animation that the object can return
* The delay will need to be worked into the overall design. This will need to call 'timeouts' and starts from one original call

Discussion:

I have created a new SVG file in InkScape and cleaned it (reduced the SVG XML markup for production use). To do this I tried various optimizers online. Even after optimization though there was still cleaning to do. This was mainly around styles, path orders/groupings, and providing class/ids to each path.

* InkScape is a really nice tool for creating and editing SVGs, and its free. It does not have all the features of a professional development tool (like illustrator) but can produce the professional results.
* I like that files to SVG format and the XML is editable in the application
* The file can be made with width and height, and the view box can be setup in the file as well.
* For my purposes this is a really great tool to quickly make good looking graphics

The tools I tried for optimization are here:

* <https://www.ofoct.com/image-converter/svg-optimizer.html>
* Lots of options, but no preview. Only provides a file download, so this is a bit of a hassle to see what the changes do
* <https://jakearchibald.github.io/svgomg/>
* Really nice interface and lots of options. Has a live file preview where you can see how the changes of the markup will effect the image
* <http://www.petercollingridge.appspot.com/svg-optimiser>
* Easy to use, simple options, and provides a preview of the file. One thing I really like about this editor is that will automatically group common styles to classes and place them with the class name in a <style> tag.
* This editor did drop 3 paths from the image and no matter what options I selected, I could not get them back
* <https://www.svgminify.com/>
* Decent tool, but did not end up going with this for my transition

In the end I used ofoct.com, but do think I will try to use petercollinridge.appspot.com first in the future, the class groping is a really smart idea.

First step is to modify the code to include a 'point-at' variable that tell me the location on the screen where the dasharrayoffset is. This will allow me to attach another object to that location. In other words:

* I will convert my line graph code to be 'spline' animator.
* I also want to attach particle effects (some neat ones from codepen.io)
* To get the current point there is a function 'getpointAtLength' and this can take a percentage, which I am already calculating for the curves. Make this a class variable and use it.

Next, we are using different element types, so far there was only polylines. I could change all types to be polylines and continue, but I do want to limit the animation possibilities for this fact.

* To do this the bar animation needs to be changed look for ‘rect’ rather than ‘polyline’.
* Also the line animation needs to accept paths (which do not have points), so no more start/stop transition.
* Once this code is altered then the need to find the center of elements that are being animated along the spline and the best way to connect them

Next is how to get updates to the object is going to follow the spline animation. Right now, the spline is (or any animated object) is being updated through a call back. But what is needed here, is for the follow object to get the updated points at each frame and pass this on. This is needed so we do not have multiple objects requesting a frame (that do not need one). Meaning the ‘spline’ will take care of the progress measurement and should pass this on to its children.

* There are two possibilities to do this (really there are many, but I will focus on two)
* One is to have the follow object exist outside of the class and use promises to get the updated position of the spline.
* The second is to make another sub-class and have this object become a member of the AnimatedObject classs.
* I am choosing to do the second scenerio. This is because there is no additional logic needed to perform the work, much can be done in object constructions.
* First to setup the 'spline' sub-class an additional parameter will be taken, this is the spline object. The box gets 'constructed' as per normal and then once created in the sub-class it calls for the construction of the spline. (This becomes a member of this class).
* After this is complete it then calls for the animation function to be performed on the spline object. Because the amination function returns a percentage complete, then this can be used to determine where the box should placed along the spline.
* Although this is complex and maybe even confusing, it will work well for the overall design of the animated objects.
* The follow object has the spline as a member, so whenever the spline is updated (every frame) then the follow object can check its new '.pointAt' position, and update itself

Finally, we need to take in an array of follow objects and aminate them over a single spline.

* To do this we need to create an array of follow objects and update their position each frame.
* This is done by incorporating a 'groupFollow' function in the code. This functions in a similar way to the 'requestFrame' list. Each object is updated to the shared spline position on each call
* Then, we need to stop the objects when they get to their final position. This is done by checking where is started and if it has passed this position on both the x and y then it places itself on its start x and y, then stops requesting updates by removing itself from the group follow list.
* The final code (in its current form is complex and convoluted. If I were to do this again, I would take a different approach for the same results. However, I do feel that this has given me a really good grasp on the javascript language.
* As a note to do this I can use the <animateMotion> tag that is part of the svg specification, however I want to stop certain boxes at points along the line, and I do not think there is a way to do this.

Notes of follow objects:

These are general notes on how the follow class should function (focusing on the group follow structure)

1. Create lead follow object
2. Create spline to be followed
3. Create follower objects (array)
4. Pass in the follower objects and the spline to the lead object
5. Set off animation for spline
   1. Note: spline should always have a pointAt position (part of construction)
6. Check the spline for timeout start/stop (even though this will have its own).
   1. Note: will this need a delay then? Could request frames and do nothing with them, but this is waste.
   2. If it does its timeout then alignment may be an issue (I.e. - could be a frame behind)
   3. Test this.
7. Set off animation on lead follow object
   1. Note: this is only to be in the callback loop with the spline, all information about the animation should come from the spline
8. Update cycle
   1. For each following object
      1. Set position to pointAt (either call as part of the parent or attach pointAt to this).
      2. Check to see if current position is farther along the path then when the follow object started
      3. If so, then remove from the list
9. Once list is empty, stop animation (is this needed if everything is based on the spline).

The above covers my thought process while getting the ‘spline’ animation to work with the original class. I did my best to explain this, but obviously one needs to read and understand the code with this explanation to see the challenges that were worked through in getting to a final working code.

# Moving On – Working with the canvas

This will be the final animation that I put together for this unit.

In this animation I am taking a step back from SVGs and the animation class I created and will focus on using the canvas element.

This animation will follow the same idea that I did in Unit 4 with some modifications:

* I want the streams to go horizontal and not vertical
* When a specific part of the canvas is reached, I want the letters to stay rendered

To do this I want to place and html element with the all the text (and white space) needed to render the final canvas.

* The streams will travel across each line and when if finds non-white space it renders the char at that location.
* This will allow for any text to be rendered this way.
* The starting array will be the total lines by the total line.length. (all lengths should be equal)
* I am using the same idea as submitted in Unit 4 but will be creating the code from the ground up (now that I have a better understanding on how this can be put together)
* The best way to do this is with a json object where the format is understood by the code and needed references are created.

The basic flow of the code is as follows:

* Take the json object in and convert it into items for the animation
* This includes finding the optimum font size, the color of the final render, and an array of random chars to use for the streaming (created from the submitted text).
* After setup is complete, streams of random text are created and flow across the screen horizontally.
* After a stream passes a char to be rendered then a random char ‘remains’ behind.
* After all the streams have passed the canvas then the final text is revealed, through a fade-in process.

One item I would have liked to incorporate into this is to have one char fade in at a time, rather than all at once, but this can be implemented in future versions of this code.

The final result took a while, with many challenges, but I am happy with the way it turned out.

# Making It Production Ready

So far, the scripts have been created in line with their own HTML document, but for the final submission they needed to be able to be used as separately linked JavaScript files in my website. The following are notes from the code reviewed that I performed.

SVG

* To clean this code up removed a lot of the printing and testing
* I also changed the bars from polylines to rects
* The bar animation is the easiest to understand, with line being the second.
* The spline sub-class uses the line sub-class for part of it and is basically custom built for the what is being done in the animation. I need to fully explain this one so that its understood. (done above and in the comments of the code)
* Looking and the code now it is plain to see that I would redo this in many ways. However, then fact that I realize this also says how much I have learned with HTML, CSS, and vanilla JavaScript, so this is good.
* Most of the code was cleaned-up. During this process it is plain that there are still some bugs to fix and a lot more efficiencies that can be realized.
* However, I am very far down the rabbit hole now and have done much to understand JavaScript language. With this in mind, there are still many things that I need to do in order for this work to work in my website. So, I am going to start splitting up the code and making ready to place into my site. It is not to say that I have given up on this endeavor, but it is to say that I need to get all of the pieces completed (to complete this unit) and after this may take some time to examine the overall approach and see if there are gains to be made there, like other libraries, etc. rather than continue to refine this approach.
* In the end I am very happy with what I have accomplished for this unit.

MatrixRain

* This is setup to be somewhere in between an animation for this unit and library that can take in any text and perform this function.
* To make this an actual library the text would need to be taken and converted to a json object, from here the script can convert it into the optimal size (font) and produce and animation.
* It is this piece of taking and converting text into a compatible json that would need to be added. This can include different colors, number of lines, and chars per line. Right now, the script can handle most of these as variables.
* To create this script, I made a lot of global variables, this was good to get the functionality working, but seems to have performance impacts. The cleaning involved making these variables objects specific.
* To get an understanding of how this needs to be done the main functions of the script need to be looked at to see what objects should be responsible for the various objects.
* There is:
  + The random char that streams across
  + The reveal char that stays in position while the other chars stream
  + The streams of chars themselves
  + The canvas 'context' that needs to know what to draw and how
* The main question is how should these be organized to remove global variables?
  + Should the canvas context take on the role of being feed (or looking for) all the items that want to be drawn? In this way the context would be a function and the items passed into it.
  + Or should the canvas context be passed around to all the things that need to be drawn so they can access the context when they are needed?
  + Should the json blob be made the main variable and have all the 'options' added to it, like stream size, speed, how quickly the chars change, their fade, etc.?
  + Should the mainstream function contain the options and access the context and/or the json blob as needed?
  + Should any of these (the context, the json blob, or the streams) be the one global object, or should all of these be global with the options contained in one or more of them?
* I will not be able to answer all these questions with great proficiency at this point, but I can take some educated guesses and pick a path forward.
* For this exercise I have made the three main objects global, the json object, the canvas context, and the streams. Because of this, there is ‘crossover’ and this can be cleaned up more.
* It should be noted that the basic functionality is complete for this script and this is something make it 'production ready' more than it is adding new features.
* I have taken some memory profiles to see what changes are made. Overall the performance is not bad. The JS heap gets to about 25mb before garbage collected on load.
* There was a lot of cleanup done and code works as hoped. However, there is still a lot of polish that can be added to this.
* Some future ideas/changes would be:
  + Have the individual letters fade and reappear in the final form (with their respective colors) rather than all of the letters fading at once
  + Add some color highlighting to the intermediate letters (while waiting for the final reveal) because without the blur they are bland.
  + General changes to speeds, blurs, etc.
  + Changing the overall design of the code to be more 'concise' and focused, right now there is all the functionality but some of it scattered across different functions with some global variables that may not be needed.
* Things that I am really happy with in this code:
  + The ability to take in larger blocks from a json object and break it down as needed to perform the final animation
  + Using the letter provided to make the pool of random letters needed.
  + The blur function to help with the visual of the letters going by
  + How each symbol is contained and the render functions are 'recursive' to the symbol level (the Streams call the Stream render, which in turns calls the Symbol render) it is at this level where context switching, symbol switching, blur, fade, etc. happen. Having this function hold all of this allowed for easy growth and changes to the rest of the code.

# Outro – Final Thoughts

In closing, I have created 3 aminations spanning 2 scripts. These scripts are flexible enough to take on different (specific) animations types. Through this process I have learned a lot about how JavaScript is created and implemented in web development.

Some items that I did not cover in great extent are accessing of the DOM and manipulating it. This was a conscious decision. The main reason I did not focus on this for unit is because I realized that there are a lot of frameworks and libraries dedicated to just this, and this is part of upcoming units.

With that said I took this opportunity to dive into vanilla JavaScript and work within it’s confines.

The final code also uses newer concepts of JavaScript, like classes, these would need to be converted to support older browsers as well. This is something that can be looked at in future units as well. One such library is “Babel” <https://babeljs.io/>.

# Mapping – Back to the personas

Mapping this back to my persona’s is more difficult. My site is one that is more content based, meaning I am trying to deliver a more information focused site and animations do not really fit this. But having said there is not much JavaScript related that is a direct connection to these personas.

However, some of what I am trying to highlight on my site is software development, and in this context having custom made scripts can help highlight what my abilities are. In addition part of being a professional is polish and having these aminations is a type of polish to compliment my skills.

I will continue this type of polish in future units, by some redesign of the layout and using CSS/JS to give the overall look of the site a cleaner look and feel.