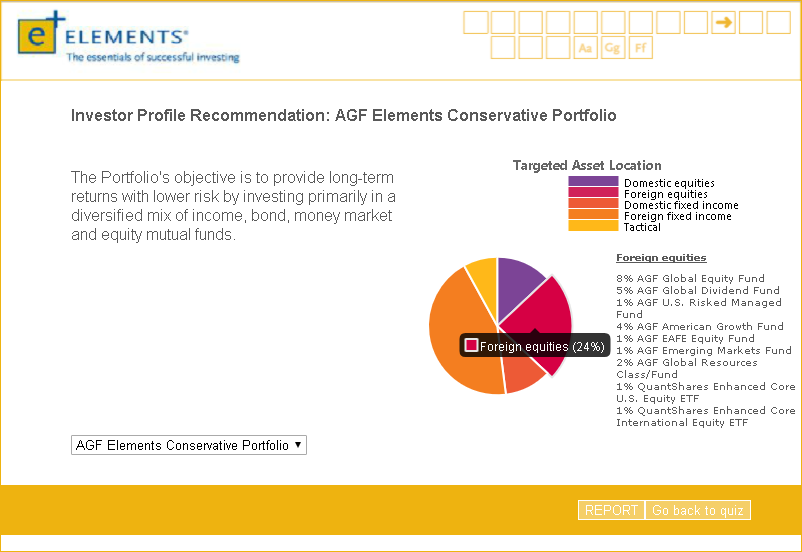
About the Pie Chart

Along with the questionnaire results that are displayed on the last slide, a pie chart is generated to deliver visual information about the targeted asset locations for each type of Elements portfolio.

Pie Chart features:

* Dynamic (number of slices, legend items, chart title, colors)
* On-hover animation (colour of slice darkens)
* Tooltip displays name of legend item + its % on hover
* Slices explode/implode upon being clicked
* Slices display associated funds list upon being clicked

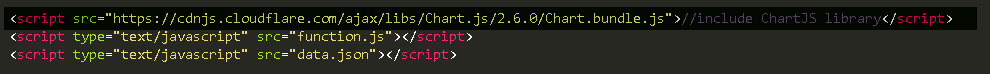


Getting into the Code

The pie chart is created using Chart.js, a JavaScript framework for developers to create different types of graphs. [Official Chart.js Documentation](http://www.chartjs.org/docs/latest/)

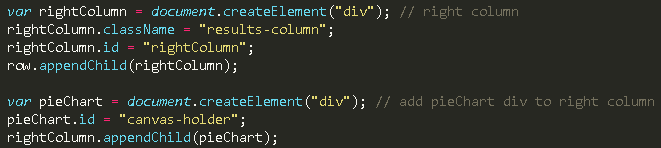
## HTML & JavaScript components

**1. To use Chart.js, the library must be referenced in the HTML as such:**



This line indicates to the browser running the script that it will be using features included in the chart.js library. The version used in this code is v2.6.0.

**2. Create the HTML canvas container (div) *within the JavaScript*:**



Due to the way the quiz layout is set up (a large banner where a section slides into view), the canvas container cannot be declared directly in the HTML. Using the lines of code above, a div with id “canvas-holder” is created. Since it will be in the right column of the results page, the div is appended to parent id “rightColumn”.

**3. Previously created canvas/iframe must be removed because another chart is loaded in its place:**



This code must go before the pie chart creation function is called so that any existing canvas and iframe is removed. If this is not done, any pie chart that is created over an existing one will glitch out and display data from previous charts.

**4. “Create Pie Chart” function:**



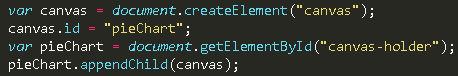
There are many instances in which a pie chart needs to be generated throughout the code; implementing a function, “createPieChart”, allows it to be easily called whenever necessary. The rest of the code regarding the pie chart goes within the two curly brackets to indicate that they make up the function. In order to perform its function, 3 parameters need to be passed in: “slide”, “i”, “portfolio”.

**“slide”** – the blank slide that the pie chart is generated on; portion that slides in/out of view

**“i”** – the numerical index of the slide on which the pie chart is generated

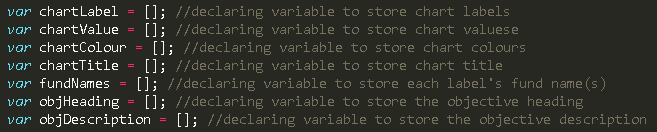
**“portfolio”** – the investment portfolio (e.g. yield, conservative, balanced, growth, global) that the investor has been matched with

**5. Create the canvas HTML node *within the JavaScript*:**

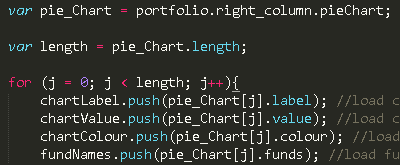


The code above creates a <canvas> node with id “pieChart” within the HTML, which is where the pie chart is rendered. It is also appended to the “canvas-holder” div so that it can be styled on the page.

**6. Declare variables that will be used to store the dynamic data from the JSON files:**



**7. Store the data from the JSON files in the corresponding variables:**



The variable, “pie\_Chart”, stores the array of objects assigned to the object key “pieChart” from the JSON file. The variable, “length”, stores the *number of indexes* this array contains. In context, this number refers to the number of slices the pie chart will have.

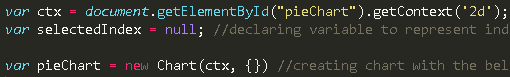
The “.push” function loads the preceding variable with data from within the brackets that follow. Arrays can also be loaded into a variable using “.push” when the index is incrementally looped (instead of replacing existing variable data, it is added onto the end and stored as the value of the next index).

Pseudocode-esque description of the ‘for loop’:

* For every array element in pieChart, push its “label” value into variable “chartLabel” until there chartLabel is an array loaded with one of these values in each increasing index   
  i.e. chartLabel = [ “label1”, “label2”, “label3”, “label4” ]
* For every array element in pieChart, push its “value” value into variable “chartValue” until there chartValue is an array loaded with one of these values in each increasing index   
  i.e. chartValue = [ value1, value2, value3, value4 ]
* Etc.

The idea is that value1 corresponds with “label1”, value2 with “label2”, and so on.

**8. Declare variables to build the pie chart:**

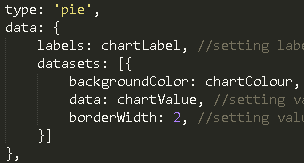


The first line in the above code is a method that returns a drawing context on the canvas. Here, the parameter value “2d” leads to the creation of an object representing a two-dimensional rendering context.

“selectedIndex” is declared as a global (in the scope of the createPieChart function) variable to be used later in slice-specific functions.

The parameters involved in the pie chart creation include the context (ctx) and an object containing chart characteristics (type, data, options).

**9. Specify the pie type and data characteristics:**

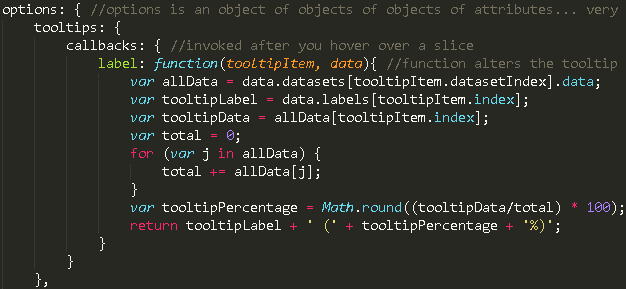


The “type” key dictates what type of chart is created (pie, bar, etc.). The “data” consists of an object of labels and datasets, which are set to the variables that were loaded earlier with arrays of values.

This part of the code is responsible for generating (one example):

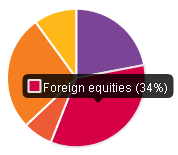


**10. Set tooltips configuration under “options”:**

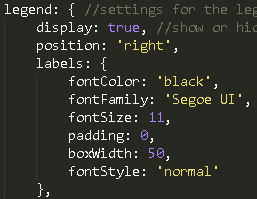


Tooltips appear when pie slices are hovered over. Here, a function has been set so that the tooltips specifically display certain information in the following format:

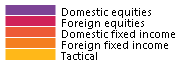
Label (X%) 🡪 The function calculates the percentage of the pie slice and displays it along with the label.



**11. Set legend configuration under “options”:**



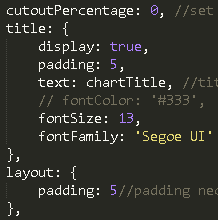
The legend characteristics are set in this section of code; “boxWidth” decides the number of pixels wide that the coloured blocks in the legend will be:





The above line of code is a do-nothing function that is also a legend object. It overrides the Chart JS library’s default legend setting which allows the user to click on a legend item to remove it from the pie chart; the code disables this feature by forcing no events when a legend item is clicked.

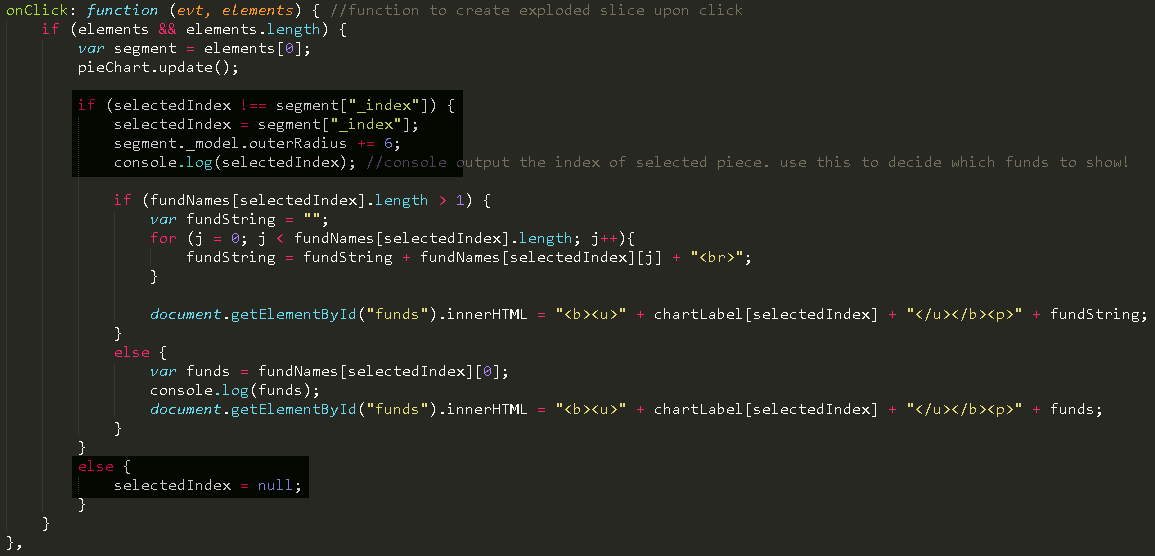
**12. Set basic cut out, title, and layout configurations:**



Cut-out percentage value sets the radius of a circle to be cut out from the center of the chart, creating a donut chart.

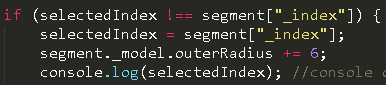
The title stylings are set within title: { }, with the title text being the chartTitle data retrieved from the JSON file.

**13. Create a function that explodes a slice and displays its fund list when it is clicked:**



An onClick: function() { } is created for this purpose. The code within will be split up and explained in the following sections:

**13-a. Create section of function that explodes a slice when it is clicked:**



The variable, ‘segment’, holds the numerical value of each slice that the ChartJS library’s elements[0] array index fed in (from the small section of code above it).

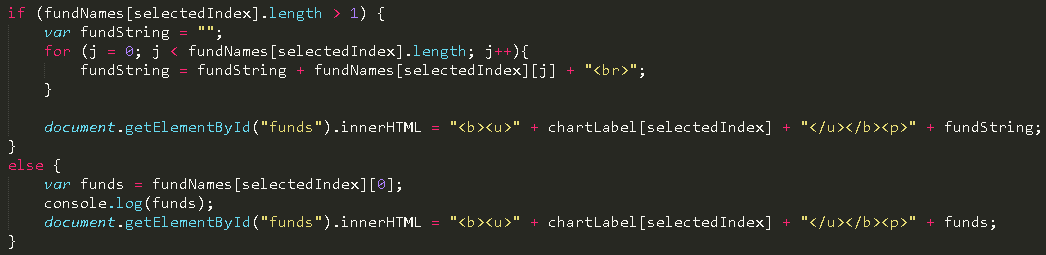
When the slice is clicked, its numerical value is assigned to selectedIndex, and its outer radius property is increased.

\*Only one slice can be exploded at a time. If another one is clicked, the previously exploded one, implodes (well, un-explodes).



If a currently exploded slice is clicked again, this piece of code makes it so that it un-explodes.

**13-b. Create section of function that displays fund list for clicked slice:**



This section is nested within the “if” statement above (not under the else, however). Some fund lists contain many funds, while some only contain one, so the ones that contain 2+ are converted to a string and outputted with line breaks between each fund name.