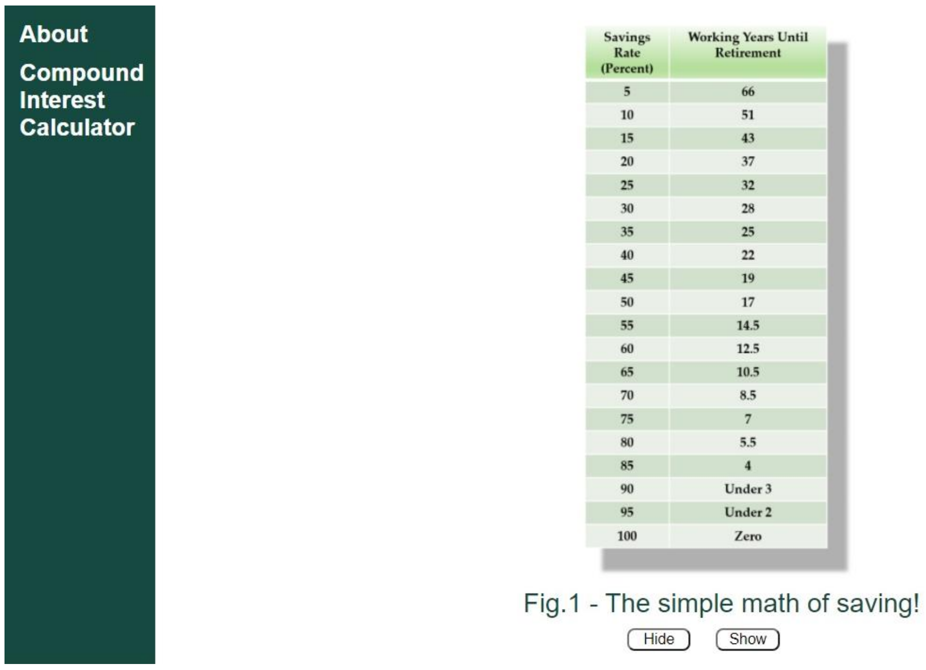
HTML5 – CSS3 – JavaScript - jQuery Project

Part 1: Project description

Website topic

The topic of this website is a financial calculator focusing on a subject known as FIRE (Financial independence/Retire early). I choose this topic as FIRE is a popular subject among software engineers, and I wished to learn more about developing charting applications. The key principle of FIRE is that by living frugally and investing a high proportion of one's income into assets that generate money, it is possible to become financially independent or retire at a much earlier age than the traditional retirement age of 65+.

The site contains an "About" section where the concept of FIRE is outlined. In addition, there is a chart which I developed using JavaScript, in order to calculate at a given expense, savings, and investment return rate how long it will take the user to become financially independent (i.e. how long it will take before their passive earnings from investments is equal to their expenses; see figure 2.).



Savings Rate (Percent)	Working Years Until Retirement
5	66
10	51
15	43
20	37
25	32
30	28
35	25
40	22
45	19
50	17
55	14.5
60	12.5
65	10.5
70	8.5
75	7
80	5.5
85	4
90	Under 3
95	Under 2
100	Zero

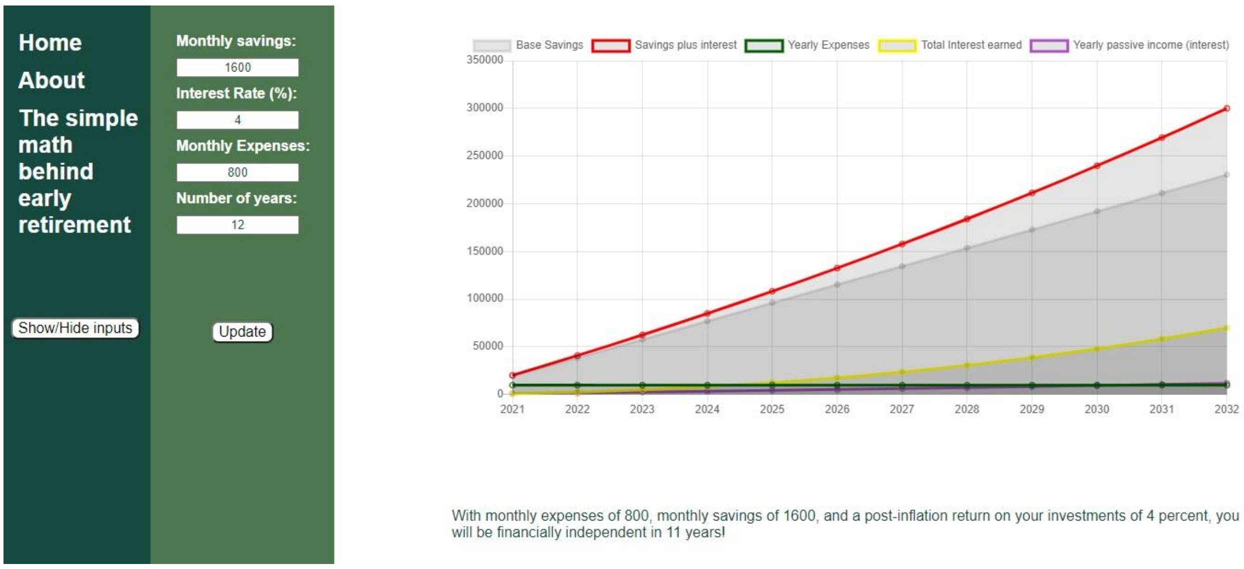
Fig.1 - The simple math of saving!

Fig.1. FIRE website page example

Technologies used

In addition to HTML and CSS to form the core pages of the website, I used JavaScript to develop the financial-independence charting application. For this, I researched how to use an open-source JavaScript charting framework known as Chart.js. This allowed me to develop charts that would project the results of the JavaScript methods I developed to calculate the financial data (e.g. the yearly compounding total savings one would have at a specific level of savings, expenses, interest rate, and over a specified number of years). The script also updates the HTML with the results of a function that calculates how long, if at all, it would take to achieve financial independence with the given parameters and timespan entered.

In addition, there is a separate JavaScript script on this page with a function (notifyEmptyValues()) which verifies that the data entered by the user is valid - for example, that text is not entered into a form which requires a number. If an invalid entry is entered, a JavaScript alert is returned to the user.



Monthly savings: 1600

Interest Rate (%): 4

Monthly Expenses: 800

Number of years: 12

With monthly expenses of 800, monthly savings of 1600, and a post-inflation return on your investments of 4 percent, you will be financially independent in 11 years!

Fig.2. Charting application

In addition, the following JavaScript and jQuery features have been added to the website:

- About page: jQuery slideUp and slideDown functions have been added to the “show” and “hide” buttons on this page.
- Compound interest calculator: In addition to the chart and the input-verification functionality (both created with JavaScript) a button has been added that shows/hides the div tag with the input forms for the calculator (“show/hide inputs”; see fig.2.). This has been achieved using jQuery’s fadeToggle method.
- The simple math behind early retirement: A jQuery show/hide functionality has been added to the buttons on this page. These buttons can be used to show or hide the image on this page.

Experience and difficulties encountered

As I had some prior experience with HTML and CSS, creating the skeleton of the website was relatively easy, and I was able to quickly develop the basic HTML and CSS framework of the site. Most of my time, effort (and difficulty!) was with using JavaScript (in particular, Chart.js) to create the financial charting application on the site.

The first difficulty encountered was getting Chart.js to show a line graph. While Chart.js' API is well documented on their website, and contains a "quick start" guide, this primarily focuses on how to create a bar chart, not a line graph like I required. Fortunately, I was able to find a solution to this problem online (using Stack Overflow). Similarly, I initially had issues with updating the data on the chart whenever new information was entered. I eventually uncovered that this issue was caused by a known bug with Chart.js and found a workaround one of the Chart.js users had posted on the library's forum.

A particularly challenging and time-consuming issue was encountered when I added the functionality to set the number of years that the financial calculator should simulate. As I had initially written all the methods to calculate the data over a 10-year period, I had to rewrite or alter every JavaScript calculation method I had created and change them to take in a variable number of years (e.g. what if the user wanted to simulate their finances over a 20-year period instead of 10?). This involved breaking the financial calculator and piece-by-piece putting the functionality back together again. In addition, there were issues with updating the label that represents the number of years, which I solved by again searching the Chart.js forums for answers.

What I would do differently

If I were to start again, I would spend more time at the beginning researching the Chart.js API to see what it can and cannot do. This would have saved me a lot of time researching workarounds when I was trying to get Chart.js to do things it wasn't designed to do. I would also have spent some time becoming aware of known issues and bugs with Chart.js (such as updating labels).

I would also spend more time planning the features I want the website to have and how the methods would be structured and laid out. I would also have spent more time properly documenting my code. For example, because I did not initially consider having a feature where the user could manually enter the number of years to calculate, I did not initially have that functionality in my JavaScript methods. This involved extensive rewriting of my code, and as my code was initially uncommented it was difficult for me to identify what the purpose of each of my methods were. As such, I had to spend additional time figuring out what several of the methods were doing, and I had to add comments to ensure I did not make the same mistake in the future as my codebase grew.