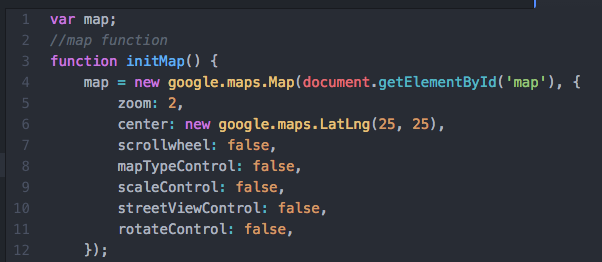
1. **INTRODUCTION**

The main objective of the website Mapping Economics is to present a tool that retrieves information from the World Bank Application Programming Interface (WBAPI). This web application displays two main indicators in Google Maps –Gross Domestic Product (GDP) and one indicator for External Debt (ED)– for each country from the database of the World Bank. The tool allows to select a specific country, an option of two indicators and a specific year from 2008 to 2015. This application requests several datasets from the World Bank API (i.e. list of countries with their corresponding ID, latitude and longitude; Gross Domestic Product annual growth; Total external debt stocks to gross national income; information for the years 2008-2015).

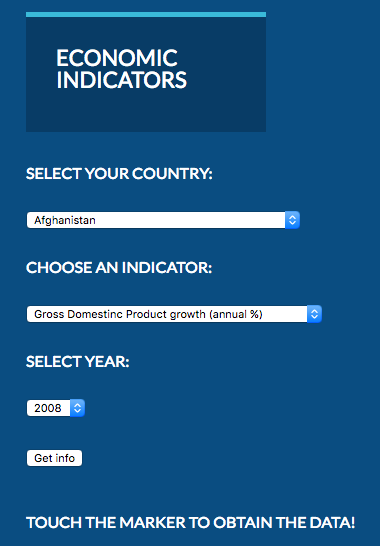
1. **MAPPING ECONOMICS TOOL**

The web application was programmed in JavaScript language. The code–in the file world\_bank\_dataset.js–combines the functionality of two APIs one with statistical data and the other for mapping purposes, i.e. World Bank API and Google Maps API. The application starts a function that makes a call to the Google Maps API, and after initiating the map, makes a request to the WBAPI.



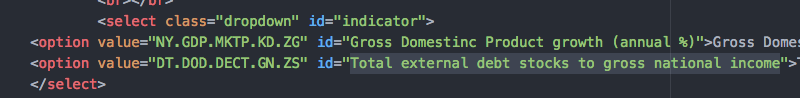
As it is shown in the first capture, the variable map is created, and then the function initMap is declared, which creates a new object, and declares the options that will request to display the map. In this case, most of the functionalities are disabled, such as the use of mouse scroll wheel, scale control, controls to change the style of the map, street view and rotate control. This way the user centres in the map and is capable of moving it or zoom in case of need.

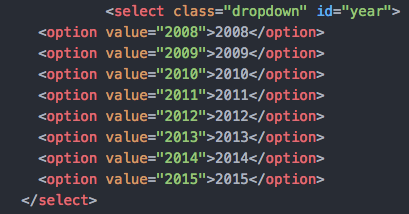
After that step, the program requests the input from the user to build a series of variables that will construct the basic call structure to request data from the World Bank. The website Mapping Economics offers an interface in which the user can input the country, a choice of two economic indicators and the year.



For this application, dropdown lists were chosen because it offered an accurate way to obtain the ID of each country, the ID of each indicator and the year, parameters required to make a precise call to the API, using the codes required by the World Bank and provided in their documentation (World Bank, 2017).







The program creates a variable and the listens the Get info button, to get the information after a click. Several variables obtain the information of each input through the command getElementsById. These variables are the constructs to make a basic call to the WBAPI. The WBAPI does not support Cross-origin resource sharing (CORS), hence it is not possible to make a json or xml request using this method. Therefore, a script is created to make a request of jsonp data, as it is shown in the next capture.

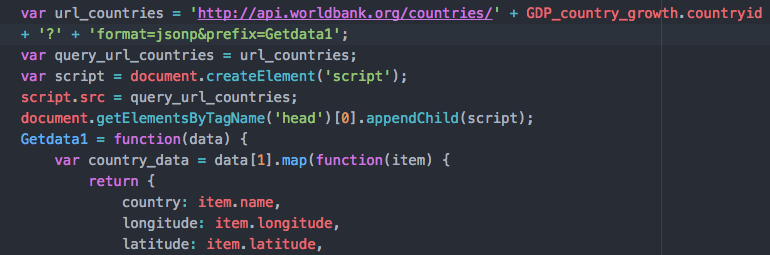


After the script is created, the program uses the variable Getdata to build a function that return the array with the information of the country requested, its ID, the value of the indicator, either GDP or external debt, as well as the year. According to an answer of a question made at the website StackOverflow (StackOverflow, 2016), the best method was to initiate a for loop function in the next step, in order to perform a closure in the next step, that allows to display correctly the values in each marker on the map.



According to the user “geocodezip”, closures “solve the problem of association between infowindow data with the geocoded location” in case the program wants to retrieve all the countries at the same time. In a first attempt, the program requested to the geocoding from Google, the latitude and longitude of each country, using the country IDs from the World Bank. Using this code, the user could retrieve all countries at once with each infowindow displaying the requested indicator.

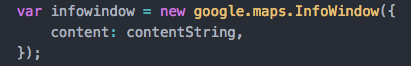
However, this method did not offer consistency between the country IDs from the World Bank and the coordinates offered by Google. The map showed geographic discrepancies, and therefore, the final code uses the country IDs from the World Bank to request of the latitude and longitude at the WBAPI. That offered a better consistency of results.



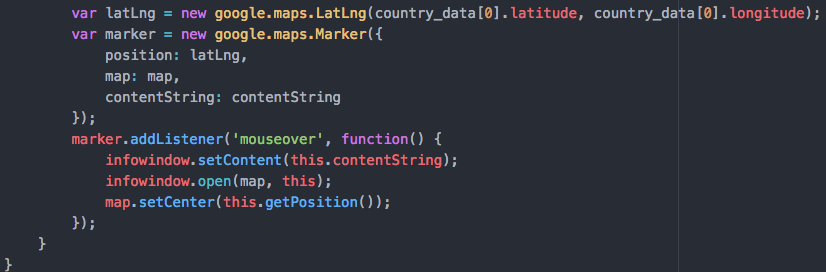
The trade-off was eliminating the option of retrieving all countries at once, preferring eliminating any discrepancy or error when the user requests a specific country.

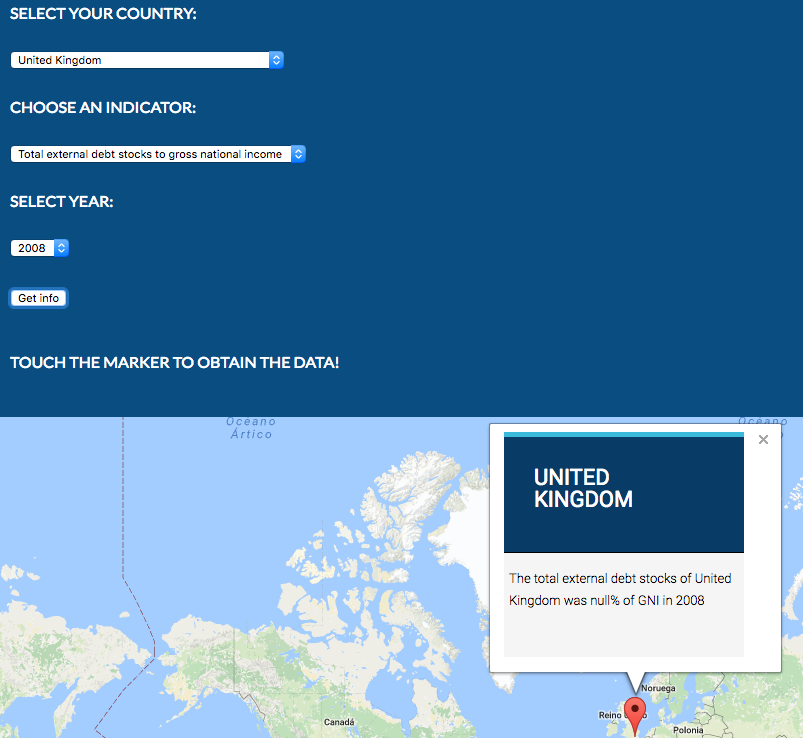
After the request of the latitude and longitude to the WBAPI, the program executes a functions that creates several variables that use all the information gathered by the API calls. Finally, using the country name, and the value of either the GDP or External Debt value, an if else conditional statement is used to show in the info windows of the Google Map, the desired information. Also, the variable includes html tags to format the information inside the info window with the same style as the rest of the website.

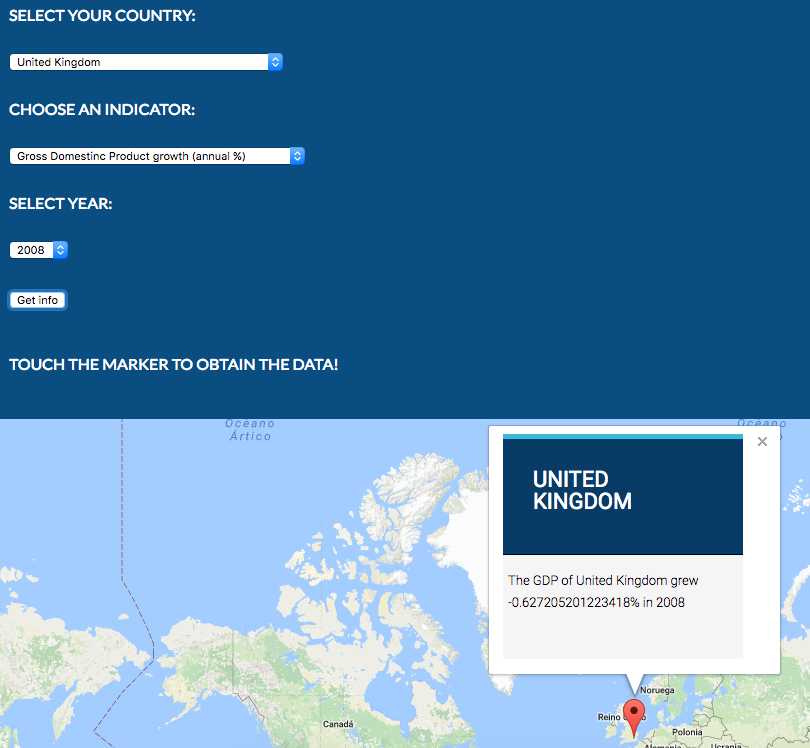




The program then uses the latitude and longitude as an input to place a marker in the map. The instruction of each marker is that it will centre the map and show an info window with the corresponding information for each country when the user places the mouse over the marker.







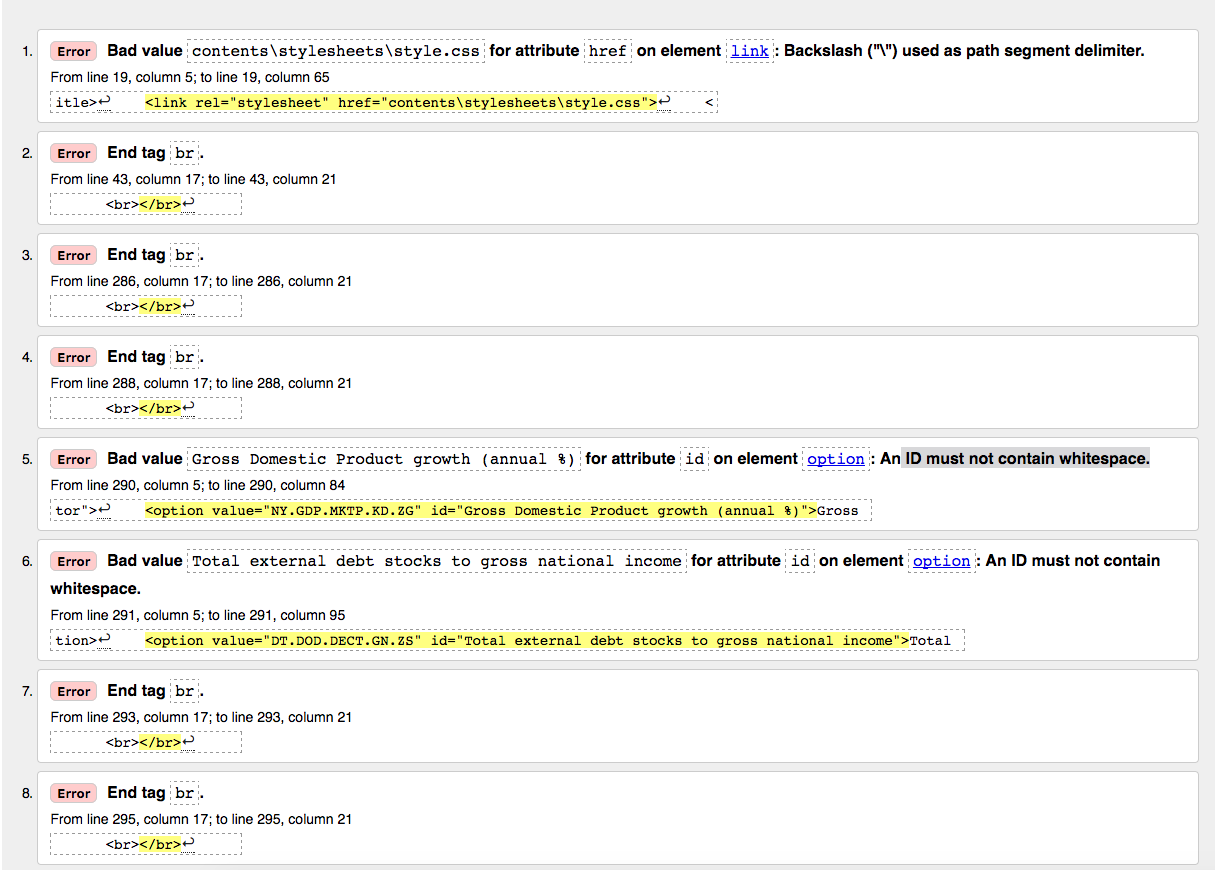
1. **DESIGNING MAPPING ECONOMICS**

The HTML and CSS design of the website had the aim to keep it as simple and elegant as possible. The site was inspired by The Guardian newspaper website, when choosing the blue colours for the navigation and main sections. The BBC News website inspired the selection of colours of the middle container and the footer. These two websites were chosen because of their impact and credibility among the public, and because the tool of Mapping Economics is intended to be useful for business journalists that need an easy access to key economic indicators.

The home page offers three options to select in the main page, with a title, and image to illustrate the topic and a short description. After one click, the user has access to the main tool of the website. The interface that rules the “hamburger” menu, was obtained from W3schools.com (W3schools.com, 2016) and adapted for this website. Only one CSS style sheet was used for the website, and only one JavaScript file.

1. **VALIDATION**

The validation of the HTML showed errors when elements “h” were not positioned in the correct hierarchy; bad values when using “backlash” in some links, and headings that are children of other headings. Also elements “a”, that can’t be children of elements “ul”. Also, the validation software at <https://validator.w3.org> showed mistakes with the end tag “br”, and ID attributes that contained whitespaces. Most of this errors did not affect the overall functionality of the website, but it is acknowledged that hierarchy facilitates reading code and make proper changes when needed, easier.



1. **EVALUATION**

Most of the learning outcomes were achieved with this project. The process by which webpages are delivered was recognised, as consistency was maintained throughout the website and to keep it functional and deliver a final HTML experience. Also in this second coursework, the ability to manipulate HTML, CSS and JavaScript was refined. Decisions were more focused to maintain consistency of objects and style in all the website, using a single CSS style sheet. It was improved the usage of the box model to keep a balance and symmetry with several objects, such as menus and main content.

Although the final HTML product was improved, with a cleaner interface and a more balanced, as well as symmetrical structure, showing a clean use of CSS code is still a challenge, since it is easy to loose track of all the variables involved, especially when several objects disappear in the process and many others are created. The trade-off is that having only one CSS style sheet reduced the work and allows symmetry and consistency in through the website. But more precision is needed in the use of tags, classes and ids to have an order that allows easy reading and reduce the time of response of any required change in a real life work environment.

As with JavaScript, it was achieved the outcome of accessing web APIs and retrieve several arrays of data from one source such as the World Bank and use another API from Google to display this information, and create functionality. For instance, the World Bank API offered the challenge to work with jsonp and organize the information in just three variables such as country, longitude and latitude. Then, it was possible to get another array of information with either the GDP or external debt value of each country, and map this information in Google Maps, using its API.

The website offers the user to make a specific request of three variables (country, indicator and year), the basic variables used to measure economic activity. This tool shows a real potential for expansion in the economic field. The Google Maps API offers many tools to build detailed choropleth map, so the user can have a better visual and understanding of economic indicators, to compare data.

Although the objective was achieved, there was functionality that had to be eliminated to in this tool, such the option of showing all the countries and have a fair comparison of the GDP or external debt. This objective can be achieved through a better understanding of tools such as Google Maps or libraries such as Leaflet.js. The potential of combining APIs offer more options to process information than packages such as Tableaux. Through the use of code and APIs it is possible to display any type of information, in real time. Whereas with pre-packaged software the developer is limited by the tools offered and restricted by the price for professional versions of software.

Also, it would be beneficial to learn more JavaScript to process data. For instance, the GDP value in the website showed too much decimal numbers, and a higher understanding of the code would allow to format this type of information, or even perform calculations such as correlations and map those results. One of this projects might include to show in a map the regions with a high correlation of inequality or economic deprivation with electoral preferences.

Moreover, JavaScript might add HTML functionality if this type of projects is combined with data visualization libraries such as D3 to show graphs in the same interface, useful to journalists that require easy access to information or tools to publish unique products.

1. **REFERENCES**

World Bank, 2016. Developer Information: Overview [Online]. [Accessed 18 December 2016]

W3schools.com, 2016. How TO - Responsive Top Navigation [Online]. Available at: <http://www.w3schools.com/howto/howto_js_topnav.asp> [Accessed 25 December 2016]

Stackoverflow.com, 2016. Javascript - For loop is not placing text in InfoWindows Google Maps API from array. Available at: <http://stackoverflow.com/questions/41355846/for-loop-is-not-placing-text-in-infowindows-google-maps-api-from-array> [Online]. [Accessed 18 December 2016]

(All images used from Creative Commons).