ASSIGNMENT COVER SHEET

Student Name : Azarias Boutin

ID Number: B00092351

Course : Rich Web application

Year : Third year

Lecturer : Orla McMahon

Title of assignment : Rich Web Applications : Assignment 1

Due Date : 5th November 2015

Date Submitted : Thursday 29th October, 2015

The material contained in this assignment is the authors original work, except where work quoted is duly acknowledged in the text. No aspect of this assignment has been previously submitted for assessment in any other unit or course.

Signed : Azarias Boutin 

Date : 29/10/2015

Abstract

In today's world, the web is a huge part of the market. Big companies such as Google, Microsoft, Yahoo all have a web search engine. The indexed Web contains at least 4.73 billion pages. Having a website is really important for all companies. But sometimes, little companies can't afford a web designer and a web developer. This is where Weasywig step in. The goal of Weasywig is to create web pages without any CSS/HTML knowledge. The user just have to add elements to his web page and therefore create his website. Moreover, if the company wants to add some more features to the editor, the API of Weasywig is here to help them.

Contents

List of Figures

I. Design of application

For this application, I wanted something simple and practical. The design chosen for the web application was a flat design. Because it is an editor, the application is not meant for the mobile. At least a tablet should be necessary to create web pages. The web application contains only three pages.

1. The homepage. The default page where the user arrive when he first open the web application. From this web page, he can access to the editor, open the existing differents projects. and to the documentation. On this page, the basics information of the company, such as the location (and so the map from google maps API).

2. The editor. The core of the web application. Where the user can edit his webpage and save it. From this page, the user can access the documentation, or come back to the homepage. Of course, once the project is saved, he can leave the web application. And open it back later to continue editing.

3. The documentation. This part of the web application is meant for developers who wants to improve the editor. Add their own blocks, customize the contextmenu, add their own states, learn more about the API.

II. Wireframing

1. Home

The homepage must be a simple page, containing the basics informations of the editor. The editor and the documentation must be accessible from it.

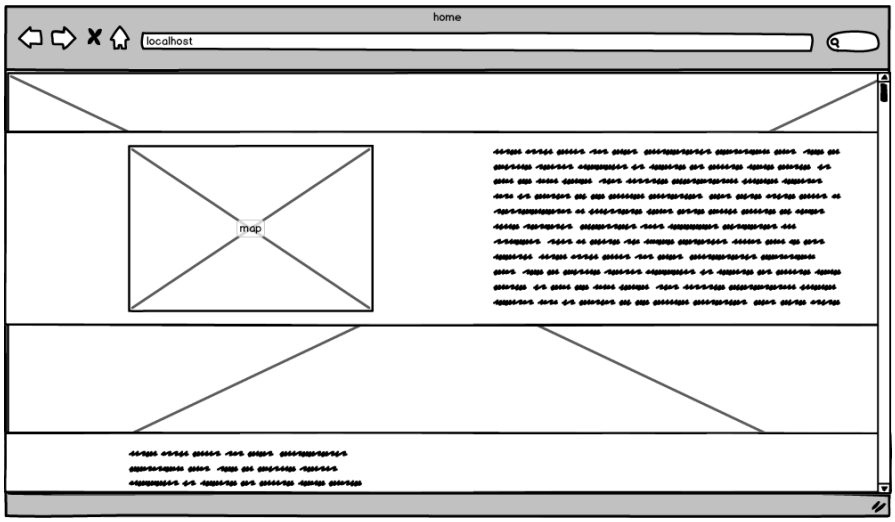

Figure 1: Home's mock-up

In this mock-up, we can clearly see that the homepage is divided into differents blocks. Each block has its own topic. One is about the location, one about the documentation and one about the editor.

To help the user identify his position, a fixed navigation bar will be added on the top of the page. And at the bottom of the page a "back-to-top" button will ease user's navigation.

2. Editor

The editor is the core of the web application. It must be simple, but must also provide as much tools as possible to the user.

We can see three distinct parts on this mock-up (figure 2) :

- **The center part.** This is where the user can add the content he wants. We can see a "+" sign displayed at differents places on the editor. This is where the user can add the content. Once the content is added, the "+" sign is removed. Of course, the user can create new editable block to create a bigger page.
- **The left sidebar.** This sidebar is where the choices of block are displayed. When the user will click on an editable div, this left sidebar will pop out, showing what types of blocks can be added.

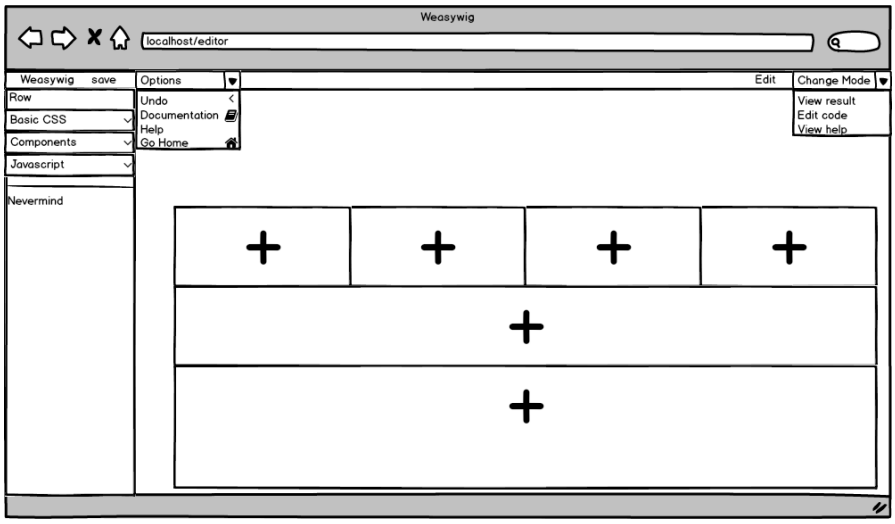

Figure 2: Editor's mock-up

- **The navigation bar on the top.** It is composed of all the others tools of the editor. Such as changing the main view, saving the project, undo the previous action, go to homepage, go to the documentation.

3. Documentation

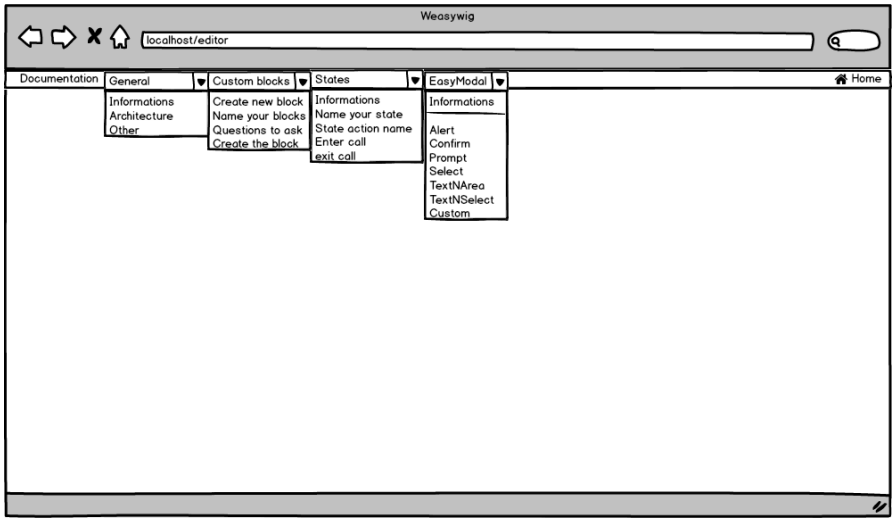

Figure 3: Documentation's mock-up

The documentation is a part meant for the developer who wants to improve the editor. Instead of having an almost-infinite scrolling page, the differents parts of the documentations are loaded dynamically. To help the user identify his position, a progressbar is added at the bottom of the page. At the top of the page, a navigation bar is displayed. This navigation bar contains dropdowns. These allows the user to access differents parts of the documentation very quickly.
The pictures displayed on the documentations have a tooltip so that when the user hover the picture with his mouse, a little help is displayed above or below the picture. When the user click on the picture, this one is displayed in bigger on the screen with his tooltip displayed at the bottom of the page.
The user can come back to the homepage after reading the documentation.

III. Operation of the project

1. The Editor

The editor is by far the most complex part of the web application. Here is an explanation of how the editor is working.

1.1. Architecture

The editor have an model-view-controller MVC architectural pattern. The model is the data-storage part. The view correspond to the display and the users interaction. And the controller is working with both the model and the view to display the good informations and modify it when needed.

The controller : There is a main controller for the editor. This controller is called 'Weasywig', the main functions of the editor are on this file.

The view : There is a view. The view is simply called 'view.js'. In the view, there are all the events listeners and the functions to interact with the DOM object.

The model : There is not a unique model object. The model is composed with the differents '.js'

files containing informations for the controller.Then, this data is given to the controller, who can manipulate it, and/or give access to the view.

Here's a sketch to understand how the editor architectural pattern is working.

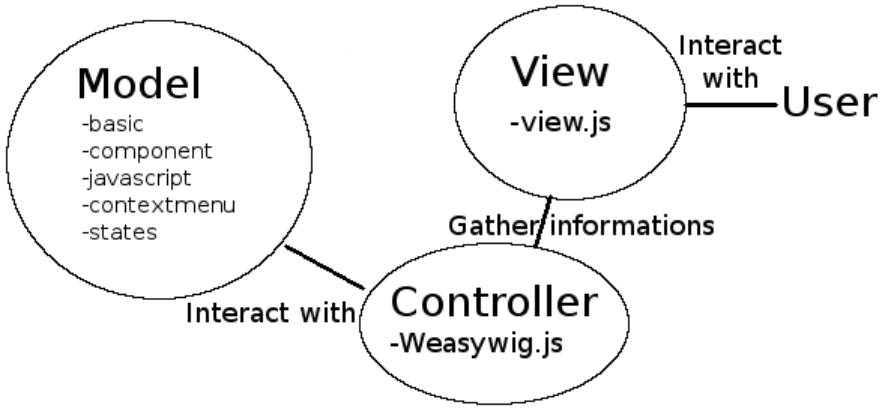

Figure 4: The MVC architectural pattern

1.2. Librairies

To help the developers improve the editor, some libraries are at his disposal. The main library is the easyModal one. Others are not really librairies, it's more plugins. Like the contextmenu and the states manager ones.

The easyModal library is a set of function to easyly create usefull modals.

The contextmenu plugin, is to customize the contextmenu wich is displayed when the user righ-cick on an element on the editor.

The states manager plugin let the developer create accessible states to navigate to when the user is editing his page.

1.3. Dependencies

The project require some basics css and js librairies. Of course, these libraries already are in the project.

jQuery : First of all, and the main dependency is the jQuery library. The minimum version must be 1.9

jQuery UI : To create some nice visual features, jQuery UI is a very usefull js/css library which is used in this project.

Bootstrap : Then to have a good css base, the well-known css framework bootstrap is necessary. The minimum used version must be 3.0

Underscore.js : To be able to do basic data-processing, the editor is using underscore.js this library offers a lot of usefull functions, and the documentation is really well-made. Developers can check the existing function if they need some of them during the creation of modules

Mousetrap : It is possible to use keyboard shorcuts in the editor. The handle of these shortcuts is made thanks to mousetrap this is a great library to handle keyboard inputs.

2. The homepage

The homepage contains three main interesting parts :

- The google map API usage
- The project's manipulation
- The navigation helper

2.1. The google map API usage

To locate where the industry is, the google map API is very useful and very simple to use. The documentation is complete and the example are well-made.
In the map, the user must-right click on the map to set his origin point and see what is the path to follow to go to the company. Once he did that, the accordion under the map will open and display the path informations.

Also, when the user clicks on the company location, a bubble will pop-up above the location to display informations about the company.

2.2. The project's manipulation

It is possible to create a new project or to delete an existing one in the homepage.
The user can create a new project by clicking the 'create new project' button. He will be redirected to the editor and will be able to begin to create his page. And then save it to access to it later.
The user can also access the existing projects he saved before. He just has to click on the button with the project name on it, and he will directly have access to the saved project.
Finally, the user can delete the existing projects. By clicking the 'delete' button. A confirmation will be asked to the user and if he accepts, the save will be deleted.

2.3. The navigation helper

To navigate more quickly between the differents blocks of the homepage, a navigation bar is displayed at the top of the page. The user can click on each links of it, and a scroll animation will begin to bring him to the part he asked for.
When the user scroll on the document without using the links, they will be updated anyway so that he can know where he is on the page.
And finally, when the user arrive at the bottom of the page, a simple button with 'back to the top' written on it is displayed to bring back the user to the top of the page.

IV. Preview

Here are three screen shots showing the three differents parts of the web application.

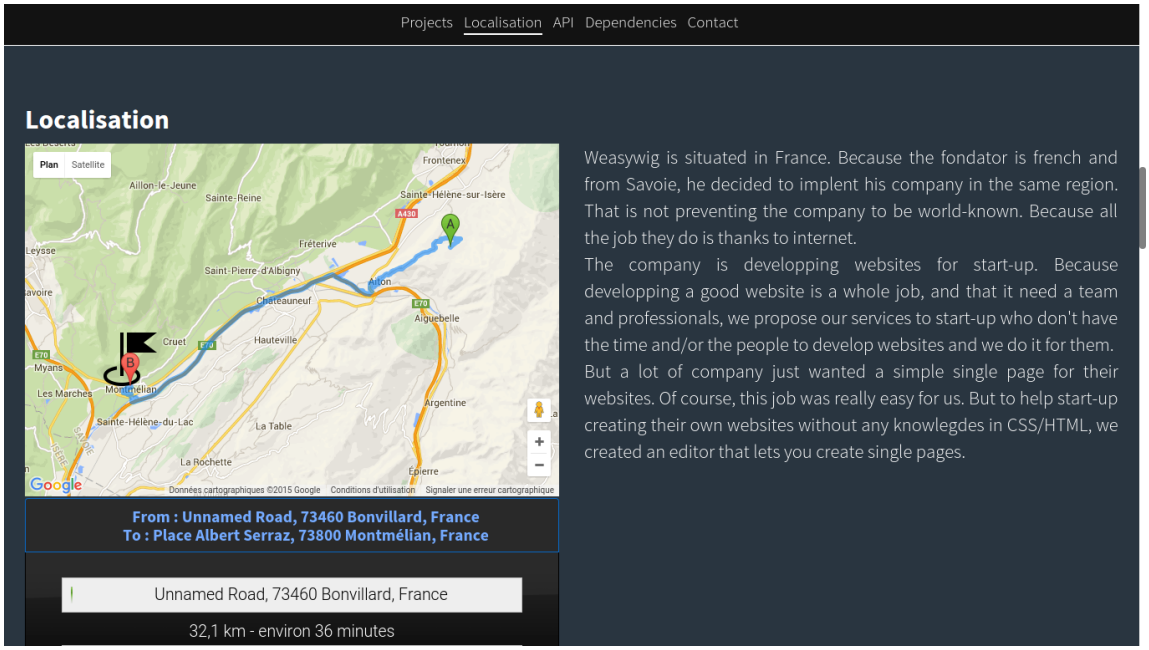

Figure 5: The homepage preview

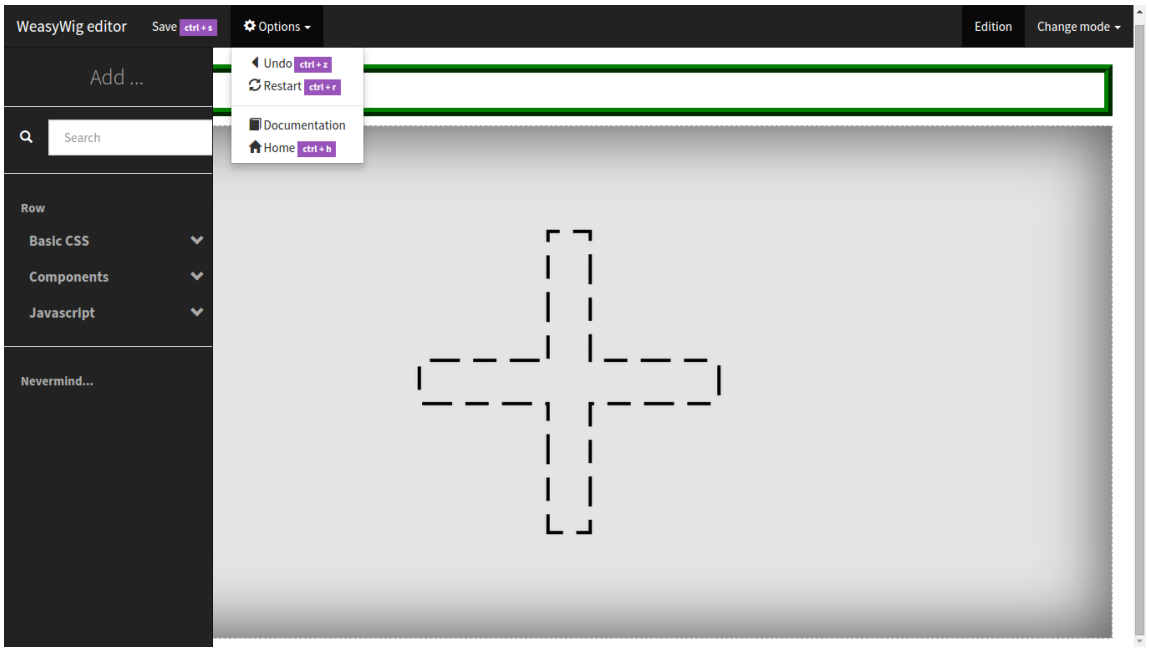

Figure 6: The editor preview

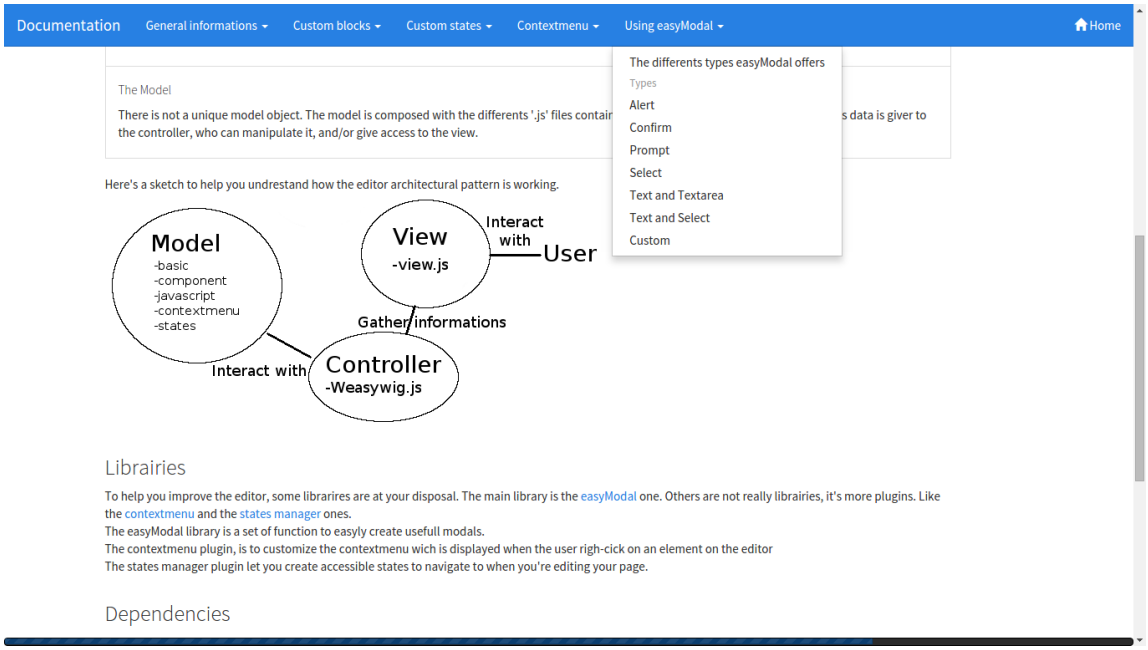

Figure 7: The documentation

V. What I learned from the project

1. In details

Here is the list of what I learned from this project :

- It is possible to access every function/data of the window without using window. For instance instead of *window.alert*, only *alert*
- Because the window is global, it is easy to add data to the web application and give access to javascript objects who needs it.
- jQuery features that I didn't know existed :
  - return false on an event prevent the default event
  - it is possible to break out a *$.each* loop by returning false
  - it is possible to use *toggleClass* function with a boolean to test if the class must be toggled or not

  - the *$.extend* function to merge two objects that can have same attributes

- How to easily create a simple parallax effect with jQuery.
- How to call a javascript function with an unknown number of arguments using *function.apply*.
- How to use Scss to write CSS faster and easier
- How to use SVG and CSS to create fancy animations

2. In general

This project reinforced my javascript and jQuery knowledges. I also learned how to be more efficient with tools such as Scss wich is a very handy way to style the page.
The hard part of the project was to create an easy-to-use API and accessible to everyone who wants to improve it. I had to do multiples improvements before being satisfied of the result.
I first used Backbone.js to handle the view. But when I realised I was using it only for the view, and not for the model, I decided to create my own little library in the view with the events and the shortcuts.

To put it in a nutshell, this projects helped me deepen my javascript and jQuery knowledges.

Glossary

API Application programming interface.

CSS Cascading StyleSheet.

DOM Document Object Model.

HTML HyperText Markup Language.

MVC Model View Controller.

SVG Scalable Vector Graphics.