olympics around the world

March 17, 2015

brigham younng university

PROVO, UTAH

Technical and Design Documentation

Technical Specifications

# Software Details

This website was built for creating an interactive map to show the locations of the modern summer and winter Olympics. The website has been designed using HTML, Javascript and CSS. The Notepad++, Dreamweaver and the GoDaddy web code editor were used as the development environment. The website was developed by Drew Li and Sarva Pulla. It can be accessed at [olympicsaroundtheworld.com](http://olympicsaroundtheworld.com/)

# Compatibility and Limitations

The website is limited to the following web browsers Mozilla Firefox, Internet Explorer and Google Chrome. However, the website works best when used in the Google Chrome web browser. Moreover, it is not as compatible in a mobile device. For any issues or troubleshooting assistance, please contact Drew Li ( [zy.li@byu.edu](mailto:zy.li@byu.edu)) or Sarva Pulla ([sarva@byu.edu](mailto:sarva@byu.edu)).

# License

Olympics Around the World by Drew Li & Sarva Pulla is licensed under a [Creative Commons Attribution­NonCommercial­ShareAlike 3.0 Unported License](http://creativecommons.org/licenses/by-nc-sa/3.0/deed.en_US)

software design

# Purpose of Design

The purpose of this website is to serve as a one stop place for information about the modern Olympics. The locations of the Olympics host venues have been mapped and projected as a single map. Thus decreasing the amount of searching a person has to do to find more about his/her favorite Olympic Host City. The Torch Relay Path for the 2002 Salt Lake City Olympics were included to show the potential of such a unique website.

All the features and functionality have been included in the homepage, so that a user doesn’t have to browse through multiple pages. Thus, making the website user-friendly and easy to navigate. A static header was use for maintaining consistency across the webpages. Also, all the interface elements were put right below the header so that the map appears to be full screen.

# Header and Interface Elements

The header (See Figure 1) is simple and just has three elements Home, About and Contact buttons. This allows the website to be consistent and look professional. Clicking on any of the three options will take you to a different page with similar format as the homepage.

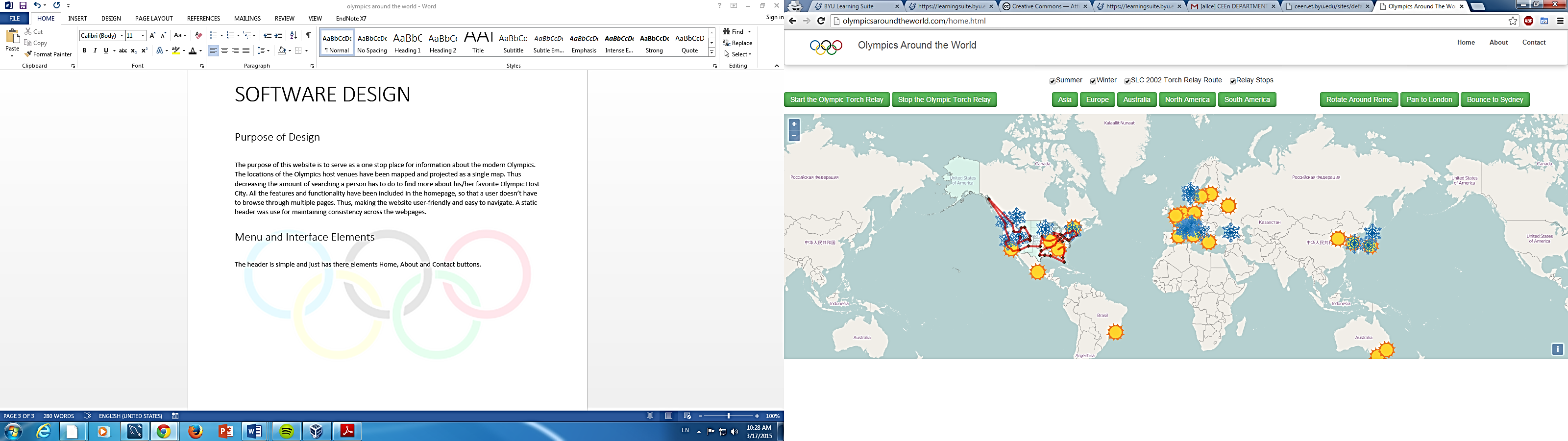


Figure : Header

Figure 2 shows the elements that are located right below the header. The website was added more functionality through these elements. Each element has been assigned a function through JavaScript. Thus, making the website interactive. The functions of each of these elements will be discussed in greater detail later in this document.

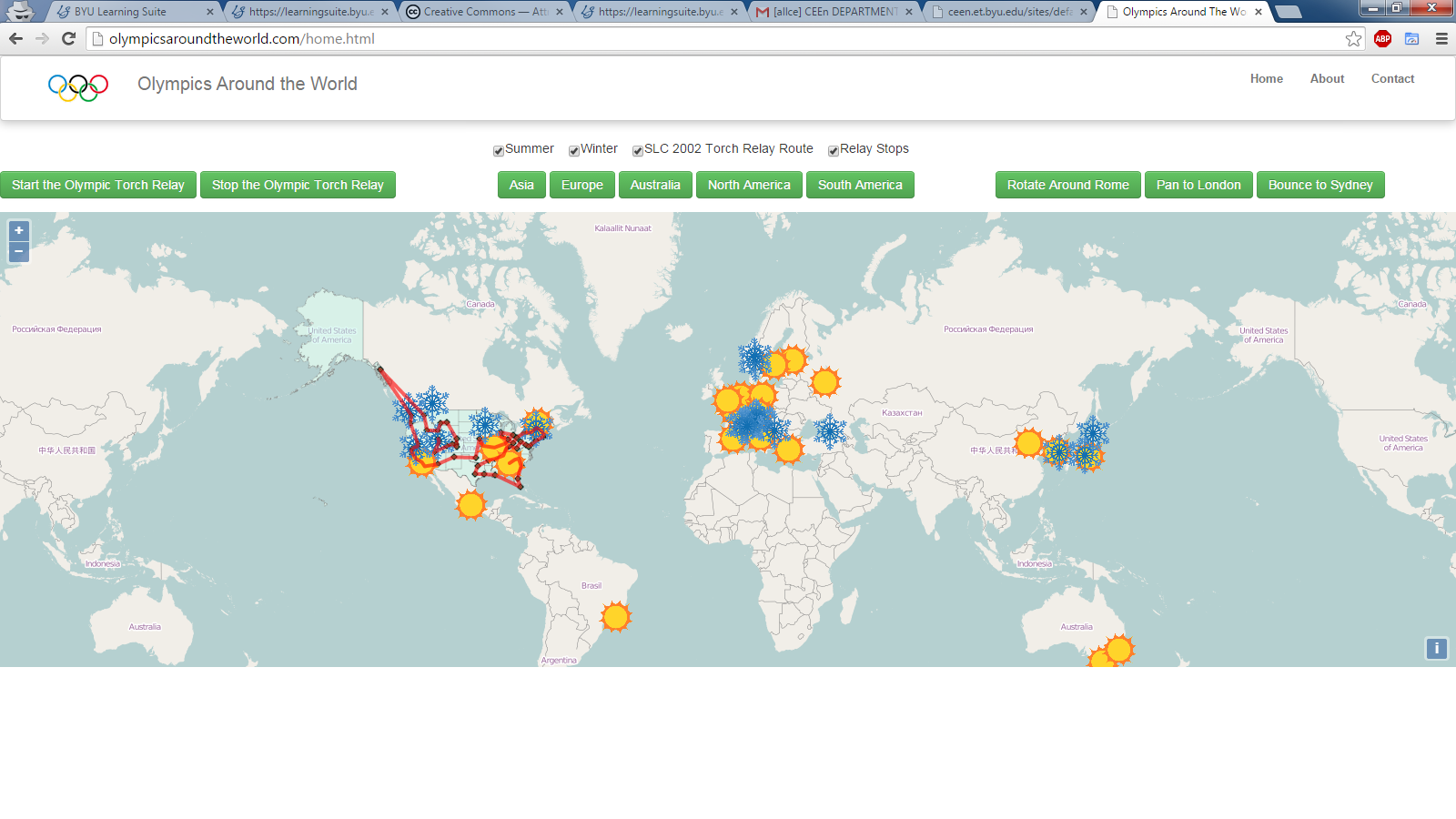


Figure : Interface elements

# Main Files

The main files of the website are home.html, about.html and contact.html along with the associated CSS files. Multiple folders were created to organize the openlayers, kml and image files. See Figure 3 for a visual of how the website folders were organized.

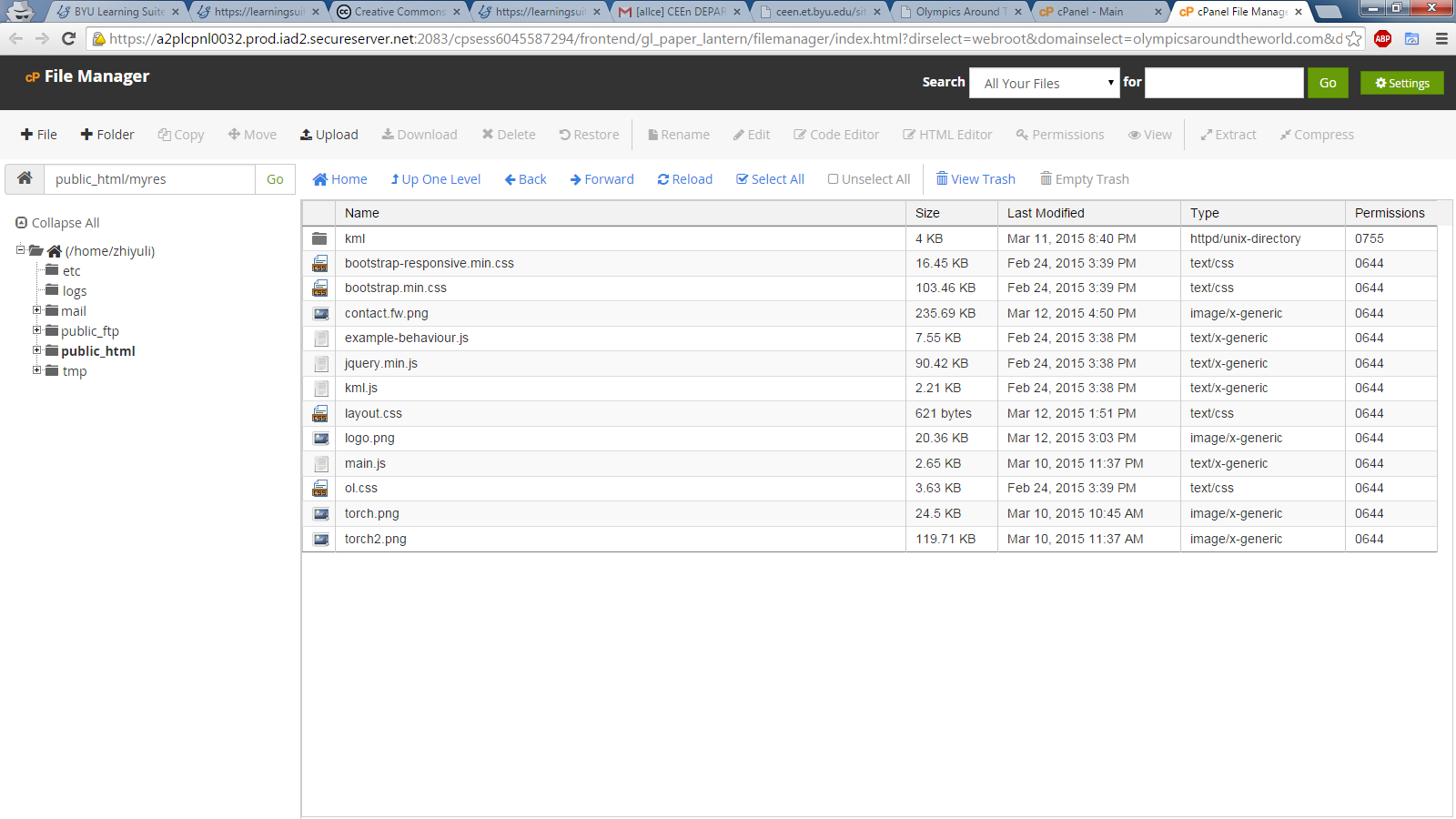


Figure 4: myres Folder

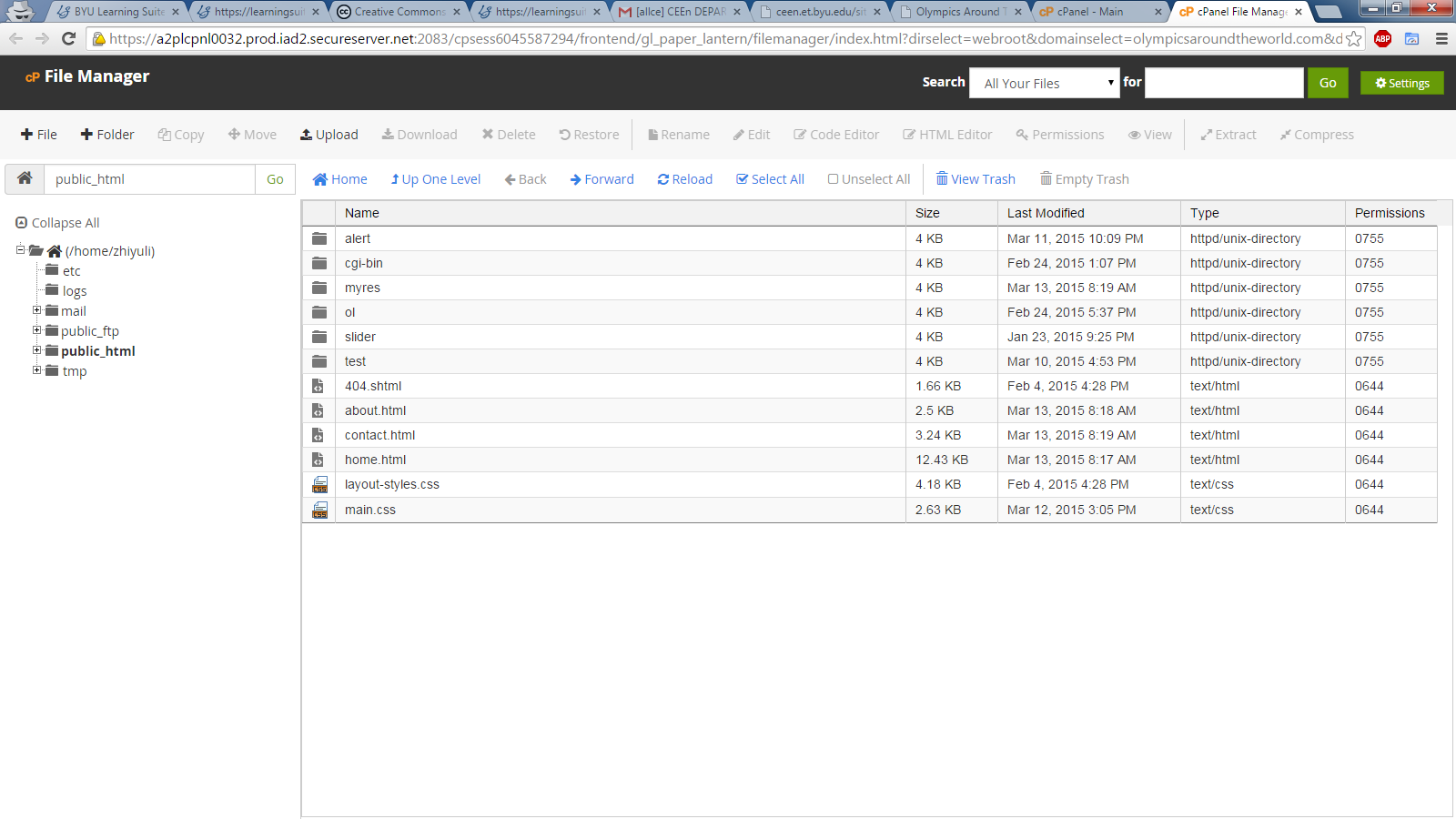


Figure 3: Main Working Folder

The Main folder doesn’t have too many files. It contains the main-css file and the html files for the different pages of the website. The HTML files use specified file paths to call upon the necessary files from the myres folder. The myres folder (See Figure 4) contains the KML files and all the images that were used in the project. It also contains the openlayers files that are necessary to generate the map.

# Code Structure

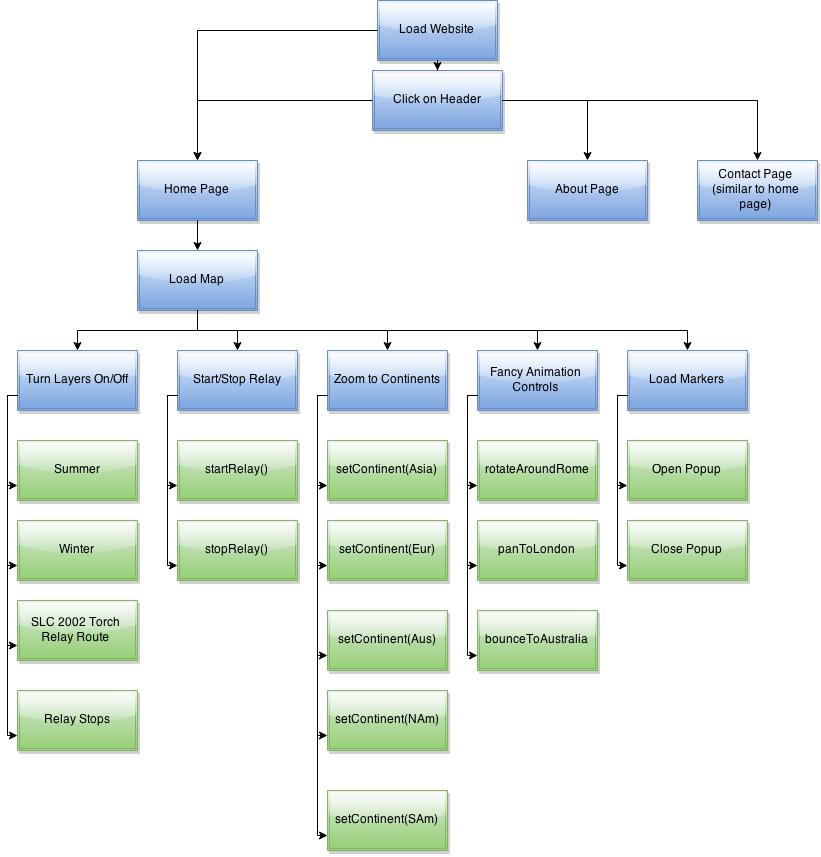
The basic structure of the code is represented in the flowchart below. The green boxes represent the custom functions that are being executed once you click on the elements. The custom functions are what make this website functional and robust. These functions will be explained in the next section.

Figure 5: Code Structure

# Key Functions

The following table contains all the custom JavaScript functions used in this website. They are presented in the order of their appearance.

Note: These functions are working due to the several vector layers that were created. They will not work if these layers weren’t created properly.

|  |  |
| --- | --- |
| **Function Name** | **Related Action** |
| setVisible(!vector.getVisible()) | On clicking this function will turn of the specified layer. Once the map was created, the following layers were all turned on. Then the checkbox was set to be checked by default. Then using this function the user can turn on and off the given layers. |
| closer.onclick | Provides the option for closing a popup box and on click it will close the popup box |
| rotateAroundRome | Using an event listener this function gets the current view of the map and rotates it to the preset coordinates of Rome. The coordinates can be tweaked to make this work for any city. Once the function as been called, a timer is used to ensure that the map goes back to normal. |
| panToLondon | Using an event listener this function gets the current view and pans to the London coordinates. Once the function as been called, a timer is used to ensure that the map goes back to normal. |
| bounce(t) | With the help of a few custom variables, you can manipulate JavaScript to animate in the direction/angle that you desire. This function was then called inside another function. |
| bounceToSydney | Using an event listener along with the bounce(t) function, this makes the map bounce to Sydney’s coordinates. Can be changed as desired. Makes use of a timer as well. |
| startRelay() | Makes use of a timer to control the runtime of the relay() function. The timer is set to 1000 seconds to allow the popup box to make its way around the path. |
| relay() | It’s a combination of several in-built open layers functions combined with custom JavaScript code. This function displays a popup box along the Torch Relay Path for the 2002 Salt Lake City Olympics. |
| stopRelay() | Clears the timer for the relay() function. Thus, allowing the user to see the popup from where it stopped. |
| setContinent(cont) | This function zooms into the continent once you click on the name of a continent. The cont parameter is retrieved from the layers that were set further in the code. This layer and the openlayers functions such as getView(), getSource(), getExtent() functions etc zoom to the set continent on the map. |

The source code is available upon request or part of it can be accessed by right-clicking on the web page and selecting view page source. This will display the HTML and the JavaScript code for the website.