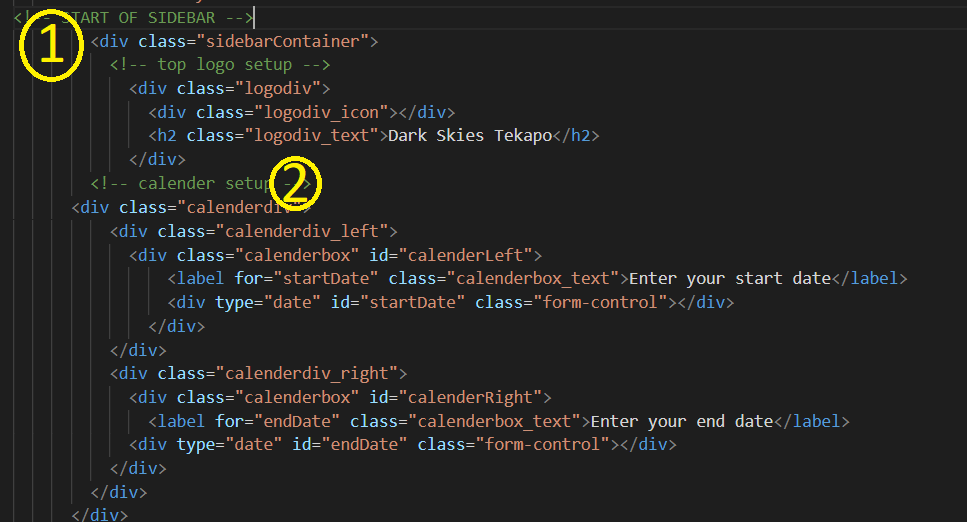
**Code Styling Guide - Dark Skies Booking System**

***By Rory Turnbull***

**HTML**

I have ordered the HTML code into sections. Large sections marked out with uppercase slashed out labels (see 1), are split into smaller sections marked with lower case slashed out labels (see 2). The ends of the large sections also have slashed out labels to mark where that section of data concludes and a new one begins.



The body section is split into three major areas, the first two of which dictate the layout of the website. The body section code starts out with a large section on the sidebar menu for the Dark Skies Tekapo booking system, this is where the user inputs most of their options and the results are displayed. This section is further separated into smaller sections of code that deal with specific parts or functions of the sidebar (such as calendar, map and result display).

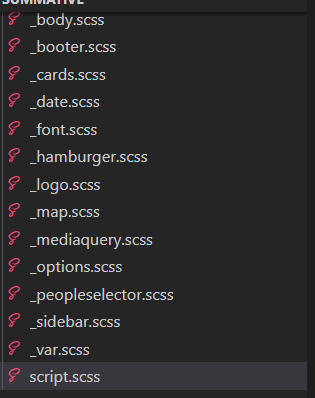
The second section, the main/card screen, dominates most of the viewers screen on the Dark Skies Tekapo web application. This is split further into three main sections, the topmost being where the buttons for filtering results and the search function is, along with a ‘Select’ button to process requests to expand upon options or selected cards. This section is replaced with inputs if the user chooses to filter accommodation by type. The second section is where the cards are displayed when enlarged from their small card format they start in, and is a holding place for the small cards when no accommodation is selected, or if accommodation has been filtered by the ‘Number of People’ filter or search conditions. The final section contains a footer in which links of Tourism NZ and its logos are displayed, to allow the user to visit our corporate sponsors and see that the web application is under the Tourism NZ brand umbrella.

The final uppercase section on the body consists of items that are moved to various places on the screen with the ‘position absolute’ function in CSS. These items are moved in this way as a compromise to get them in the right spot when problems arose during coding.

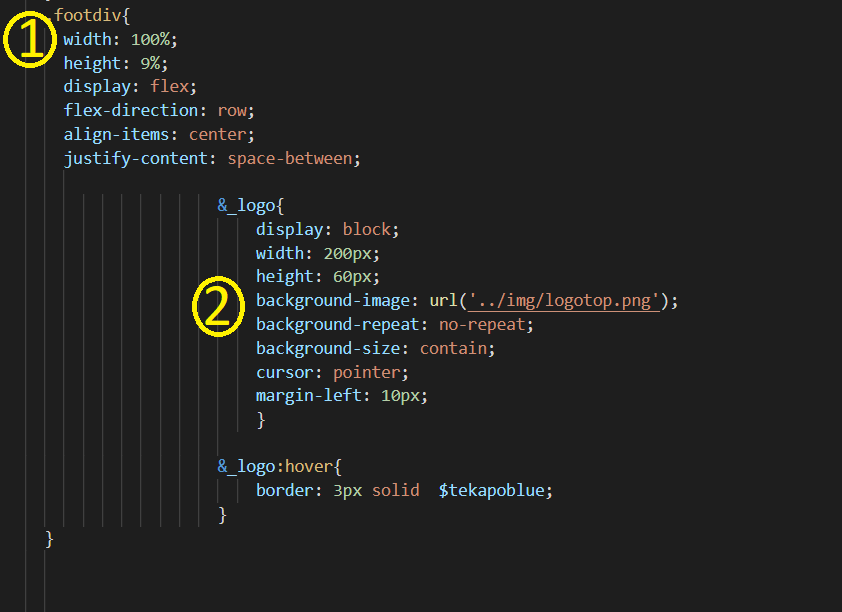
Finally, all the code on the web application has been spaced out with a tab function when applicable, to make the code more readable. In the HTML code, container opening and closing tags come first, with other containers and items within containers tabbed further to the right. This makes the code easier to visually digest, and shows what data is inside what container (div). This can be seen in the above example.

**CSS/SCSS**

In the Dark Skies Tekapo web application project for Tourism NZ I decided to use the Sass accessory to CSS to organise my styling code more efficiently. This necessitated the use of the Koala plugin to manage the SCSS and link it to my CSS styling file. Koal proved to be a very troublesome application and is extremely prone to bugs, so much of the code was originally produced in plain CSS, but set up so a conversion was able to take place when I could iron the bugs out of Koala.



The code style of CSS within this web application revolves around several factors. The first was to make use of SCSS partials, with which I could parcel different areas of the styling code knot different pages for ease of management. The screenshot above shows the many partials I made that linked to my script SCSS, which was my Sass hub that imported all the partials and fed them through Koala to a linked CSS file. The partials are named after what function or part of the web application they manage the styling for.



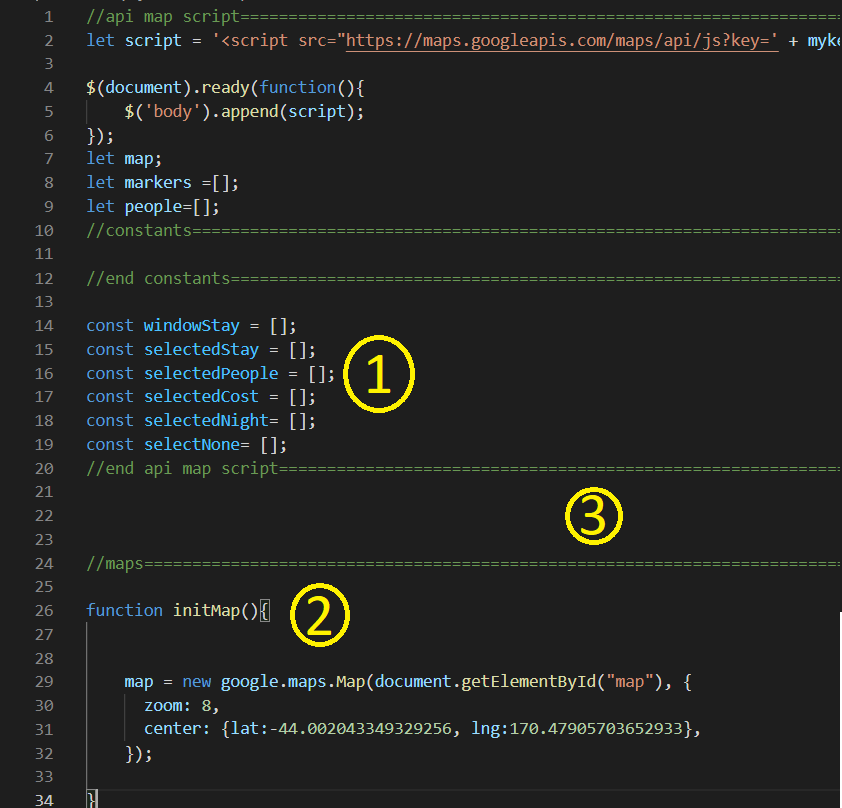
The second factor in organising my SCSS/CSS code was to utilise naming conventions for classes, the keyword system utilised to give containers and items styling. The leading name of a particular container was devised following its position on the page or the function of the data inside it (as in the example above, 1, ‘footdiv’ is a divbox container which holds all the elements for the footer section of the sidebar.) Other containers and items within these master classes were given a suffix name that incorporated the master name with a secondary descriptive title separated out with an underscore (See 2 aove, the class dootdiv\_logo, which styles the logo in the footer, has been named with this convention and indented below the master class in SCSS).

LIke the HTML code before, I have made use of tabulating sections of data underneath the master classes to the right, to make the code more readable. Also, code that is adjusted through media queries is tabulated even further out than all other data to make it visible. Slashed out labels have also been used within the partials to break the data down into manageable sections by type.

**Javascript**

In this project for Tourism NZ there is a large amount of javascript code in the final product. As I was not able to break this down into partial files as with SCSS, I endeavored to clearly label each function, object array, and section of variables clearly.

The first rule I utilised to correctly style the javascript so it was readable and manageable was to make sure to start a new line whenever possible and reasonable, such as when a new variable is being introduced (as in example one), or a new parameter, subfunction, calculation or opening/closing bracket or parenthesis. Some exceptions to this rule are when multiple brackets, parentheses and semicolons follow each other, and it's easier and less wasteful of space to clump them together, or when the data being entered only makes sense if it is on a single line.



The second rule I utilised was to do with ID, object names and function names .I made sure to capitalise the second or subsequent word in these labels to make it clear they were for javascript and so no conflicting errors came up from use of similar terms (they would not function without the capitalisation. See 2 on the example function name, initMap). Variables did not normally follow this rule unless they were global constants.

I split the javascript code data into clearly visible sections by using slashed out starting and ending labels for each individual function, object or set of variables. This is made more visible by creating a line across the screen using = symbol (see the end label above 3 and the start label below 3). The data on the script.js document is ordered specifically, some functions like the global constants need to be at the top of the page so they load first. Data sections are also ordered by type where possible, so the Number of People filters and buttons can be found together for example. Data is also tabulated out where applicable like the CSS and HTML styling.