EXPLORATION OF RIA TECHNOLOGIES AND DEVELOPMENT OF A WEB-BASED DIAGRAMMING TOOL

Ngo Thanh Vu

December 13, 2013

Contents

List of Figures

List of Tables

Acknowledgement

I dedicate special gratefulness to my instructor, Dr.Do Lenh Hung Son for guiding and assisting me during the time i was working with him. I would like to thank all members in the Advanced Program of Computer Sciences including my classmates, seniors, professors, and all members in the Teaching Staff of Faculty of Information Technology and Academic Affairs. They play a very important part to the success of my thesis

Abstract

Since 1999, when the term ”Web 2.0” was introduced , which is associated with a richer web facilitating social media, user-generated content as well as interaction and collaboration, business applications have become increasingly web-oriented . While the web in its roots was limited to passive viewing of content created by others, it has moved towards an application platform for desktop-like applications within the last decade. Utilizing these advantages, in this thesis, I present a web application to design diagrams which can be considered as a Rich Internet Application (RIA) and Single Page Application (SPA). I also took a survey to explore and choose state of the art technologies which support to build such web application.I demonstrated by building an web application using the chosen technologies and integrating them . I also introducing about RIA and SPA as well as HTML5, CSS3, AngularJS (Javascript Framework), GoJS (Diagram Library), and other technologies i have been using for this application.

Chapter 1

INTRODUCTION

Nowadays Internet is increasing its popularity dramatically, people tend to work online with web browsers more than ever[1, 2, 3] . A survey in early 2010 [4] showed that 74% of American adults and 93% of teenagers of age 12-17 use the Internet. More specific, email and search engines are the two most popular services that people do online[5]. Web browsers play a key role of being a door to link users to an enormous source of information: websites. Therefore, The popularity of web browser is premise for the emerging of web applications which uses web browser as a client. The ability to update and maintain web applications without distributing and installing software on potentially thousands of client computers is a key reason for their popularity, as is the inherent support for cross-platform compatibility[7].

Low-level JavaScript libraries like jQuery[1], Prototype[2], Underscore.js[3], etc., provide a convenient API for manipulating the Document Object Model (DOM) in a uniform way across different browser implementations. However no means to structure the application code are provided, usually resulting in lots of DOM element selector and callback code[8]. This ”spaghetti code of the 21st century” [9] is reminiscent of software development in the 1970s and leads to poor maintainability of applications. Thats where web application frame-works come into play. Besides disburden developers from writing boilerplate code, frameworks can structure web applications enforcing the separation of different application parts. According to the Model-View-Controller design pattern [10, 11] frameworks usually separate user interface definition, application data, and business logic. Due to this modularization,

[1] http://jquery.com/

[2] http://www.prototypejs.org/

[3] http://documentcloud.github.com/underscore/

frameworks support the fast development of applications and promote reuse as well as maintainability. As of today, several frameworks are available.

The extensive usage of JavaScript in today 's web application induces the need for frameworks supporting faster development, better reusability and maintainability. As Model-View-Controller(MVC)is a well-known design pattern for server-side application development. It becomes even more important on client-side leading to several prevalent JavaScript MVC frameworks[8]. Therefore client-side JavaScript application frameworks will be of great importance for the development of futer web-based business application.

The goal of this thesis is to analyze existing JavaScript frameworks with respect to various criteria such as structure, data-biding, Testing ability,ect. From the set of reviewed frameworks, one should be selected to design and implement a web application for edit and design interactive diagrams. Besides, JavaScript libraries supporting implementing interactive diagrams are also analyzed and selected. Other selected technologies for front-end development and database are also integrated. This thesis proposes a way of survey and select as well as integrating state of the art technologies into a web application.In fact, there are web applications supporting edit interactive diagrams. I can list several famous name as follow. Big guy Google's draw.io[4] has his own online diagram drawing application which use MxGraph[5] and his own cloud storage to implement. Creately[6] provide an environment for designing diagrams with support of collaboration implemented with Adobe Flash[7]. Gliffy[8] and Lucidchart[9] are also famous for their beautiful diagrams with collaboration support. Although the features demonstrated cannot be fancy and various as in those example application , This thesis is still an evidence of using JavaScript frameworks and cloud storage actually makes web application development less complex, less cumbersome and more maintainable than in the past.

[4]https://www.draw.io

[5]www.jgraph.com/mxgraph.html

[6]http://http://creately.com/

[7]http://www.adobe.com/software/flash/about/

[8]http://www.gliffy.com/

[9]https://www.lucidchart.com/

Concretely, This thesis contains

1. Conducting surveys about JavaScript frameworks and JavaScript Libraries which support implementing
2. Exploration of selected technologies
3. Implementing a web application using selected technologies
4. Evaluation about the advantages that those technologies brought as well as some future works remain to be done

This thesis consists of six chapters. Below is each chapters preview:

Chapter 1- Introduction. A brief introduction about the trend of web application as well as the importance JavaScript frameworks in web application development, their advantages and briefly describe what i do.

Chapter 2 - Technology survey. A brief introduction about RIA and SPA. I present surveys about JavaScript frameworks and JavaScript graph libraries, Their result and reasons why i choose them

Chapter 3 - Exploration of key technologies. Describe in details about AngularJS, GoJS, Mongolab, Angular Boostrap, Jquery, Semantic-UI, their advantages and reasons for choosing them.

Chapter 4 - Building application using selected technologies. Describe in details how i implemented using selected technologies.

Chapter 5 - Discussion & conclusion. I present further disscussion about the web application, pros and cons, vision for future works. i end the thesis by summing up what i have done.

Chapter 2

TECHNOLOGY SURVEY

In this chapter, the concept of Rich Internet Application(RIA) and Single Page Application(SPA) is introduced. Besides, I need to define some criteria based on what Javascript framework can bring. Consequently, i conducted surveys about JavaScript frameworks and JavaScript Graph libraries in order to support the web application development

2.1 RIA and SPA

2.1.1 Introduction

Since 1990, when the first web browser prototype called "WorldWideWeb"[12] was released by Tim Berners-Lee, the web has evolved from a simple document sharing system to a multimedia content distribution and application runtime environment. The major evolution step which is depicted in the picture below. At first, web pages were simple documents contain nothing but text and images.Users could navigate between different pages by hyperlinks.

At that time the capability of the web was limited. Due to emerging software development capabilities, the web was used increasingly as application platform in the second period. This is the time when the term **single-page application** also known as **single-page interface (SPI)** appears. It is a web application or web site that fits on a single web page with the goal of providing a more fluid user experience akin to a desktop application.

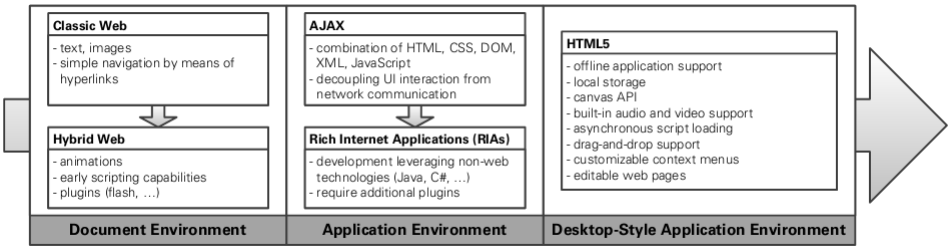

Figure 2.1: Major evolution steps of the Web[13]

Asynchronous JavaScript and XML (AJAX), introduced by Jesse James Garret in 2005[14] ,changed the primary interaction model of the web and therefore increased its use as an application environment. However, the browser at this time was not supplied enough comprehensive sets of **API** and complex graphic capabilities.To provide such sets of APIs which were similar to the desktop application, **RIA** is introduced. It is a Web application that has many of the characteristics of desktop application software, Users generally need to install a software framework using the computer's operating system before launching the application, which typically downloads, updates, verifies and executes the RIA[6] .Examples for RIA platforms are Adobe Flash,Java FX

and Microsoft Silverlight.

2.1.2 The current approaches

JavaScript

Currently JavaScript is the predominant implementation technique for plug-in free applications in the web. JavaScript applications can be built by means of pure JavaScript, leveraging low-level libraries like jQuery or high-level frameworks like Knockout.js. Since applications built by means of JavaScript use standardized web technologies, they do not require a dedicated plug-in as runtime environment.

Non-JavaScript

Non JavaScript, but plug-in based applications require the installation of additional browser plug-ins serving as runtime aaenvironment. Example technologies belonging to this category are Adobe Flash , Java FX and Microsoft Silverlight.Some non JavaScript, but plug-in free implementation techniques are available too. Some of them enable the developer to write the applications code in a language abstracting from concrete web technologies like HTML, CSS, etc. Two example technologies are Google Web Toolkit enabling web application development in Java, and CoffeeScript.

2.2 JavaScript Framework Selection

2.2.1 Overviews of some JavaScript frameworks

JavaScript Library view

we may define this group consists of the JavaScript framework which slots into your existing architecture and add specific functionality. Example to this kind are **Knockout.js, JQuery, JQueryUI**.JavaScript widget libraries such as **Ext JS , DHTMLX , Dojo Toolkit** were developed, allowing for developers to concentrate more upon more distinctive applications of Ajax.

JavaScript Framework view

This kind of framework gives you an architecture (file structure, etc.) that you are meant to follow and, if you do, are intended to handle all common requirements. Some example are **Ember.js , Angular.js , YUI, Backbone.js**

2.2.2 Framework Selection Characteristics

These are some criteria for choosing the framework

- The license under which the framework is released
- The size of the framework
- Community: the number of users or posts in the forum they use or on GitHub(Jan 2013) indicating the community support, as well as the website of the framework usually providing tutorials and a more or less comprehensive documentation.
- Programming language
- Js include indicates whether we need to include a js file into the website or not
- Basic UI component support: indicates that whether the framework provides their own basic component like text field, check box, combo box, form, date picker, ...
- Advanced UI widget library indicates that whether the framework support advanced components like grid view, tree, auto-complete,

2.2.3 Survey of Existing Frameworks

Google Web Toolkit(GWT)

- Overview: **GWT** provides an open source set of tools that allows web developers to create and maintain complex JavaScript front-end applications in Java. Besides,it supports writing both the client-side code and the server-side code in Java
- Pros:

Figure 2.2: An example created by GWT

  - supports ability to debug
  - Strong community
  - good documents,demos, samples
  - supports lots of features
  - No JavaScript syntax errors
  - it's good for those who had java background and doesn't really appreciate the power of JavaScript (for both GWT and Vaadin

- Cons:
  - kind of a little too few stunning UI components comparing with Vaadin
  - don't see much demo or support about drag drop

Vaadin

- Overview: features a server-side architecture and client-side is built on top of GWT.
- Pros:
  - Wide variety of UI components

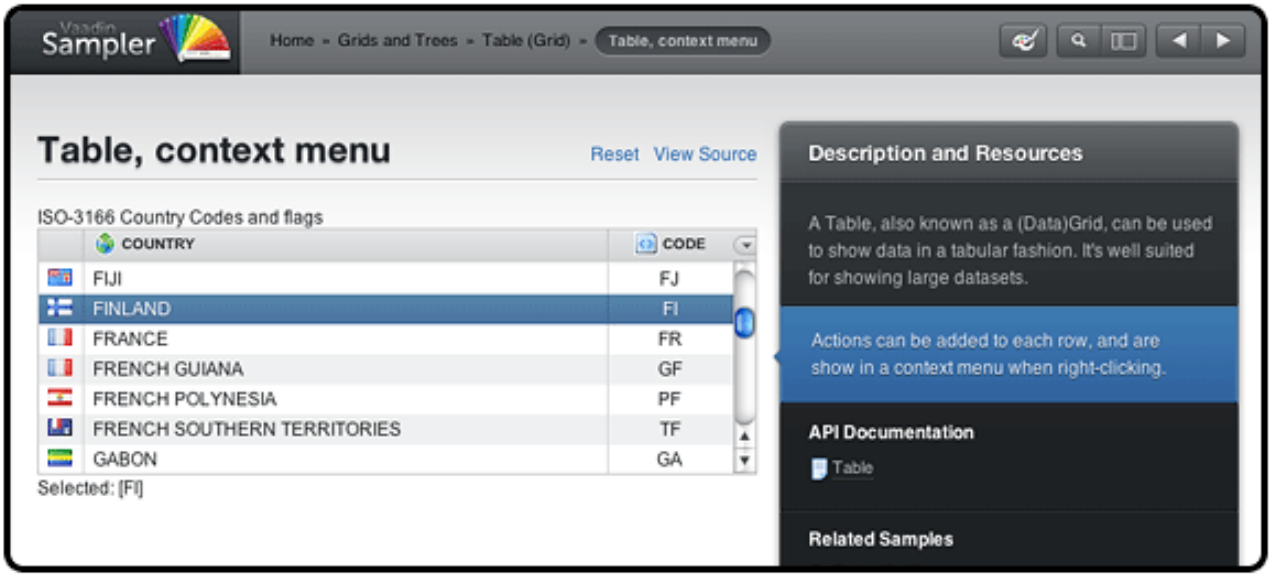

Figure 2.3: An example created by Vaadin

  - Drag and Drop for Tables, Panels and Trees components have real nice demos and tutorial
  - Simple to learn, it has a book, a road map, a forum
  - With Vaadin, you can build a nice app with almost a half of lines of code and still the application seems more complete compare with GWT

- Cons:
  - for advanced customization you'll need to write more in CSS, HTML, JavaScript
  - Client side is not extensible and very chaotic
  - html files is really heavy and takes long render time due to lots of components are added

SproutCore

- Overview: an open-source framework for building blazingly fast, innovative user experiences on the web supported by Apple. It is also one of the largest frameworks.

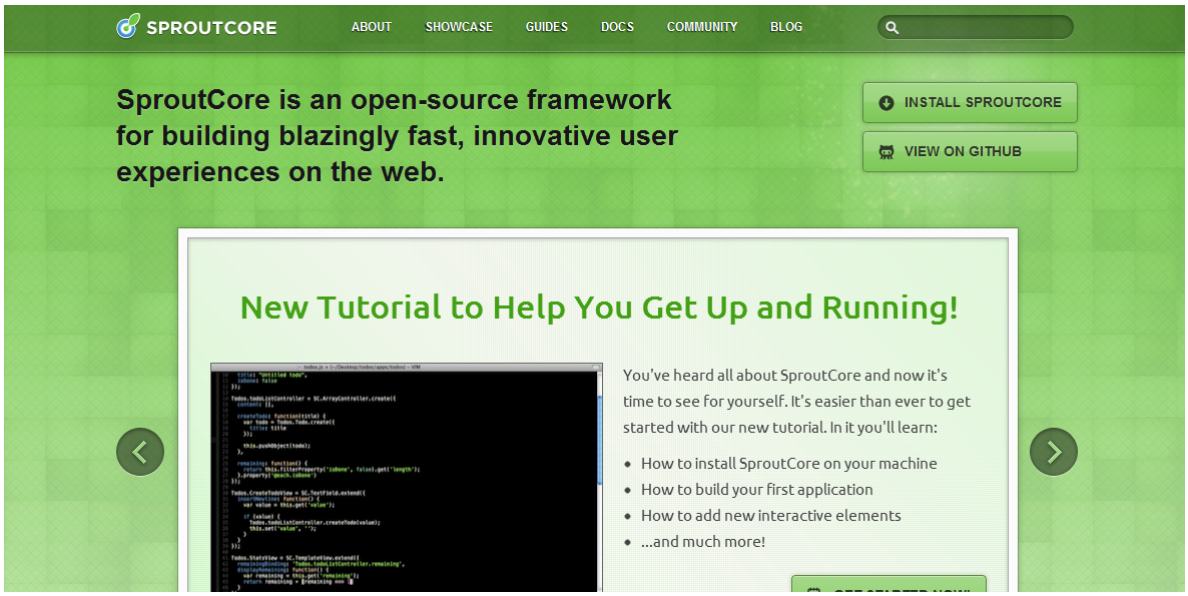

Figure 2.4: An example created by SproutCore

- Pros:
  - MIT license
  - Bindings support.
  - Solid community.
  - Tons of features.
- Cons:
  - Overly prescriptive.
  - Hard to decouple from unneeded features.
  - Forces a native-like paradigm

Dojo

- Overview: DOJO is one of the leading JavaScript framework.
- Pros:
  - documents and tutorials in details
  - has all the components you would most likely use

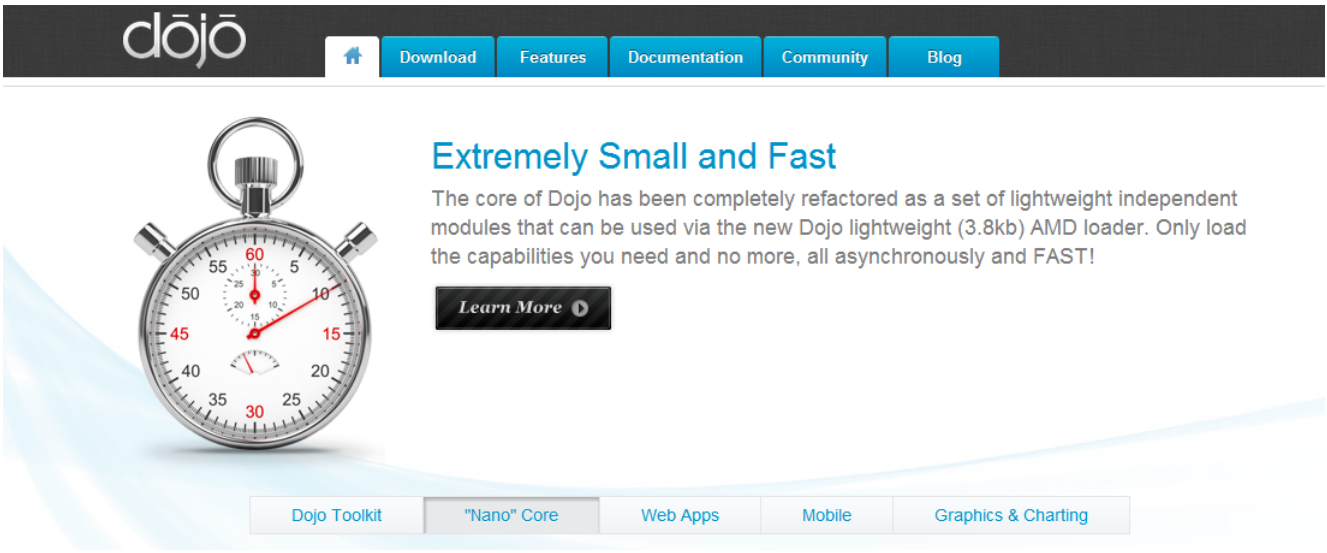

Figure 2.5: An example created by Dojo

- Cons:
  - API stability
  - Demo is not very clear and scatter
  - Many have commented that Dojo seems difficult to learn and get started with

JavaScriptMVC

- Overview: an open-source framework containing the best ideas in jQuery development,a collection of the best practices and tools for building JavaScript applications. Built on top of jQuery"
- Pros:
  - using Controllers can make a clean, tight code that is easy to find
  - is very lightweight, it's jQuery based
- Cons:
  - No preset UI layer implementations

Figure that created by JavaScriptMVC

JQuery

- Overview: JQuery is free, open source software, licensed under the MIT License. It's also one of the best JavaScript frameworks at presence.
- Pros:
  - Easy to use
  - Large library
  - Strong community
  - Great documents and tutorials
  - Ajax support
- Cons:
  - Functionality maybe limited

Cappuccino

- Overview: an open source framework that makes it easy to build desktop-caliber applications that run in a web browser which look and feel like desktop applications on Mac OS X.
- Pros:
  - allow you to create true desktop-like apps right inside the browser

Figure 2.6: An example created by JQuery

  - dont rely on a continous web connection
  - as fast as desktop app
  - you can build asyncronous, offline, robust web apps right inside the browser
  - Cappuccino is compatible with many of the latest browsers

- Cons:
  - Adding a layer of abstraction (Objective-J objects to Javascript) also adds to the overhead. The result is that the application can be slow, particularly if the user's computer is slow.
  - It is not designed to make full-fledged web sites. That leads to complex and interactive sites may not suitable
  - is still very early in its development stages, so some parts are still unimplemented
  - Different Underlying Model

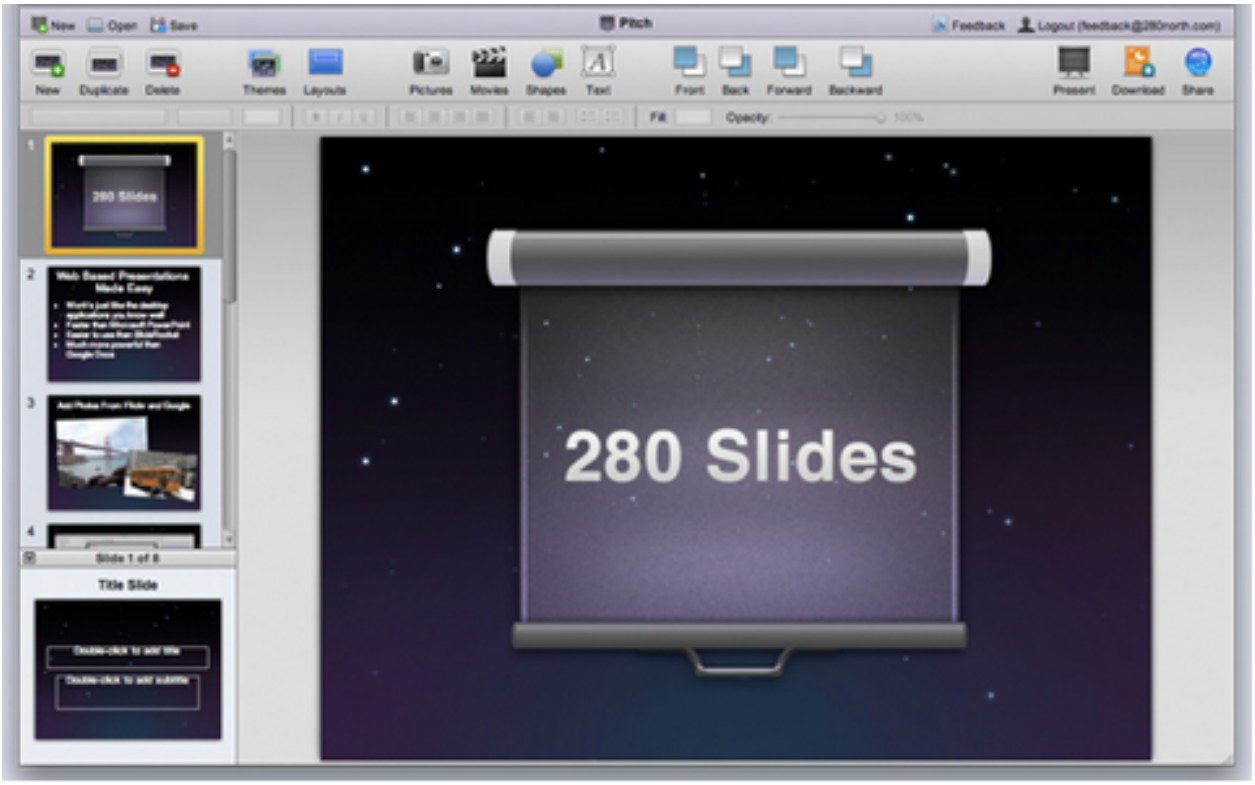

Figure 2.7: An example created by cappucino

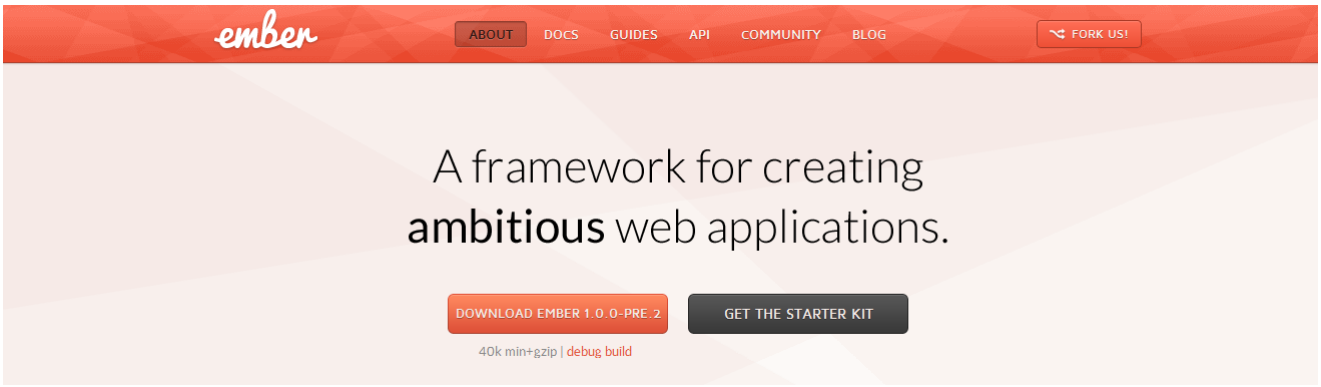

Figure 2.8: An example created by ember

Ember.js

- Overview: Ember is a JavaScript framework for creating ambitious web applications that eliminates boilerplate and provides a standard application architecture. Same company with Sproutcore.
- Pros:
  - easily implementing MVC functionality.
  - extremely easy to create computed properties in JavaScript
  - It is designed so you don't have to worry about whether or not you have 2000 bindings.
  - is intended for "web-styled" applications.
- Cons:
  - Documents are not quite clear and understandable.

Flame.js

- Overview: It's a widget/UI library for Ember.js, so it has all the properties of Ember.js listed above.

Angular.js

- Overview: an open-source JavaScript framework.

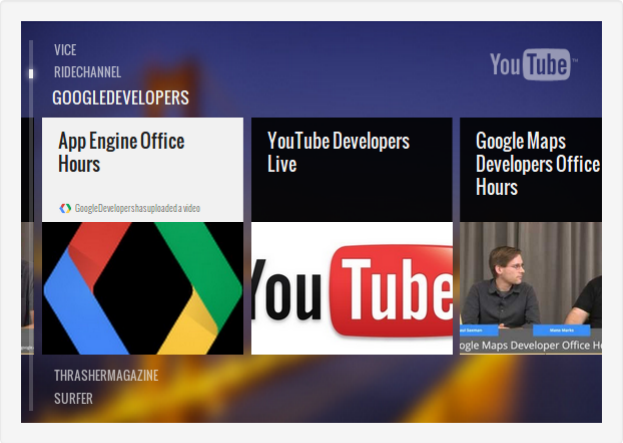

Figure 2.9: An example created by AngularJS

- Pros:
  - You don't have to write all the event listening and event triggering. Its automatic.
  - if you want something declarative that uses the View to derive behaviour.
  - focuses on achieving this through custom HTML tags.
  - great for small- to intermediate-scale applications
- Cons:
  - Obtrusively mixes all controller, model and view into the html
  - invents its own syntax which requires learning
  - Javascript errors happen if you DON'T clutter the window object.
  - the more bound elements in your app, the slower it gets
  - Difficult to allow browsers to optimize this in native code.
  - It only provides an interface between a combined M, V and C. What if you wanted a Model to subscribe to another Model but didn't want to update a view. You can't do this with Angular.js, but you can with Backbone.

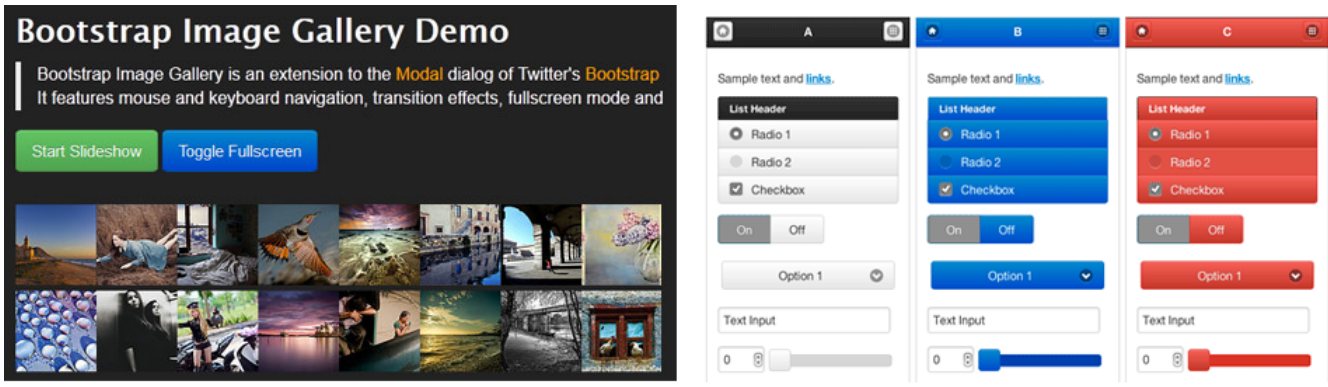

Figure 2.10: An example that created by Twitter Bootstrap

Twitter Bootstrap

- Overview: is a free collection of tools for creating websites and web applications. It contains HTML and CSS-based design templates for typography, forms, buttons, charts, navigation and other interface components, as well as optional JavaScript extensions.
- Pros:
  - The framework can be adapted to any CMS or blogging platform like WordPress, Drupal or Joomla.
  - Easy visual consistency in your application.
  - This framework is designed for a web application (not a website) and all the elements fit nicely together to get the app done fast.
  - many beautiful theme”
  - Speed, Consistency across applications
  - People find it simple and elegant”
- Cons:
  - incomplete support for HTML5 and CSS 3, but it is compatible with all major browsers.
  - It uses Less css. but many people like to use other tool to mangage css

Ext JS/Sencha

- Overview: a pure JavaScript application framework for building interactive web applications using techniques such as Ajax, DHTML and DOM scripting.
- Pros:
  - is like a superset of the widgets like simple label, textbox buttons to complex grids, drag-drop panel
  - It has quite good documentation with tutorials, samples and user community.
  - Active and currently most adopted javascript RIA framework
  - Good code quality/readability
  - Ext JS makes it simple to edit tickets inline
- Cons:
  - Loading time would is high for home page on web.
  - CSS very easy to get lost. It is difficult to find correct class names.
  - HTML full of divs and overly complex generated code. Difficult to debug even with FireBug.
  - Customization is not easily achievable.
  - Loading even simple things requires few lines of coding which is simpler in plain html or jQuery
  - Need quite experienced developer"

quooxdoo

- Overview: ooxdoo is a universal JavaScript framework with a coherent set of individual components and a powerful toolchain
- Pros:
  - supports namespace,eventbiding,cross-browser back button, bookmarkability, AOP

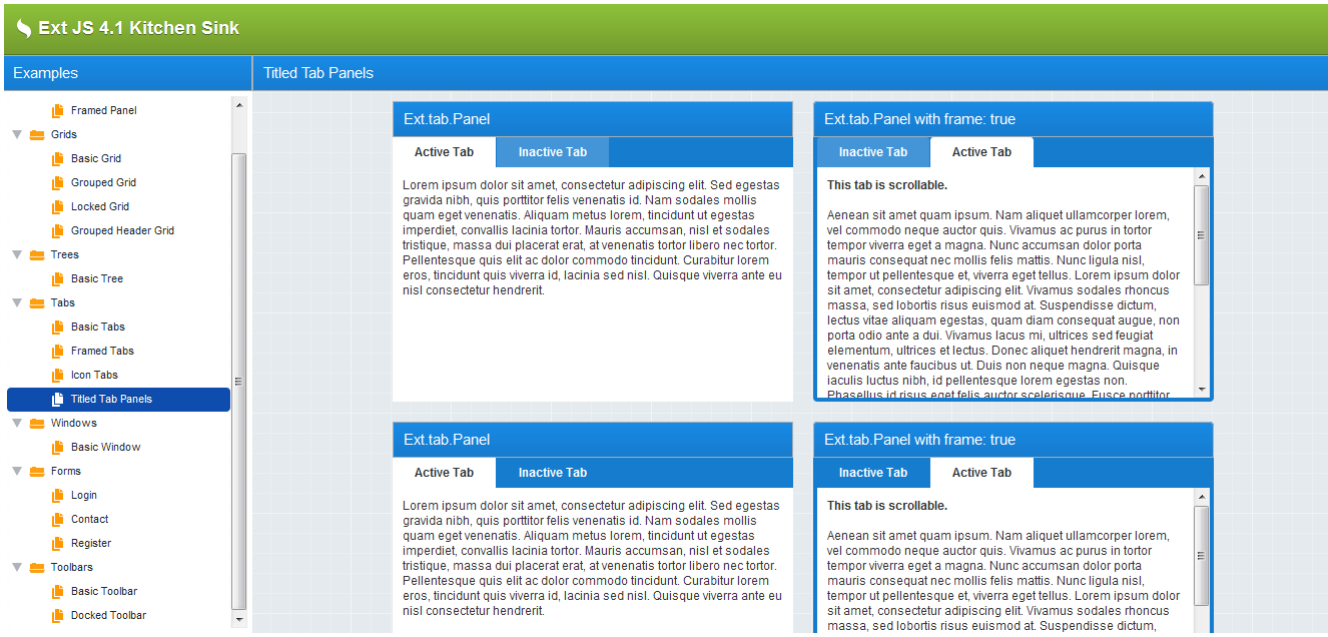

Figure 2.11: An example that created by sencha

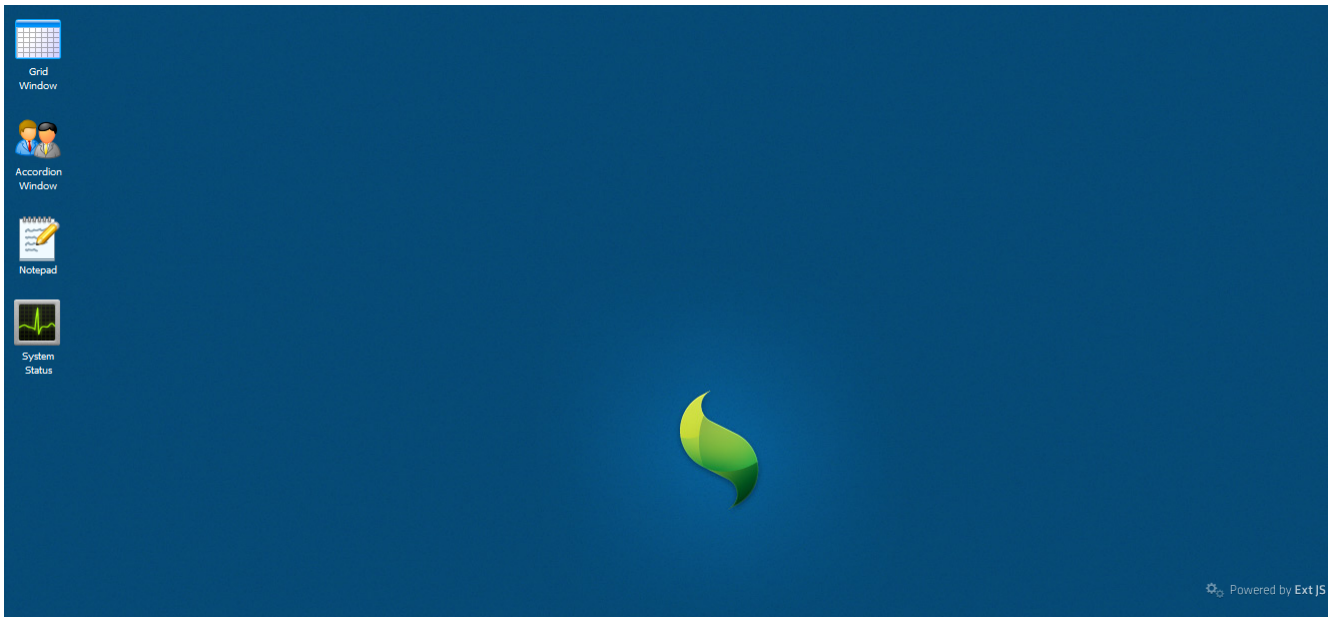

Figure 2.12: Another example that created by sencha

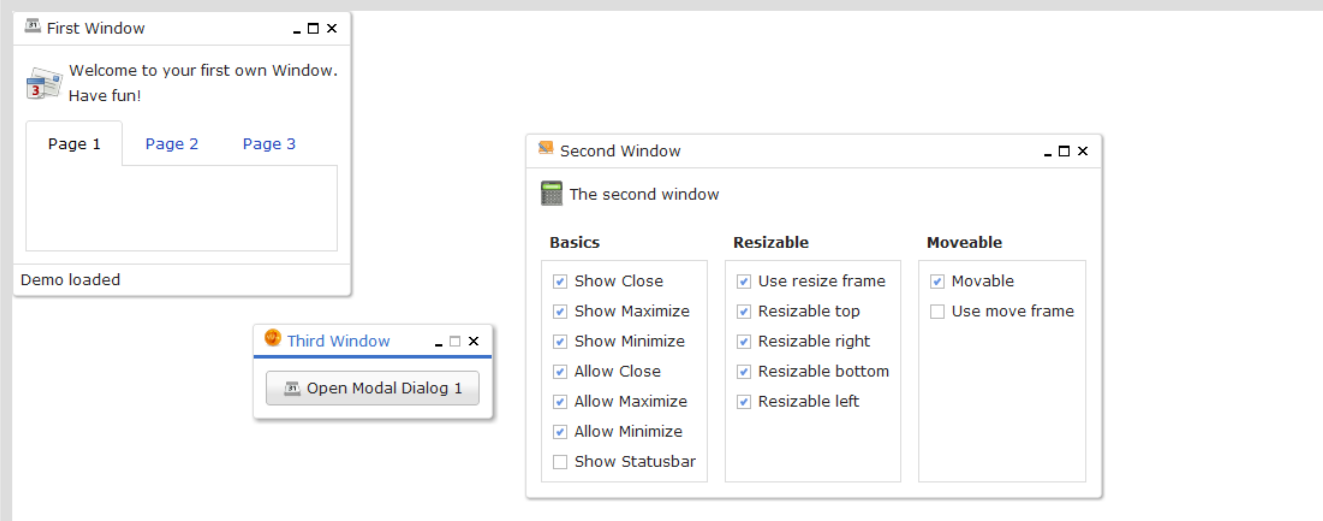

Figure 2.13: An example that created by qooxdoo

  - feature supports Browser abstraction, DOM manipulation, Events, Templating, Animation.

- Cons:
  - non CSS -based styling

JQueryUI

- Overview: a curated set of user interface interactions, effects, widgets, and themes built on top of the jQuery JavaScript Library. Whether you're building highly interactive web applications or you just need to add a date picker to a form control, jQuery UI is the perfect choice.
- Pros:
  - Base on one of the most popular framework now.
  - Stable
  - Nice Theme
  - support interaction
  - MIT license
  - One of the nicest things from jQuery.UI I think is the widget factory, which gives you a quick way of creating your own plug-ins.

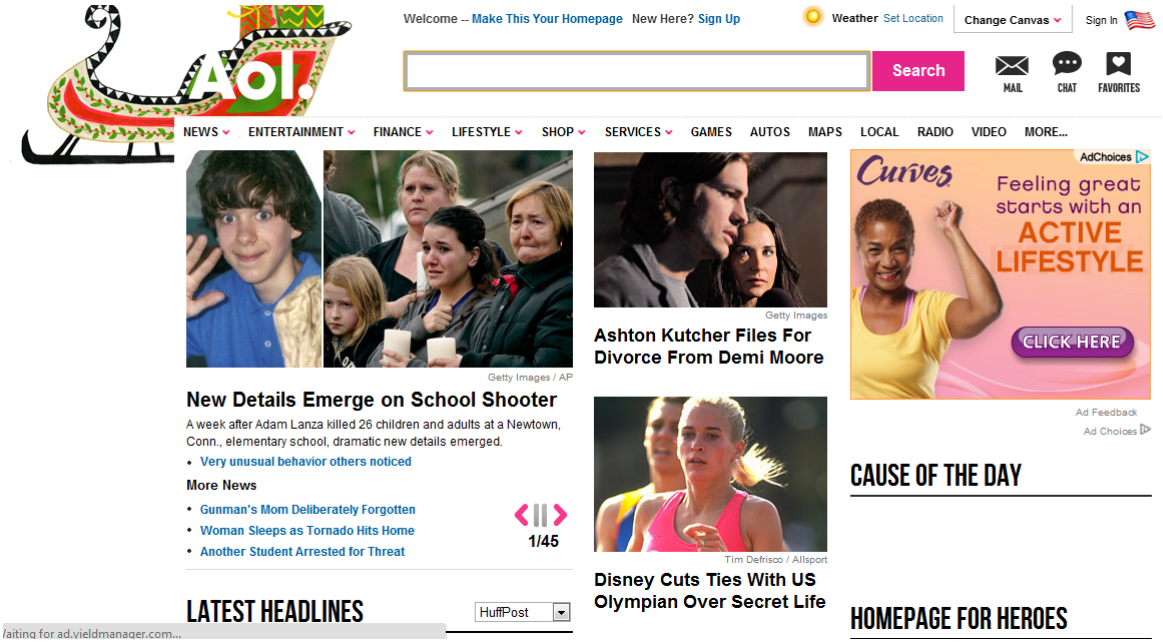

Figure 2.14: An example that created by jqueryui

  - is extremely easy to use. Built-in functions are very comprehensive. Compatibility is good.

- Cons:

  - too heavy
  - javaScript 's cons

YUI

- Overview: an open-source JavaScript library for building richly interactive web applications using techniques such as Ajax, DHTML and DOM scripting.

- Pros:

  - support models, views and routers and make it simple to write multi-view applications supporting routing, View transitions and more.
  - it is a complete solution that includes widgets/components as well as the tools needed to create an organized application architecture.

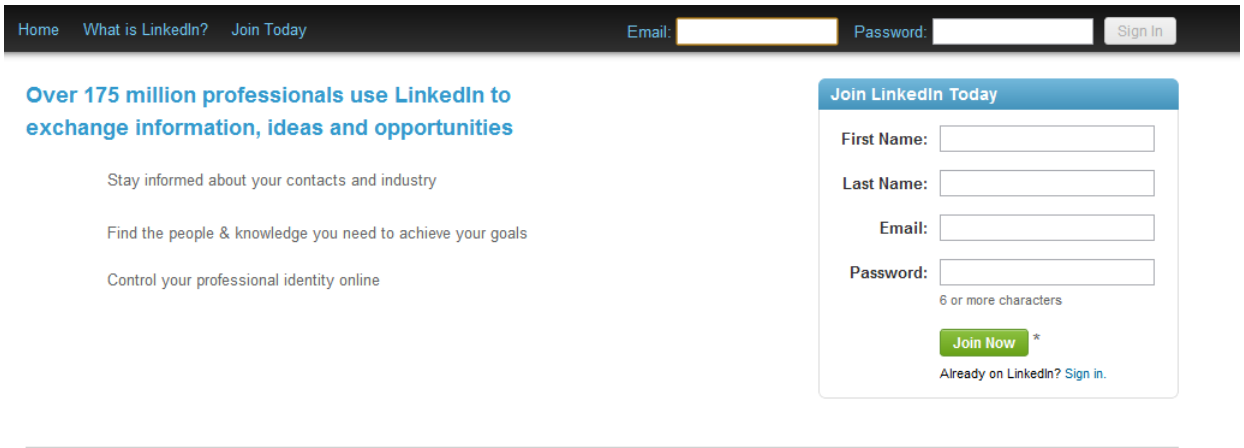

Figure 2.15: An example that created by YUI

  - have scaffolding tools (yuiproject), but these need to be updated
  - includes all of the goodies of Backbone

- Cons:

  - it should support some of the auto-wiring (optional) of Ember
  - should have more AMD-compatible module loader"

Backbone.js

- Overview: Backbone.js gives structure to web applications by providing models with key-value binding and custom events, collections with a rich API of enumerable functions, views with declarative event handling, and connects it all to your existing API over a RESTful JSON interface.

- Pros:

  - support a persistence layer and RESTful sync, models, views (with controllers), event-driven communication, templating and routing.
  - suitable to build non-trivial applications
  - quickly depart from one another in how they expect you to approach building apps.

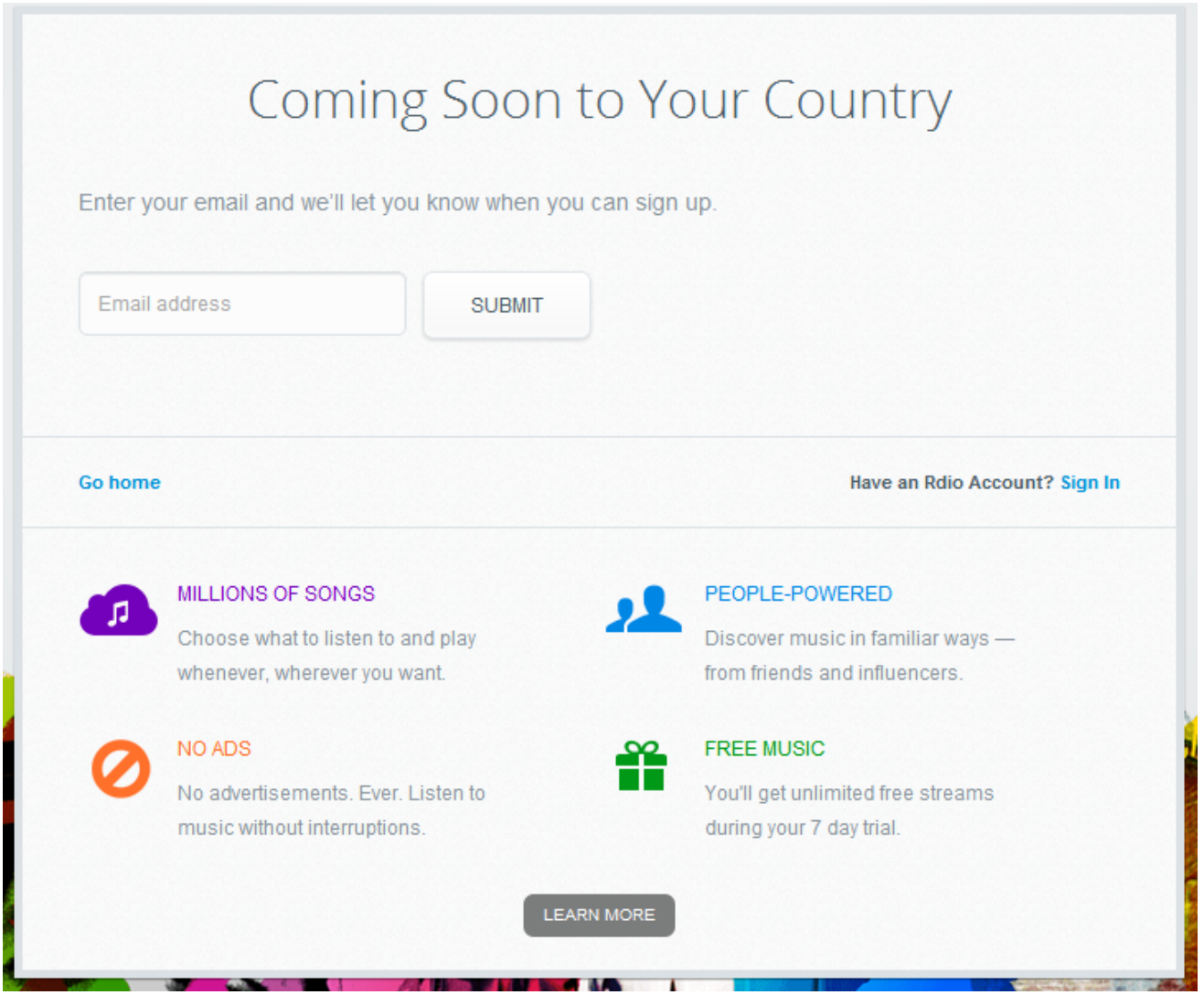

Figure 2.16: An example that created by backbone

  - more suitable with something flexible which offers a minimalist solution to separating concerns in my application
  - you want control, and compatibility with other frameworks.

- Cons:
  - A bit clunky, always having to create both the event trigger and event listener in all all view, model, router etc, which makes for larger code and longer figuring it out writing it.

DHTMLX

- Overview:DHTMLX Touch is an HTML5-based JavaScript library for building mobile web applications. Its not just a set of UI widgets, but a complete framework that allows you to create eye-catching, cross-platform web applications for mobile and touchscreen devices.
- Pros:
  - Great features and UI components.
  - Great tutorials and samples.
  - Doesn't conflict with well-known AJAX framework like: JQuery, YUI,..
  - The library works in all modern browsers:
- Cons:
  -

Rialto

- Overview: Rialto (Rich Internet Application Toolkit) is a cross browser ajax based JavaScript widgets library. Because it is technology agnostic it can be encapsulated in JSP, JSF, Python, .Net or PHP graphic components.
- Pros:
  - is designed for SPA
  - Widgets library includes: forms, dragdrop, tree, data list with fix header and resizable columns, pop up, splitter.
- Cons:
  - The documents and demos seem to be vague
  - the community is not so active

Figure 2.17: An example that created by DHTMLX

The Tables below provide an overview of available MV* frameworks. Columns in the table show the different frameworks, while the rows classify them with respect to the classification dimensions introduced .The character ”x” equals to ”yes” and ”-” equals to ”no”. Furthermore additional information for every framework is provided: These following tables show more information about those JavaScript frameworks. The tables also reveal the uniform distribution of MVC and MVVM frameworks as well as JavaScript being the predominant programming language. Furthermore most of the frameworks require no special integration process to be used within an application. Usually it is enough to include a single JavaScript file into the applications source code.

2.2.4 Selected Framework

Angular.js

Angular.js is a JavaScript framework released under the open source MIT license. The implementation is based on JavaScript and has a very small footprint (78 kB minified). Since Angular.js has no dependencies it can be used in conjunction with any other JavaScript library. For developers a very comprehensive set of documentation is available comprising an API specification, interactive tutorials as well as running examples. Moreover, Angular.js is special JavaScript framework which is built particularly for software engineering purpose.Angular.js lets you extend HTML vocabulary for your application. The resulting environment is extraordinarily expressive, readable, and quick to develop.Besides, Angular.js is designed from ground up to be testable. It encourages behavior-view separation, comes pre-bundled with mocks, and takes full advantage of dependency injection. It also comes with end-to-end scenario runner which eliminates test flakiness by understanding the inner workings of AngularJS.

2.3 JavaScript Graph Libraries Selection

2.3.1 Overview of some javaScript Graph libraries

There are several graph libraries providing APIs for implementing and managing various kinds of diagram. Among them , there are some names that have been used a lot like Joinjs, mxgraph,Yfiles, Draw2D. While some of

	Comparison of Javascript framework					
	Google Web toolkit	**Vaadin**	**Sproutcore**	**Dojo**	**JavaScriptMVC**	**jQuery**
overview	- provides an open source set of tools that allows web developers to create and maintain complex JavaScript front-end applications in Java - write both the client-side code and the server-side code in Java	- features a server-side architecture - client-side is built on top of GWT	an open-source framework for building blazingly fast, innovative user experiences on the web supported by Apple. It is also one of the largest frameworks.	DOJO is one of the leading JavaScript framework.	- an open-source framework containing the best ideas in jQuery development. - A collection of the best practices and tools for building JavaScript applications. Built on top of jQuery	Query is free, open source software, licensed under the MIT License
Pros	- supports ability to debug - Strong community - good documents,demos, samples - supports lots of features - No JavaScript syntax errors	- Wide variety of UI components - Drag and Drop for Tables, Panels and Trees components have real nice demos and tutorial - Simple to learn, it has a	- MIT license - Bindings support. - Solid community. - Tons of features.	- documents and tutorials in details - has all the components you would most likely	- using Controllers can make a clean, tight code that is easy to find -is very lightweight, it's jQuery based	-Easy to use - Large library - Strong community - Great documents
Cons	- kind of a little too few stunning UI components comparing with Vaadin - don't see much demo or support about drag & drop	- for advanced customization you'll need to write more in CSS, HTML, JavaScript - Client side is not extensible and very chaotic - html files is really heavy	- The tutorial is nice - don't see much animation features	- API stability - Demo is not very clear and scatter	- No preset UI layer implementations	- Functionality maybe limited - JQuery javascript file required
source Language	java - for both GWT and Vaadin, all you have do with is java, it's good for people who don't like HTML, CSS, Javascript	java	javaScript	javaScript	JavaScript	JavaScript
Community	strong community with thounsands of topic discussed and updated on	Strong community,	Strong community on some social networks, blogs	they have a forum to discuss	quite a solid community since it's based jQuery	Very strong
features	- Supports lots of features include visual animations - Drag & Drop is not a basic feature , need to install plugins	-support a lot of features like GWT, moreover it has many UI components	- don't see much animation features and drag&drop - not too much UI components	- provide large range and almost every things you need	- almost all features needed to build a web app	- Almost all features - include JQuery UI supports a
documents	has good tutorials for features in details	good documents, tutorials, and demos	-nice tutorial and demos with source code	-in details but incomplete and scatter	-nice tutorial	great tutorial
Some opinions from the community(*)	link: http://www.javacodegeeks.com/2012/01/gwt-pros-and-cons.html - it's good for those who had java background and doesn't really appreciate the power of JavaScript (for both GWT and	link: http://tc-innernerd.blogspot.com/2011/09/vaadin-pros-and-cons.html - With Vaadin, you can build a nice app with almost a half of lines of code and still the	- Overly prescriptive. - Hard to decouple from unneeded features. - Forces a native-like paradigm	- Many have commented that Dojo seems difficult to learn and get started with	- Lightweight - Fast performance - easy to control	one of the most popular which means one of the best libraries of JavaScript

Table 2.1: Some of JavaScript frameworks in the survey (first six ones)

them have just been introduced like Gojs, Raphael, d3js. They are all remarkable libraries which provide stunning UI and animations. It would be hard to choose one of them if there are not criteria a to compare. Therefore, the some criteria will be presented in the next section.

2.3.2 JavaScript Graph Libraries Selection Characteristics

These are some criteria that will be based on to choose graph library

- Language: javaScript

- Feature supports various types of shapes
- Drag & drop support
- data binding support
- Light weight and easy to integrate
- Nice designed documents and lots of samples
- The license under which the framework is released
- The size of the library
- Community: the number of users or posts in the forum they use or on GitHub(Jan 2013) indicating the community support, as well as the website of the framework usually providing tutorials and a more or less comprehensive documentation.

2.3.3 Survey of Existing graph libraries

Conducting a survey about JavaScript graph libraries is an inevitable step before choosing one library for implementation. This survey provides an overview of what these libraries are capable of and base on that a library will be chosen with the features fit the most with the proposed criteria.

JoinJS

JointJS is an open source modern JavaScript library for creating diagrams. It can be used to create either static diagrams or, and more importantly, fully interactive diagramming tools and application builders.

mxGraph

mxGraph is an interactive JavaScript HTML 5 diagramming library, with full fallback support for IE 6-8. mxGraph is simple, you include it as a JavaScript link in your HTML file and you instantly have access to the cleanest, most functional native browser diagramming component available. Draw.io[1] is a great demo built by Google inc using mxGraph library.

[1] https://www.draw.io/

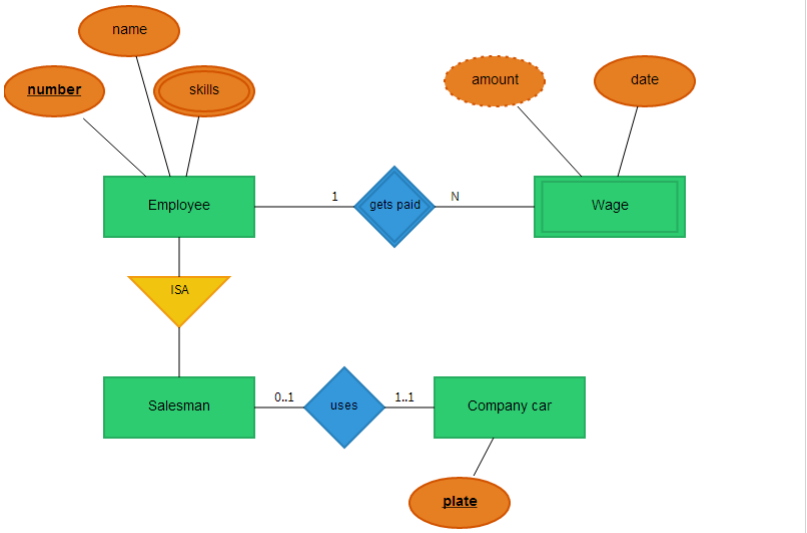

Figure 2.18: A sample created by JoinJS

YFiles

yFiles for HTML brings the proven power and ease of yFiles diagramming to your cutting-edge HTML5 applications. yFiles for HTML has almost all functionality known from our diagramming libraries for the .NET platform. It contains UI controls for viewing and editing diagrams and our layout algorithms for automatically arranging complex graphs and networks at the click of a button.

Canviz

Canviz is a JavaScript library for drawing Graphviz graphs to a web browser canvas. More technically, Canviz is a JavaScript xdot renderer. It works in most modern browsers.

Draw2D

Create drawings, diagrams or an workflow editor with the Javascript library. The User interface allows interactive drawing by using your standard browser.

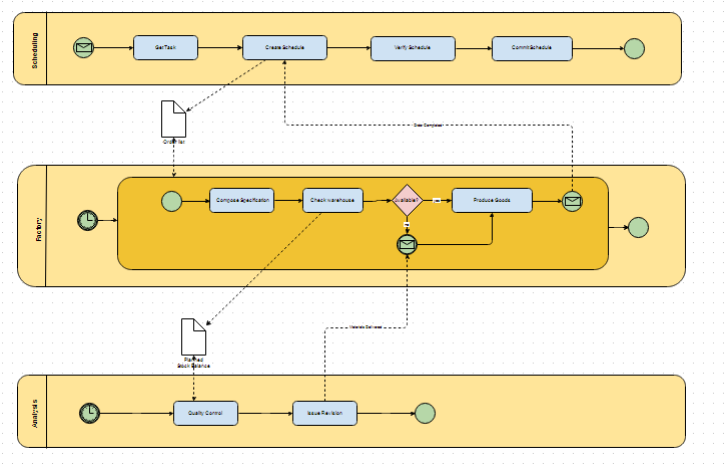

Figure 2.19: A sample created by mxgraph

Data-driven documents

D3.js is a JavaScript library for manipulating documents based on data. D3 helps you bring data to life using HTML, SVG and CSS. D3s emphasis on web standards gives you the full capabilities of modern browsers without tying yourself to a proprietary framework, combining powerful visualization components and a data-driven approach to DOM manipulation.

Gojs

GoJS is a feature-rich JavaScript library for implementing interactive diagrams across modern browsers and platforms. GoJS makes constructing diagrams of complex Nodes, Links, and Groups easy with customizable templates and layouts.

mindfusion

The library is written 100% in JavaScript and uses the HTML5 Canvas element for drawing. JsDiagram depends on the Microsoft Ajax library for type system implementation and browser independence. With JSDiagram users can create, resize, select, move and modify nodes and links as they wish. You can allow or permit any action or customize how the control response to a certain action - like pressing the 'Del' key for example.

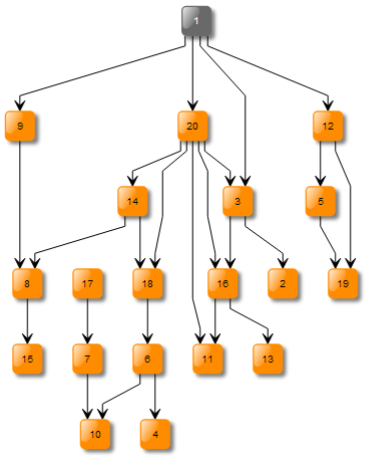

Figure 2.20: A sample created by yFiles

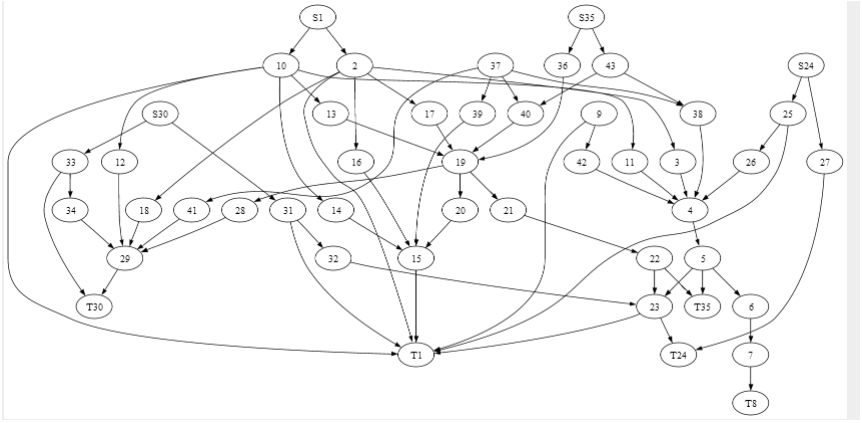

Figure 2.21: A sample created by canviz

Raphael

Raphal is a small JavaScript library that should simplify your work with vector graphics on the web. If you want to create your own specific chart or image crop and rotate widget, for example, you can achieve it simply and easily with this library.

JavaScript InfoVis Toolkit

The JavaScript InfoVis Toolkit provides tools for creating Interactive Data Visualizations for the Web.

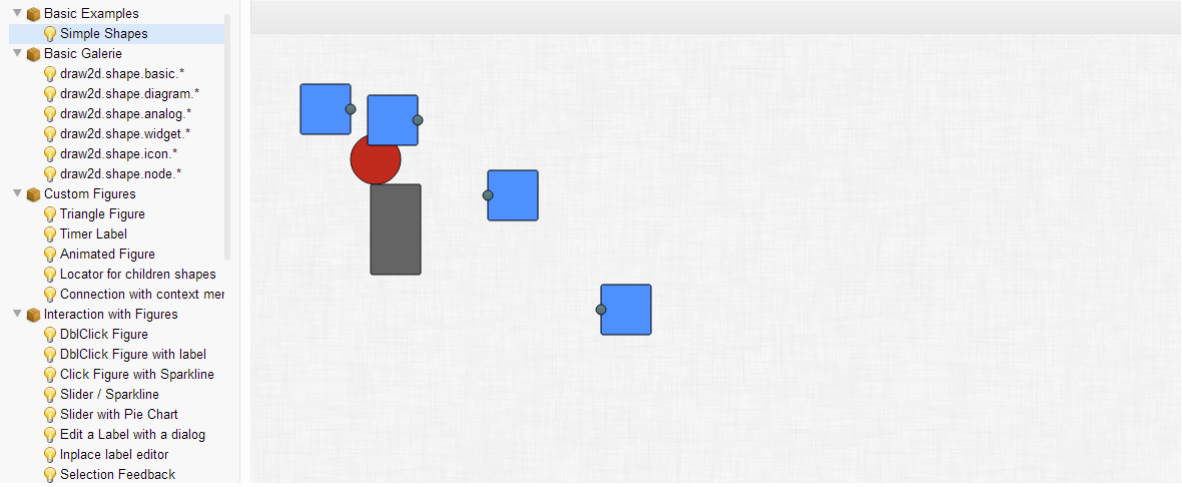

Figure 2.22: A sample created by draw2D

jsUML2

jsUMLT2 is a lightweight HTML5/javascript library for UML 2 diagramming. It allows the developer to easily embed UML 2 diagram in web applications, just invoking a few javascript methods.

JsPlumb

jsPlumb allows you to connect elements on the screen using SVG, Canvas or VML, depending on the capabilities of the browser. go with jQuery/MooTools/YUI3.

Nodeviz

NodeViz includes PHP code for managing assembling relational data on a server, formatting it for Graphviz network layout, sending SVG or JPG versions of the network to client, JavaScript classes for embedding network graph image and associated lists in client web page, as well as methods and events for zooming, panning, highlighting and other interactions.

The two following table 2.7 and table 2.8 show the information of all JavaScript graph libraries which were surveyed.

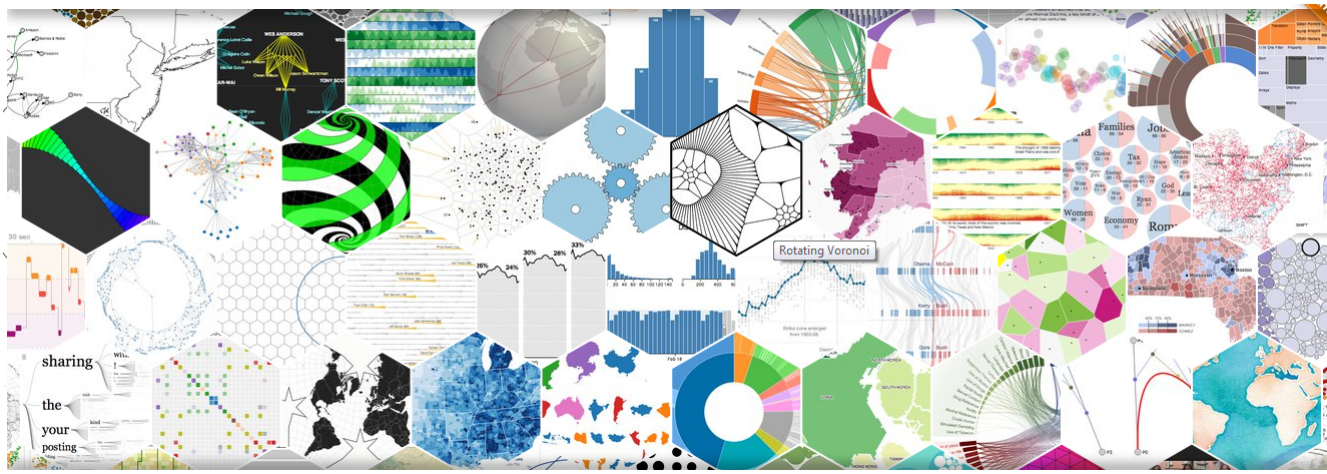

Figure 2.23: A sample created by D3js

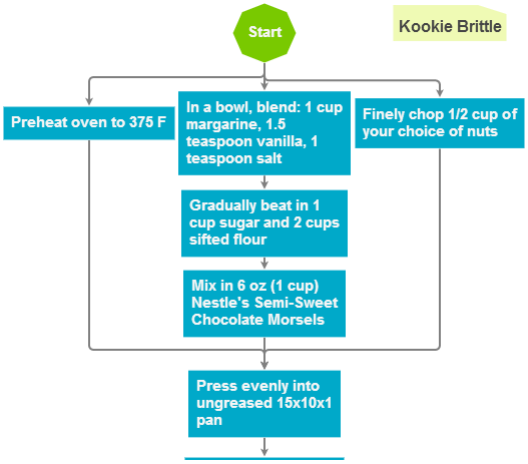

Figure 2.24: A sample created by Gojs

2.3.4 Selected graph library

Gojs

Gojs is a fast and powerful JavaScript library for implementing interactive diagrams in HTML5 web applications. it is released under nwoods company license. The reason Gojs is chosen it that this library is a new product and providing such beautiful graphics and various kind of sample shapes. The implementation is based on JavaScript and has a small footprint (659KB (free trial)). Since Gojs has no dependencies it can be integrated with Angular.js. For developers there is an in-details introduction, a lot of sample code and

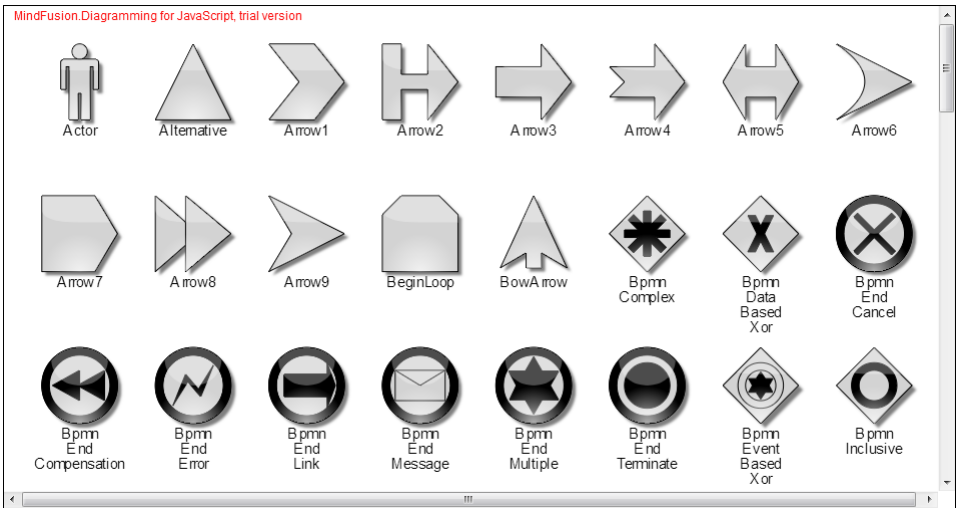

Figure 2.25: A sample created by Mindfusion

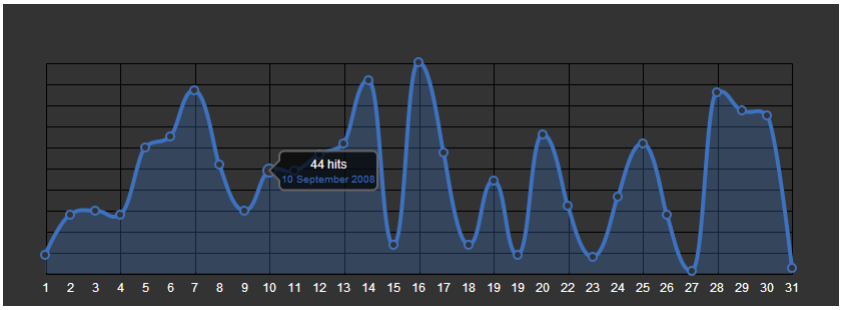

Figure 2.26: A sample created by Raphael

demos. Besides, there is a forum for developers to ask question and it is well-replied by the admins of the site.

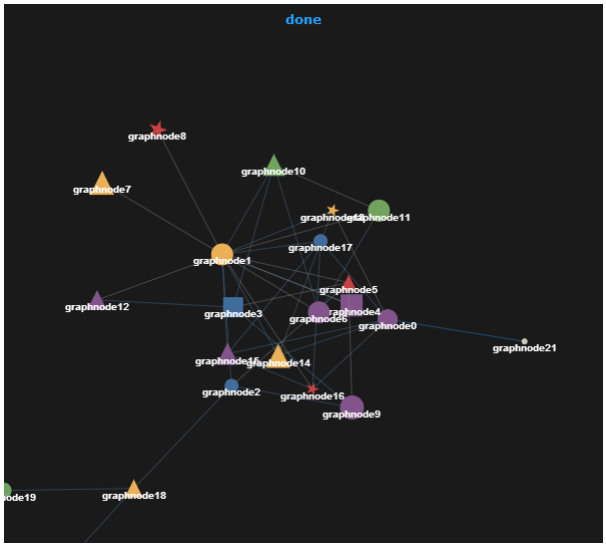

Figure 2.27: A sample created by JavaScript InfoVis Toolkit

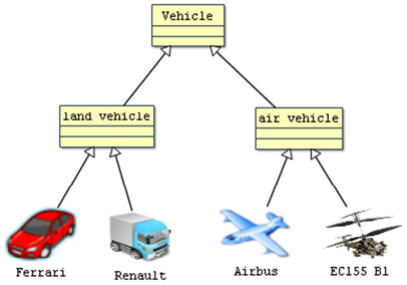

Figure 2.28: A sample created by jsUML2

	Cappuccino	ember.js	Flame.js	Angular.js	Twitter Bootstrap	Ext JS/Sencha
overview	an open source framework that makes it easy to build desktop-caliber applications that run in a web browser which look and feel like desktop applications on Mac OS X	- Ember is a JavaScript framework for creating ambitious web applications that eliminates boilerplate and provides a standard application architecture. - Same company with Sproutcore	a widget/UI library for Ember.js	an open-source JavaScript framework	is a free collection of tools for creating websites and web applications. It contains HTML and CSS-based design templates for typography, forms, buttons, charts, navigation and other interface components, as well as optional JavaScript extensions.	a pure JavaScript application framework for building interactive web applications using techniques such as Ajax, DHTML and DOM scripting.
Pros	- allow you to create true desktop-like apps right inside the browser - don't rely on a continous web connection - as fast as desktop app - you can build asyncronous, offline, robust web apps right inside the browser - Cappuccino is compatible with many of the latest browsers	- easily implementing MVC functionality. - extremely easy to create computed properties in JavaScript		- You don't have to write all the event listening and event triggering. Its automatic. -if you want something declarative that uses the View to derive behavior - focuses on achieving this through custom HTML tags	- The framework can be adapted to any CMS or blogging platform like WordPress, Drupal or Joomla. - Easy visual consistency in your application. - This framework is designed for a web application (not a website) and all the elements fit nicely together to get the app done fast. - many beautiful theme	- is like a superset of the widgets like simple label, textbox buttons to complex grids, drag-drop panel - It has quite good documentation with tutorials, samples and user community. - Active and currently most adopted javascript RIA framework - Good code quality/readability - Ext JS makes it simple to edit tickets inline
Cons	- Adding a layer of abstraction (Objective-J objects to Javascript) also adds to the overhead. The result is that the application can be slow, particularly if the user's computer is slow. - It is not designed to make full-fledged web sites => Complex and interactive sites may not suitable - is still very early in its development stages, so some parts are still unimplemented - Different Underlying Model	- i don't see complaints just document is not quite clear		- invents its own syntax which requires learning - Obtrusively mixes all controller, model and view into the html - Javascript errors happen if you DON'T clutter the window object.	- Harder to be original	-Loading time would is high for home page on web. - CSS – very easy to get lost. It is difficult to find correct class names - HTML – full of divs and overly complex generated code. Difficult to debug even with FireBug. - Customization is not easily achievable. - Loading even simple things requires few lines of coding which is simpler in plain html or jQuery - Need quite experienced developer
source Language	Objective-J(a superset of Javascript)	JavaScript	JavaScript	JavaScript	HTML,CSS,JavaScript	JavaScript, CSS
Community	pretty strong community	strong community on stackoverflow		- they are on twitter and Google+ but i find the best community is on stackoverflow - i found a lot of comment for this	there is a community on twitter of course, and lots of developers discuss on stackoverflow or other sites	quite strong community
features	many features	good features		quite lots of great features for specific	- incomplete support for HTML5 and CSS 3, but it is	has so many great components
documents	bad document	not good		- nice page tutorial	-nice doc	Good API documentation.
Some opinions from the community( )	- doesn't require you to know about HTML, CSS, or even the DOM - feel is the same as that of Cocoa programs - It works well for focused, single-purpose applications (like photo-editing, document editing, etc) - focused on more modularized code and so called "semantic-templates" for views	- is designed so you don't have to worry about whether or not you have 2000 bindings - is intended for "web-styled" applications.		- the more bound elements in your app, the slower it gets - Difficult to allow browsers to optimize this in native code. - great for small- to intermediate-scale applications - It only provides an interface between a combined M, V and C. What if you wanted a Model to subscribe to another Model but didn't want to update a view. You can't do this with Angular.js, but you can with Backbone	-It uses Less css. but many people like to use other tool to mangage css - Speed -Consistency across applications -People find it simple and elegant	- Nice widgets - Active and currently most adopted javascript RIA framework - Good code quality/readability - It is not possible for the user to bookmark a certain page . Since the objects are rendered by DOM manipulation, page can not be indexed by search engines

Table 2.2: Some of JavaScript frameworks in the survey (second six ones)

	quooxdoo	jQueryUI	YUI	BackboneJS	DHTMLX	Rialto
overview	ooxdoo is a universal JavaScript framework with a coherent set of individual components and a powerful toolchain	a curated set of user interface interactions, effects, widgets, and themes built on top of the jQuery JavaScript Library. Whether you're building highly interactive web applications or you just need to add a date picker to a form control, jQuery UI is the perfect choice.	an open-source JavaScript library for building richly interactive web applications using techniques such as Ajax, DHTML and DOM scripting	Backbone.js gives structure to web applications by providing models with key-value binding and custom events, collections with a rich API of enumerable functions, views with declarative event handling, and connects it all to your existing API over a RESTful JSON interface.	DHTMLX Touch is an HTML5-based JavaScript library for building mobile web applications. It's not just a set of UI widgets, but a complete framework that allows you to create eye-catching, cross-platform web applications for mobile and touchscreen devices.	Rialto (Rich Internet Application Toolkit) is a cross browser ajax based JavaScript widgets library. Because it is technology agnostic it can be encapsulated in JSP, JSF, Python, .Net or PHP graphic components.
Pros	- namespace - eventbiding - cross-browser back button support - bookmarkability - AOP	- One of the most popular framework now - Stable - Nice Theme - support interaction - MIT license	- support models, views and routers and make it simple to write multi-view applications supporting routing, View transitions and more. - it is a complete solution that includes widgets/components as well as the tools needed to create an organized application architecture. - have scaffolding tools (yuiproject), but these need to be updated - includes all of the goodies of Backbone	- support a persistence layer and RESTful sync, models, views (with controllers), event-driven communication, templating and routing. suitable to build non-trivial applications		- provides easy and fast interface creation - chieved by powerful tags which display panels, input fields, tables, treeviews, sortable lists, datagrids, popups, calendars, etc. With those tags, the developer have neither need to write nor even know HTML.|...
Cons	- non CSS - based styling	- too heavy - javaScript 's cons	-it should support some of the auto-wiring (optional) of Ember -should have more AMD-compatible module loader	A bit clunky, always having to create both the event trigger and event listener in all all view, model, router etc, which makes for larger code and longer figuring it out writing it.		
source Language	JavaScript	JavaScript	JavaScript	JavaScript	JavaScript	JavaScript
Community	Lack of overall community and following	strong community	Strong community	- there to be an active extension community around the framework that have already tried addressing larger problems (Marionette, Chaplin, Aura, Thorax) - it has a group, a wiki page, github source	not strong	seem not a strong community
features	- Browser abstraction	tons of widgets, effects, utilities	A lot of great features	many great features	various of features	many features
documents	Lack of overall community and following	good document, with details and tutorial	nice document	- there are also scaffolding tools (grunt-bbb, brunch) available for the solution - there are tutorial from beginner to advance	nice demo but the document seem not good for me	nice document
Some opinions from the community(*)	Commitment from their core development team (defects are slow to be fixed).	-One of the nicest things from jQuery.UI I think is the widget factory, which gives you a quick way of creating your own plug-ins. - is extremely easy to use. Built-in functions are very comprehensive. Compatibility is good	- The loader today works very well; however, it would be nicer if I could start a new projects based on AMD modules and pull in certain YUI3 components and other things from other places that are also using AMD.	- quickly depart from one another in how they expect you to approach building apps. - more suitable with something flexible which offers a minimalist solution to separating concerns in my application - you want control, and compatibility with other frameworks	cannot find much	cannot find much

Table 2.3: Some of JavaScript frameworks in the survey(third six ones)

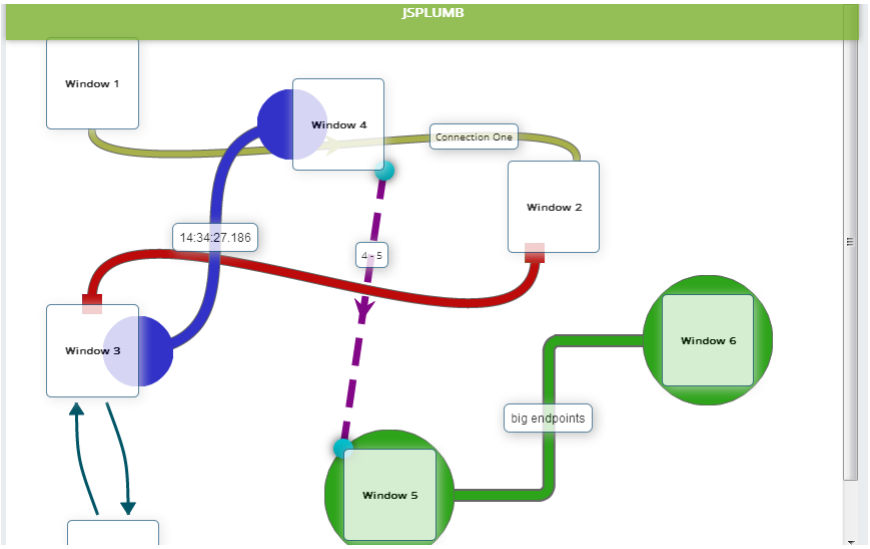

Figure 2.29: A sample created by jsPlumb

	Google Web toolkit	Vaadin	Sproutcore	Dojo	JavaScriptMVC	jQuery
Programming Language	java	java	javaScript	JavaScript + HTML	JavaScript	JavaScript
JS include	-	-	-	x	x	x
Data Binding	-	-	x	x	x	x
Basic UI components support	x	x	x	x	-	-
Advanced UI widget library (grid, tree ,..)	x	x	x	x	x	x
Validation support	x	x	-	x	x	x
Drag&Drop	x	x	x	x	x	x
License	Apache	Apache	MIT	BSD & AFL	MIT	MIT
Size	105 MB (SDK ver 2.5)	4.8MB(ver 6.8.7)	23.5MB(ver 1.9.1)	41KB (ver 1.8.3)	31.19MB (ver 3.2.4)	32KB (ver 1.8.3)
Community	- 29104 members 40742 topics (discussion group) - 1896 members 18149 topics (contributor group)	46262 posts 4,977 members (forum) 144 posts(blog) 155 posts(wiki)	253 (fork on Github)	6061 users (forum)	2809 Posts 9623 Responses (forum)	110854 posts 241350 Responses (forum)
Website	https://developers.google.com/web-toolkit/examples/ https://groups.google.com/forum/?fromgroups#!aboutgroup/google-web-toolkit-contributors	https://vaadin.com/home	http://sproutcore.com/	http://dojotoolkit.org/	http://javascriptmvc.com/index.html	http://jquery.com/

Table 2.4: First six JavaScript frameworks surveyed with new criteria

	Cappuccino	ember.js	Flame.js	Angular.js	Twitter Bootstrap	Ext JS/Sencha
Programming Language	Objective-J(a superset of Javascript)	JavaScript	JavaScript	JavaScript	HTML,CSS,JavaScript	JavaScript, CSS
JS include	x	x	x	x	x	x
Data Binding	x	x	-	x	x	x
Basic UI components support	x	-	-	-	x	x
Advanced UI widget library (grid, tree ,..)	x	-	x	-	x	x
Validation support	-	-		-	x (jquery)	x
Drag&Drop	x	x	x	x	x (plugin)	x
License	LGPL	MIT		MIT	Apache	Commercial & GPL v3
Size	(ver 0.9.5)	40k (ver 1.0.0)		78KB (ver 1.0.3)	83.32KB (ver 2.2.2)	44.66MB(ver 4.1.1a)
Community	268 forks (github)	862 forks(github)	29 fork (github)	900 forks(github)	11434 forks (github) ~47K followers(twitter)	436802 members 845,307 posts(forum)
Website	http://cappuccino.org/	http://emberjs.com/	https://github.com/flamejs	http://angularjs.org/#/	http://twitter.github.com/bootstrap/index.html	http://www.sencha.com/products/extjs

Table 2.5: Second six JavaScript frameworks surveyed with new criteria

	quooxdoo	jQueryUI	YUI	BackboneJS	DHTMLX	Rialto
Programming Language	JavaScript	JavaScript	JavaScript	JavaScript	JavaScript	JavaScript
JS include	x	x	x	x	x	-
Data Binding	x	x	x	-	x	
Basic UI components support	x	x	x	-	x	x
Advanced UI widget library (grid, tree ,..)	x	x	x	-	x	x
Validation support	x	x	x	-	x	x (not much)
Drag&Drop	x	x	x	-	x	x
License	LGPL & EPL	MIT	BSD	MIT	GPL & Commercial	Apache
Size	205.51KB (ver 2.1)	1.66MB(ver 1.9.2)	26.81MB (Ver 3.8.0)	56kb (Ver 0.9.9)	35.15MB(v.3.5 build 120731)	313.9 kB (ver 2.1.5)
Community		as JQuery forum	12086 members 33344 posts (forum)	2082 (fork on github)	13609 members 85773 posts(forum)	
Website	http://qooxdoo.org/	http://jqueryui.com/	http://developer.yahoo.com/yui/examples/ http://yuilibrary.com/	http://backbonejs.org/	http://www.dhtmlx.com/index.shtml	https://rialto.improve-technologies.com/redmine/projects/rialto-gwt http://www.improve-foundations.org:8080/Rialto-GWT-Demo/demo.html https://groups.google.com/forum/?fromgroups#!forum/rialto-gwt

Table 2.6: First six JavaScript frameworks surveyed with new criteria

	Graph and Visualization Libraries						
	JointJS	mxGraph	YFiles	CanViz	Draw2D	Data-driven	gojs
Overview	JointJS is a JavaScript library for creating diagrams. The diagrams can be fully interactive	mxGraph is an interactive JavaScript HTML 5 diagramming library, with full fallback support for IE 6-8	YFiles offers fully interactive diagram viewing and editing support accross flatform	Canviz is a JavaScript library for drawing Graphviz graphs to a web browser canvas	Create drawings, diagrams or an workflow editor with the Javascript library.The User interface allows interactive drawing by using your standard browser.	a JavaScript library for manipulating documents based on data	a fast and powerful JavaScript library for implementing interactive diagrams in HTML5 web applications.
Performance	-fast and very smooth	Quite fast and smooth	- Not quite fast , need time to load when moving -	- doesn't support interactive	- Fast and smooth	vividly	very smooth
Ease of use	- Still have troubles interaction with the lines connecting elements - Quite easy to use	easy to use except the interaction with the lines connecting elements	- difficult to interact with lines and rearrange them	- doesn't support interactive	- seem easy but still have trouble interact with the lines - movements with the lines go along => easy to arrange	just graphs and charts with stunning elements and vivid animation => the user don't have to do much with these things	support arrangement pretty good
Graphics & Features	- Whole graph is in a canvas - don't have too many defined elements	-nice elements	- Nice colors - supports great auto arrangement and grouping features	- can draw various type of diagram but not support interaction is a huge disadvantage	- supports lots of components - seem to have only basic shape not the defined diagrams like UML, ERD	- real nice graphics , but the component seem not the diagram type (like UML, ERD,..) and more like charts, calendars, graphs	-seem very powerful -various components and types, especially supports drawing ERD and some other kind of diagrams -grouping features, undo/redo/labeling/ auto layout
Document	- good examples	- lots of example for JavaScirpt, PHP and .NET	- not very comprehensible and easy to read but in details	on a wiki page. not much		- there are lots of example with sample code, an api-reference, plugins	have a page for samples and an API-reference page in details and a nice document with great examples
License	MIT	JGraph Ltd	yWorks (only free for 30 days for evaluation)	MIT	4 euros (trial) 499 euros (commercial) vendor: Andreas Herz	BSD license	Northwoods Software®, unlimited time trial for development
Size	6MB	12.27MB(ver 1.10.4.1)		988 KB (ver 0.1)		581 KB	659KB (free trial)
Community	39 forks (github) 137 membes (google group)	58 fork (github)		98 members (google group)		1181 forks (github)	4185 members (forum), very good and fast support
Website	http://www.jointjs.com	https://github.com/jgraph/mxgraph	http://www.yworks.com/en/products_yfilesajax_about.html	http://code.google.com/p/canviz/	http://draw2d.org/draw2d_touch/jsdoc/#!/example/shape	https://github.com/mbostock/d3	http://www.nwoods.com/components/canvas/gojs-overview.htm
Easy to integrate	Yes (just need to include file)	Yes (just need to include files) and there is tutorial how to that too	yes	yes	yes	yes, include js file only	include 1 javascript file
Easy to program	seem easy since we only need to call the functions	not too difficult when we have a good document like this	not too hard	not too hard	not too hard	not too hard	a nice document like this
API support	supports create a canvas, drag, drop on that	supports different kind of graphs and diagrams, rearrangement, drag&drop	different kind of components	does not support interaction with the elements so i don't choose this	different kind of components	canvas, nice components and vivid animation	various components, canvas, drag&drop, tooltip, tree, ...

Table 2.7: Survey information of first sevens javascript graph libraries

	Graph and Visualization Libraries					
	mindfusion	**Raphael**	**JavaScript InfoVis**	**jsUML2**	**JsPlumb**	**Nodeviz**
Overview	JavaScript lets you create beautiful flowcharts that are customized and arranged in a few easy steps	a small JavaScript library supports create your own specific chart or image crop and rotate widget.	provides tools for creating Interactive Data Visualizations for the Web.	A lightweight HTML5/javascript library for UML 2 diagramming. It allows the developer to easily embed UML 2 diagram in web applications, just invoking a few javascript methods.	jsPlumb allows you to connect elements on the screen using SVG, Canvas or VML, depending on the capabilities of the browser. go with jQuery/MooTools/YUI3	It adds a PHP back-end onto some javascript front-end tools.
Performance	ok	kinda smooth	quite smooth, not very fast	quite fast and smooth	very smooth,fast	not smooth, not fast
Ease of use	not easy to use, kind of make the users confused	seem easy since it's not for complicated diagram	just for visualized purpose, seem easy and vivid	not very easy	quite easy to arrange	not very easy
Graphics & Features	not too stunning but support pretty many components for diagram-type	colorful components just for visualization	- nice components but support most of chart-type and tree-type and some other kinds , see no diagram example	- not very nice just ok - support only UML	-seems limited in features compared to others - nice components, graphics and animation, most of the features support diagram-type - don't support modify text on the diagram directly	limited features
Document	has a api-reference and lack of sample for each elements	nice documents with sample code and result	document in details with example code but no result respectively	a document in spanish in pdf file without any bookmark => difficult to navigate	nice and detailed document with some examples and an api-reference page	not good
License	MindFusion Ltd.	MIT License	SenchaLabs - Author: Nicolas Garcia Belmonte	GNU GPL v3	triple-licensed (JQuery - MooTools -YUI)	GNU GPL v3
Size	3.26MB (free trial version include docs and samples)	88.52KB (ver 2.1.0)	313KB (ver 2.0.1)	3.21MB	31.8MB rar	25MB rar
Community	2116 registered members.	doesn't have one	1403 members (google group)	doesn't have one	91 forks (github)	doesn't have one
Website	http://www.mindfusion.eu/download-jsdiagram.html	http://raphaeljs.com	http://philogb.github.com/jit/demos.html	http://code.google.com/p/jsuml2/	http://www.jsplumb.org/jquery/demo.html	https://code.google.com/p/nodeviz/
Easy to integrate	include a js file	Download and include raphael.js into HTML page	yes, just include into the page	need to set up a little complicated by using required tools such as: Sprocketize and YIU Compressor	Just include js files like using the above framework	Just include js files like using the above framework
Easy to program	ok, not too hard. But for further purpose , i have to reconsider the document	yes	mostly work with JSON and if familiar with it then the programming is ok	yes	yes, it inherits the syntax of these popular framework	yes
API support	some components and the interaction is not smooth and confusing	various elements generator, canvas, drag&drop but not much diagram type and text on it	canvas, drag, drop, interaction but not much for edit and rearrangement	canvas, drag,drop	canvas, drag&drop, dynamic anchors	canvas, drag,drop

Table 2.8: Survey information of the rest of graph libraries

Chapter 3

EXPLORATION OF KEY TECHNOLOGIES

In this chapter, I will present all the key technologies that i have been using in this thesis. For each technology, a brief overview will be introduced as well as their main features, advantages. Some important concept are also presented

3.1 AngularJS

3.1.1 overview

AngularJS is a structural framework for dynamic web apps. It lets you use HTML as your template language and lets you extend HTML's syntax to express your application's components clearly and succinctly. Out of the box, it eliminates much of the code you currently write through data binding and dependency injection. And it all happens in JavaScript within the browser, making it an ideal partner with any server technology.

Angular is what HTML would have been had it been designed for applications. HTML is a great declarative language for static documents. It does not contain much in the way of creating applications, and as a result building web applications is an exercise in what do I have to do to trick the browser into doing what I want.

The impedance mismatch between dynamic applications and static documents is often solved with:

- a library - a collection of functions which are useful when writing web apps. Your code is in charge and it calls into the library when it sees fit. E.g., jQuery.
- frameworks - a particular implementation of a web application, where your code fills in the details. The framework is in charge and it calls into your code when it needs something app specific. E.g., knockout, ember, etc.

Angular takes another approach. It attempts to minimize the impedance mismatch between document centric HTML and what an application needs by creating new HTML constructs. Angular teaches the browser new syntax through a construct we call directives. Examples include:

- Data binding, as in {{}}.
- DOM control structures for repeating/hiding DOM fragments.
- Support for forms and form validation.
- Attaching code-behind to DOM elements.
- Grouping of HTML into reusable components.

3.1.2 Why Angular?

DECLARATIVE HTML APPROACH

Declarative programming is when you say what you want, and imperative language is when you say how to get what you want.

Lets take an example:

```
# Declarative
small_nums = [x for x in range(20) if x < 5]

# Imperative
small_nums = []
for i in range(20):
 if i < 5:
 small_nums.append(i)
```

ANGULARJS S PHILOSOPHY

The AngularJS philosophy is based on embracing and extending HTML, while meeting the needs of both new and experienced developers. Moreover, Angular is built around the belief that declarative code is better than imperative when it comes to building UIs and wiring software components together, while imperative code is excellent for expressing business logic.

3.1.3 Dependency Injection

Introduction

There are only three ways an object or a function can get a hold of its dependencies:

1. The dependency can be created, typically using the new operator or asking a factory object to make one.

```
public SomeClass() {

 myObject = new Object(objectname);

}
```

```
public SomeClass() {

 myObject = Factory.getObject(objectname);

}
```

2. The dependency can be looked up by referring to a global variable.

```
public SomeClass() {

 myObject = GLOBAL_OBJECT;

}
```

3. The dependency can be passed in to where it is needed.

```
public SomeClass (Object myObject) {

 this.myObject = myObject;

}
```

The first two option are quite bad. Hardcoding the *objectname* does not really solve the problem as you cannot easily change your mind later without changing the that class again. Using a constant *GLOBAL_OBJECT* is also a bad idea as the *SomeClass* now depends on a constant to be set.

But there is yet another problem that cannot be solved easily: How can I change Object class? For instance, to replace it with a mock object to ease testing. This makes the class difficult to modify the dependencies. This is especially problematic in tests, again ,where it is often desirable to provide mock dependencies for test isolation[].

The third option , which simply passed the dependencies into where it is needed , is the most viable, since it removes the responsibility of locating the dependency from the component. The dependency is simply handed to the component.That's Dependency Injection. Nothing more! Using the SomeClass class is now a bit more involving as you first need to create the object:

```
myObject = new Object(objectname);

someInstance = new SomeClass(myObject);
```

Basically, DI can be simply explained as follow, instead of having the objects creating a dependency or asking a factory object to make one for them, you pass the needed dependencies in to the constructor, and you make it somebody else's problems . One of the major advantages of dependency injection is that it can make testing lots easier.

This is just for your first sight of what will happen to the code and how it looks like when we use DI. Ill explain why we are doing this and its benefits later. First, lets see what is the definition of this design pattern.

Definition

Dependency injection (DI) is a software design pattern that allows removing hard-coded dependencies and making it possible to change them, whether at run-time or compile-time[1]

Dependency injection involves at least three elements:

- a dependent consumer,
- a declaration of a component's dependencies, defined as interface contracts,
- an injector (sometimes referred to as a provider or container) that creates instances of classes that implement a given dependency interface on request.

The dependent object describes what component it depends on to do its work. The injector decides what concrete classes satisfy the requirements of the dependent object, and provides them to the dependent.

Being able to make this decision at run-time rather than compile time is the key advantage of dependency injection. Multiple, different implementations of a single software component can be created at run-time and passed (injected) into the same test code. The test code can then test each different software component without being aware that what has been injected is implemented differently.

Types

Dependency Injection is not restricted to constructor injection:

- Constructor injection

```
public SomeClass {

SomeClass(myObject){

 this.myObject = myObject;

}
}
```

- Setter Injection:

```
public SomeClass {

public Setter(myObject){

 this.myObject = myObject;

}
}
```

- Property Injection:

```
public SomeClass {

public ObjectProperty;

}

someClass->ObjectProperty = property;
```

Benefits

- Reduction of boilerplate code which is included in many places without any alteration and the programmers have to write more code to do minimum jobs in the application objects since all work to initialize or set up dependencies is handled by a provider component .
- Very useful when we have objects whose implementations change often
- Very useful for large projects where there is issue of maintainability, simplicity.
- Useful in unit testing, as it is easy to inject a fake implementation of a service into the object being tested by changing the configuration file, or overriding component registrations at run-time.
- Dependency Injection facilitates the writing of testable code.

Examples and explanation in AngularJS

```
1. function SomeClass(greeter) {
2. this.greeter = greeter;
3. }
4.
5. SomeClass.prototype.doSomething = function(name) {
6. this.greeter.greet(name);
7. }
```

In the above example SomeClass is not concerned with locating the greeter dependency, it is simply handed the greeter at runtime.This is desirable, but it puts the responsibility of getting hold of the dependency on the code that constructs SomeClass.To manage the responsibility of dependency creation, each Angular application has an injector. The injector is a service locator that is responsible for construction and lookup of dependencies.

```
1. // Provide the wiring information in a module
2. angular.module('myModule', []).
3.  // Teach the injector how to build a 'greeter'
4. // Notice that greeter itself is dependent on '$window'
5. factory('greeter', function($window) {
6. // This is a factory function, and is responsible for
```

```
7. // creating the 'greet' service.
8. return {
9. greet: function(text) {
10. $window.alert(text);
11. }
12. };
13. });
14.  // New injector is created from the module.
15. // (This is usually done automatically by angular bootstrap)
16. var injector = angular.injector(['myModule', 'ng']);
17.  // Request any dependency from the injector
18. var greeter = injector.get('greeter');
```

3.1.4 Separation of Concern

Introduction - What is Separation of Concerns?

Development today is not what it used to be. Applications are often far too large for any one person to maintain. Most large applications now require fairly large development teams just to keep them running. The applications they develop get bigger and bigger, application structure and organization grow in importance.Therefore, we need a way to organise and separate components of the application which will make the code maintainable, easily accessible and better performanced.

There is a typical example that you can have a look for your first image what separation of concerns is. Have you ever see a site where HTML, CSS, JavaScript are all mixed together in one single file and all the components are glued like a mess. And its a nightmare to make change or develop any part of that site.

A Bad Example where the separation of concerns is violated

```
<a href="/some/url/"onclick="someFunction(); return false;" style="
```

Good Example

In this good example the same thing is done with a clear separation between content, style and behavior.

HTML

```
<a href="/some/url/" id="some_link">A good link</a>
```

CSS

```
#some_link { font-size: 12px; color: #f00; }
```

JavaScript

```
document.getElementById('some_link').addEventListener('click',
```

Definition

Separation of Concerns (SoC) is a standard practice of dividing up major components of an application and segregating them from the rest of the application. There are many different ways to accomplish SoC and one of them is to utilize the magic of Dependency Injection (DI). SoC via DI is the practice of splitting up the application into components, and then resolving any dependencies those components have on other components through the use of injection. For now, you dont need to worry about what DI is, ill explain this concept in another place.

Benefits

- Allow people to work on individual pieces of the system in isolation.
- To enable everyone to better understand the system.
- To facilitate reusability simplifying development and maintenance of computer programs. When concerns are well separated, individual sections can be developed and updated independently.
- the ability to later improve or modify one section of code without having to know the details of other sections, and without having to make corresponding changes to those sections.

Example of Separation of Concerns in AngularJs

This is an unit test example within the Test Driven Development(TDD) process, in which we will write the unit tests before writing anything else. This can only achieved by implementing Separation of Concerns
We can declare what we want in our view:

```
<a href="/hello" when-active>Hello</a>
```

Okay, now we can write a test:

```
it( 'should add "active" when the route changes', inject(function()
  var elm = $compile( '<a href="/hello" when-active>Hello</a>' )( $s

  $location.path('/not-matching');
  expect( elm.hasClass('active') ).toBeFalsey();

  $location.path( '/hello' );
  expect( elm.hasClass('active') ).toBeTruthy();
}));
```

We run our test and confirm that it fails. So now we can write our directive:

```
.directive( 'whenActive', function ( $location ) {
  return {
    scope: true,
    link: function ( scope, element, attrs ) {
      scope.$on( '$routeChangeSuccess', function () {
        if ( $location.path() == element.attr( 'href' ) ) {
          element.addClass( 'active' );
        } else {
          element.removeClass( 'active' );
        }
      });
    }
  };
});
```

3.1.5 Inversion of Control

Introduction

Inversion of control (IoC) is not a new term in computer sciences anymore, although its still a pretty new topic in software development. This term was first brought by Martin Fowler in 1988[4] and its sometime referred as the Hollywood Principle [4] which states Don't call us, we'll call you". I found this more obscure and hard to understanding if i throw a definition in the first stage when someone just tries to get what is it. Therefore, there will an example and explanation before we re going to the definition of IoC.

Lets see this example, say my app has a diagram editor and to implement this diagram editor i use the gojs library s APIs. In this case, i hard code the class of object that i use to this App class and handle everything. This is what the could look likes when we work in the procedure process.

```
public class App
{
    private GojSDiagram Diagram;
    public App()
    {
        Diagram = new GojSDiagram ();
    }
}
```

In an IoC scenario we would instead do something like this:

```
public class App
{
    private GojSDiagram Diagram;
    public App(GojSDiagram  Diagram)
    {
        this.Diagram = Diagram;
    }
}
```

Now, the client creating the App class has the control over which Diagram implementation to use. We're injecting the App with the dependency. If you ve read about the Dependency injection (DI) , you may say that this example of IoC is kind similar to DI. And the answer is yes it is, in fact DI is a special kind of IoC and it s a way to implement IoC.

Definition

Inversion of Control is an abstract principal describing an aspect of some software architecture design in which the flow of control of a system is inverted in comparison to procedural programming.[6]

Benefits

This is the main reason why we have to understand these obscure definition and take our time to read about IoC. These are the advantages what IoC can bring to your applicaiton

- Flexibility - Changing the implementation class for a widely used interface is simpler (e.g. replace a mock web service by the production instance)
- Readability - the project has one unified and consistent component model and is not littered with factories. The code is briefer and is not littered without dependency lookup code
- Testability - dependencies are easy to replace mocks when they're exposed through a constructor or setter. Easier testing leads to more testing. More testing leads to better code quality, lower coupling, higher cohesion

3.1.6 Angular 's key features

Directives

One of the best parts of Angular is that you can write your templates as HTML. You can do this because at the core of the framework weve included a powerful DOM transformation engine that lets you extend HTMLs syntax. Angular comes with many directives that help you define the view for your app. Well see more of them soon. These directives can define what we commonly view as the template. They can declaratively set up how your application works or be used to create reusable components.

And youre not limited to the directives that Angular comes with. You can write your own to extend HTMLs template abilities to do anything you can dream of.

A basic pseudo-code template for creating any directive follows:

```
var myModule = angular.module(...);
myModule.directive('namespaceDirectiveName', function factory(injec
 var directiveDefinitionObject = {
   restrict: string,
   priority: number,
   template: string,
   templateUrl: string,
   replace: bool,
   transclude: bool,
   scope: bool or object,
   controller: function controllerConstructor($scope,
                                              $element,
                                              $attrs,
                                              $transclude),
   require: string,
   link: function postLink(scope, iElement, iAttrs) { ... },
   compile: function compile(tElement, tAttrs, transclude) {
     return {
       pre: function preLink(scope, iElement, iAttrs, controller) {
       post: function postLink(scope, iElement, iAttrs, controller)
     }
   }
 };
 return directiveDefinitionObject;
});
```

The following table provides an overview of when you would use each of the options.

Two-way DataBinding

Data-binding in Angular web apps is the automatic synchronization of data between the model and view components. The way that Angular implements data-binding lets you treat the model as the single-source-of-truth in your application. The view is a projection of the model at all times. When the model changes, the view reflects the change, and vice versa.

- One-way databinding Most templating systems bind data in only one direction (as in 3.1): they merge template and model components to-

Property	Purpose
restrict	Declare how directive can be used in a template as an element, attribute, class, comment, or any combination.
priority	Set the order of execution in the template relative to other directives on the element.
template	Specify an inline template as a string. Not used if you're specifying your template as a URL.
templateUrl	Specify the template to be loaded by URL. This is not used if you've specified an inline template as a string.
replace	If true, replace the current element. If false or unspecified, append this directive to the current element.
transclude	Lets you move the original children of a directive to a location inside the new template.
scope	Create a new scope for this directive rather than inheriting the parent scope.
controller	Create a controller which publishes an API for communicating across directives.
require	Require that another directive be present for this directive to function correctly.
link	Programmatically modify resulting DOM element instances, add event listeners, and set up data binding.
compile	Programmatically modify the DOM template for features across copies of a directive, as when used in ng-repeat. Your compile function can also return link functions to modify the resulting element instances.

Table 3.1: Directive definition options

gether into a view, as illustrated in the diagram. After the merge occurs, changes to the model or related sections of the view are NOT automatically reflected in the view. Worse, any changes that the user makes to the view are not reflected in the model. This means that the developer has to write code that constantly syncs the view with the model and the model with the view.

- Two-way databinding The way Angular templates works is different, as illustrated in 3.2. They are different because first the template (which is the uncompiled HTML along with any additional markup or directives) is compiled on the browser, and second, the compilation step produces a live view. We say live because any changes to the view are immediately reflected in the model, and any changes in the model are propagated to the view.

MVC

MVC application structure was introduced in the 1970s as part of Smalltalk[]. From its start in Smalltalk, MVC became popular in nearly every desktop development environment where user interfaces were involved. Whether you were using C++, Java, or Objective-C, there was some flavor of MVC available. Until recently, however, MVC was all but foreign to web development.

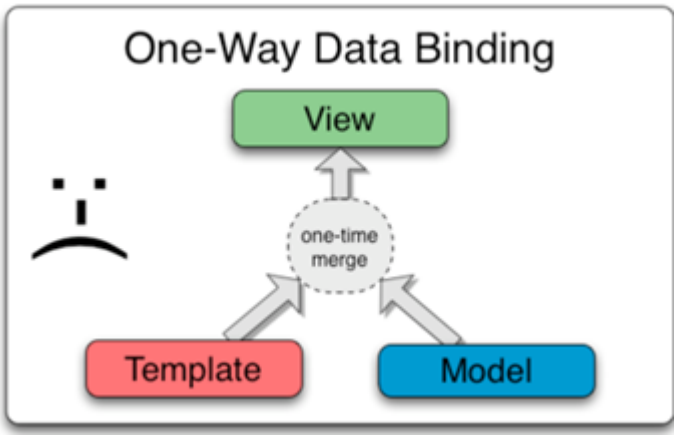

Figure 3.1: one-way databinding

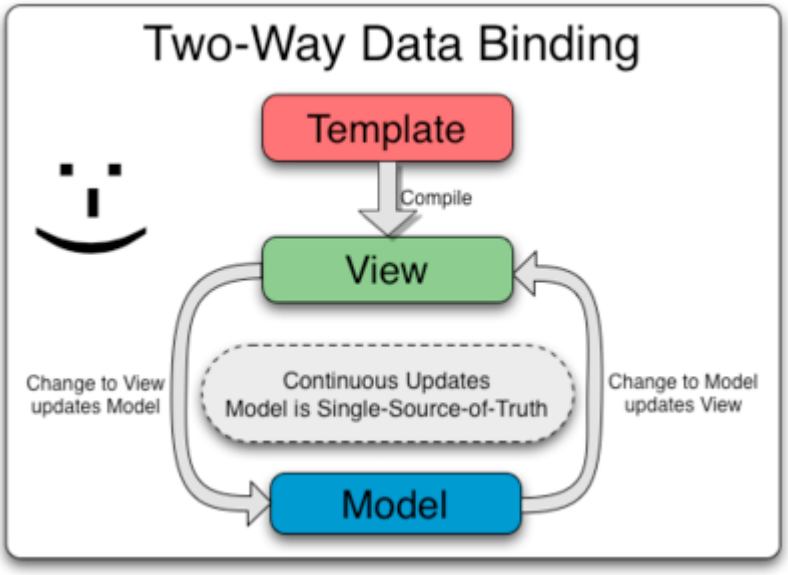

Figure 3.2: two-way databinding

The core idea behind MVC is that you have clear separation in your code between managing its data (model), the application logic (controller), and presenting the data to the user (view). Code for AngularJS applications is always organized into models, views, controllers, and (optionally) services.

Models are JavaScript objects that represent the data your application can access. Models are also used to represent the application's current state.

Views play two important roles. First, they are responsible for presenting the data from your models to the user in a visual and useful format. Second, they intercept generic user interactionsincluding clicks and option list selectionsand translate them into application-specific actions. Views in AngularJS are defined declaratively using HTML and CSS.

Controllers define the actual behavior of your application (also called "application logic") and play a key role in connecting the right models to the right views.

Services are specialized objects that perform work on behalf of other objects. Services have many uses, from fetching remote data to providing an implementation of a useful algorithm. Services are intended to be highly reusable and are designed to be swapped easily with other similar services. This concept will be presented more in details in the next part.

Services

Services are a feature that Angular brings to client-side web apps from the server side, where services have been commonly used for a long time. Services in Angular apps are substitutable objects that are wired together using dependency injection (DI).

While Angular offers several useful services, for any nontrivial application you'll find it useful to write your own custom services. To do this you begin by registering a service factory function with a module either via the Module#factory api or directly via the $provide api inside of module config function.
c Example of Using the angular.Module api:

```
1. var myModule = angular.module('myModule', []);
```

```
2. myModule.factory('serviceId', function() {
3.   var shinyNewServiceInstance;
4.   //factory function body that constructs shinyNewServiceInstanc
5.   return shinyNewServiceInstance;
6. });
```

Using the $provide service:

```
1. angular.module('myModule', [], function($provide) {
2.   $provide.factory('serviceId', function() {
3.     var shinyNewServiceInstance;
4.     //factory function body that constructs shinyNewServiceInsta
5.     return shinyNewServiceInstance;
6.   });
7. });
```

Unit Testing

With the rapid development of software industry, more and more huge applications are produced. Going along with that is a problem how to know the quality of the software products or services before publishing them? That is where software testing taking places. In fact testing is an investigation including several processes whose purpose is to provide an objective, independent view of the software to allow the business to appreciate and understand the risks of software implementation[].

Unit testing, also known as component testing, refers to tests that verify the functionality of a specific section of code, usually at the function level. In an object-oriented environment, this is usually at the class level, and the minimal unit tests include the constructors and destructors[2]. Unit testing is usually performed by the developers working on the code themselves to verify the functions meet their design and behave as intended. There may be multiple tests on one function in order to catch all the cases. Unit testing alone cannot verify the functionality of a piece of software [3] , but it is known to be most effective means to test individual software components

3.2 GoJS

3.2.1 Overview

Gojs is a fast and powerful JavaScript library for implementing interactive diagrams in HTML5 web applications. it is released under nwoods company license.GoJS is a feature-rich JavaScript library for implementing interactive diagrams across modern browsers and platforms. GoJS makes constructing diagrams of complex Nodes, Links, and Groups easy with customizable templates and layouts. GoJS offers many advanced features for user interactivity such as drag-and-drop, copy-and-paste, transactional state and undo management, palettes, overviews, data-bound models, event handlers, and an extensible tool system for custom operations.

3.2.2 Why Gojs?

The reason why Gojs is chosen it that this library is a new product and providing such beautiful graphics and various kind of sample shapes. The implementation is based on JavaScript and has a small footprint (659KB (free trial)). Since Gojs has no dependencies it can be integrated with Angular.js. For developers there is an in-details introduction, a lot of sample code and demos. Besides, there is a forum for developers to ask question and it is well-replied by the admins of the site.

3.2.3 Features

GoJS makes constructing diagrams of complex Nodes, Links, and Groups easy with customizable templates and layouts. GoJS offers many advanced features for user interactivity such as drag-and-drop, copy-and-paste, transactional state and undo management, palettes, overviews, data-bound models, event handlers, and an extensible tool system for custom operations.Besides GoJS supports scrolling around and zooming into the diagram. it also has built-in support for customizable tooltips and programmer-defined context menus that can be different for the diagram's background or for each node, group, or link template. Moreover, data binding, in-place editing, multiple selection and over 195+ common shapes are supported.

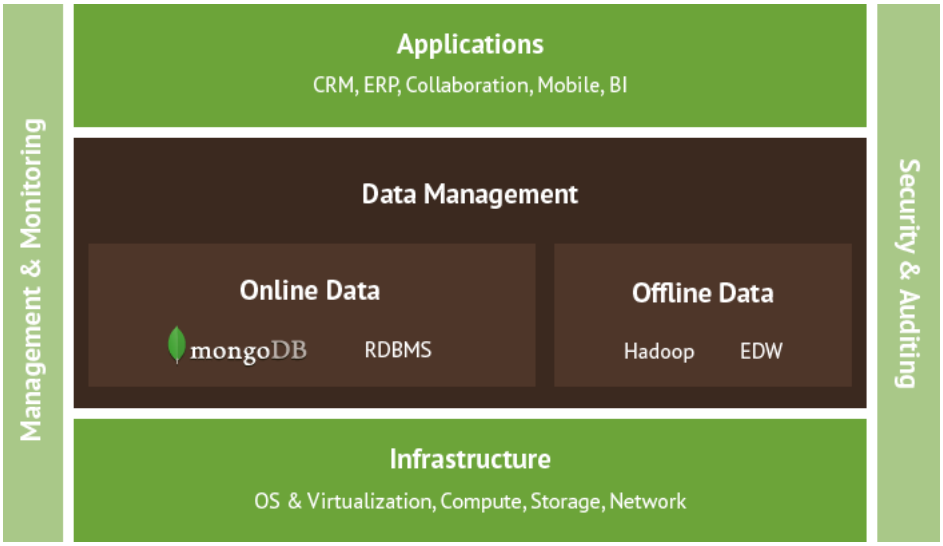

Figure 3.3: mongodb stack

3.3 Mongodb

3.3.1 Overview

MongoDB is the leading NoSQL database, empowering businesses to be more agile and scalable. Fortune 500 companies and startups alike are using MongoDB to create new types of applications, improve customer experience, accelerate time to market and reduce costs.It is used by companies of all sizes, across all industries and for a wide variety of applications. It is an agile database that allows schemas to change quickly as applications evolve, while still providing the functionality developers expect from traditional databases, such as secondary indexes, a full query language and strict consistency.

3.3.2 Why Mongodb?

MongoDB is built for scalability, performance and high availability, scaling from single server deployments to large, complex multi-site architectures. By leveraging in-memory computing, MongoDB provides high performance for both reads and writes. MongoDBs native replication and automated failover enable enterprise-grade reliability and operational flexibility.

3.3.3 Introduction to MongoLab

MongoLab is a fully-managed cloud database service featuring highly-available MongoDB databases, automated backups, web-based tools, 24/7 monitoring, and expert support. Since its inception in 2011, MongoLab has grown like wildfire and now manages tens of thousands of databases across five major cloud providers in 13 datacenters worldwide.

MongoLab help to connect to your database using standard MongoDB drivers in the language of your choice or use MongoLab's JSON-based REST API. It keeps your database close to your app server! MongoLab provides MongoDB hosting on all the major cloud platforms Amazon, Google, Joyent, Rackspace, and Windows Azure and has partnered with all of the major Platform-as-a-Service providers to enable seamless integration with the application tier.

3.4 Some UI Widget libraries

3.4.1 Angular Bootstrap

Angular Bootstrap is Bootstrap components written in pure AngularJS which provide directive for using Twitter Bootstrap 's components and go along with AngularJS. They re-write some of Twitter Bootstrap 's components like accordion, drop-down, Modal, buttons into AngularJS 's directive for easier use and integration.

3.4.2 JqueryUI

Query UI is a curated set of user interface interactions, effects, widgets, and themes built on top of the jQuery JavaScript Library. It provides abstractions for low-level interaction and animation, advanced effects and high-level, themeable widgets, built on top of the jQuery JavaScript library, that can be used to build interactive web applications.

3.4.3 Semantic-UI

Semantic-UI is the vocabulary of the web.Semantic empowers designers and developers by creating a language for sharing UI. It provides various kinds of widgets and components with modern design and effects.

3.5 HTML5

HTML5 aims to make HTML more useful for creating web applications as well as semantically marked up documents, is not yet a formal Recommendation as of this writing, however, it is beginning to gain browser support and is already being used for web and mobile application development.HTML5 offers new features (elements, attributes, event handlers, and APIs) for easier web application development and more sophisticated form handling. There are also new semantic elements for marking up page content. Most of the purely presentational or poorly supported elements and attributes in HTML 4.01 have been dropped from HTML5, however, a few have been redefined or reinstated.

3.6 CSS3

CSS (Cascading Style Sheets) consist of a group of formatting rules that you use to control the layout and appearance of the content on a web page. One really great feature of CSS is that you can store all the CSS rules in one document and keep that document separate from the HTML content and link the two together. Then, when you make a change to the CSS that change is instantly and automatically updated on all the HTML files. Another great feature is that it ”cleans up” the appearance of the code on web pages. In addition it will speed up browser loading times.

Chapter 4

BUILDING AN APPLICATION USING SELECTED TECHNOLOGIES

The story in this chapter is about how all the state of the art technologies are integrated and utilized in every single piece of the application. Besides, All features are presented accordingly

4.1 Features

At first sight, the application consists of 4 main parts: Palettes, Canvas, Menu & toolbar and sidebar as in figure 4.1. Therefore, all the features can also be divided into 4 different groups and will be described in details. Generally, the user can see all the shapes displayed based on category in the palettes on the left and drag into the canvas on the right. The user also interact with the diagram directly as well as see more information about a selected node in the sidebar. There are also menu and toolbar for the user to have many more options.

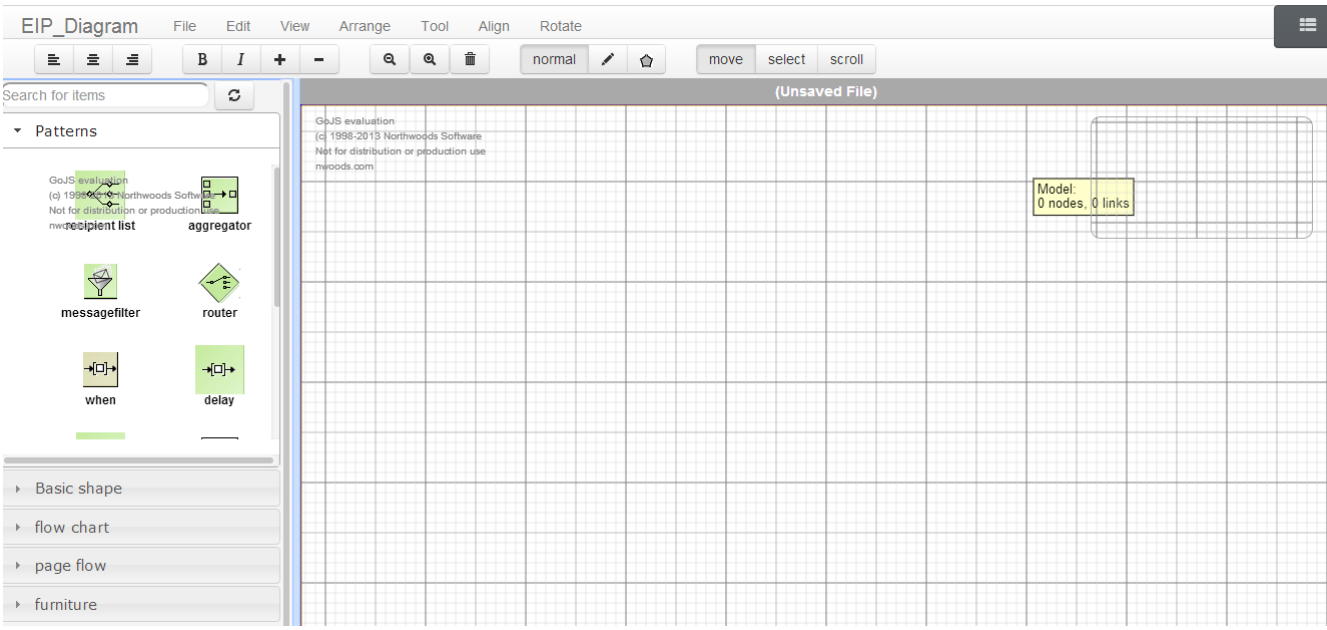

Figure 4.1: application 's interface

4.1.1 Palettes & search

As in figure 4.1 The palettes are on the left of the application separated with the canvas by a splitter with can be handled to change the size of the palettes. Besides, many types of shapes are sorted by category, users can open the section with suitable category to see all the shapes. Moreover, there is a search box on top of the palettes. Users can find shapes by names or description and Enter to add that shape to the canvas as in figure 4.2

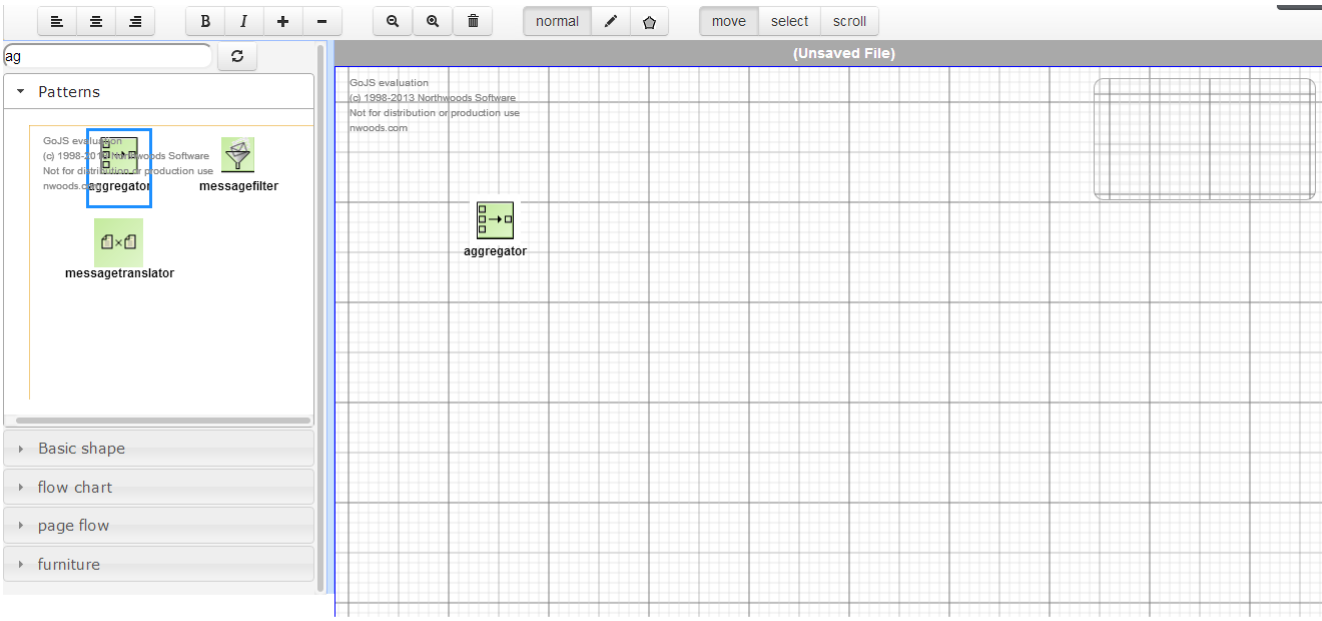

Figure 4.2: Search function

4.1.2 Interactive Diagram

Double-click create

Doubld-click create cuts down several actions in traditional diagram software to 1 click action. That is much faster drawing! Add the next shape similar to the selected one immediately to the canvas with 1 double-click.

Context menu

The context menu shows up by a single right-click base on what the user selected(node, link or background).

Connection

connecting object by mousing down on a port, move (drag) to nearby an input port, and then mouse-up to complete the link. The second way is to drag the node close in range and they will automatically connected. They will be connected by the whole node instead of a single port. By default the user

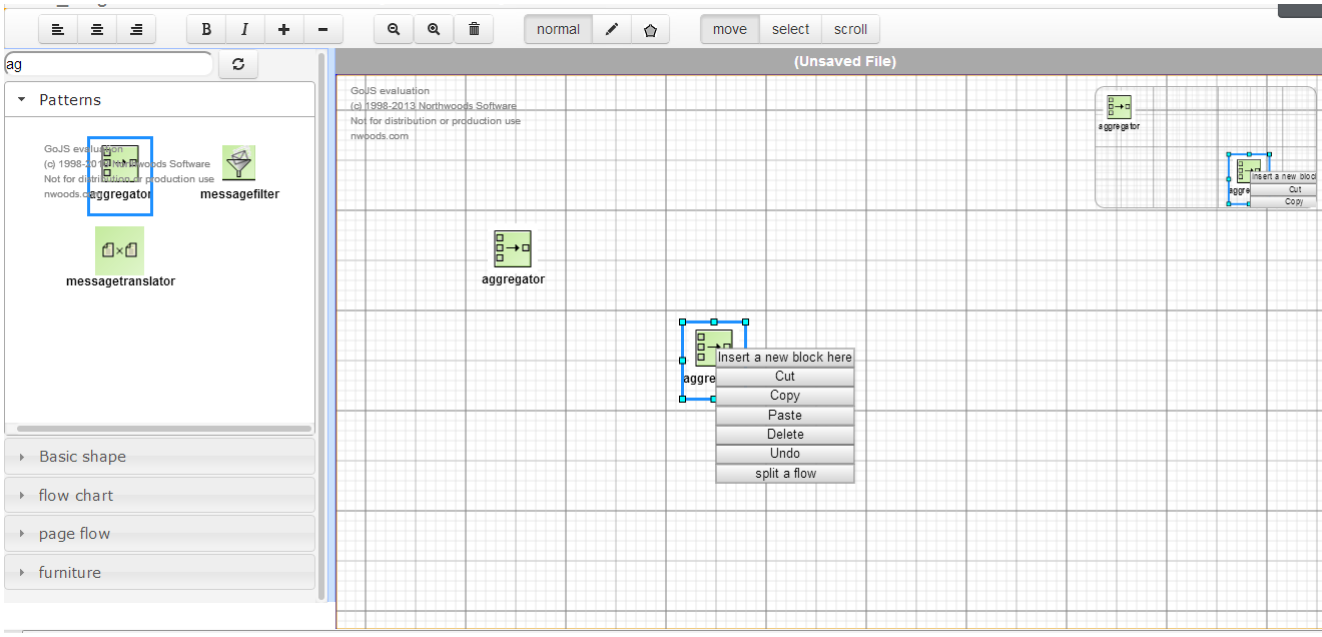

Figure 4.3: context menu

may not draw more than one link in the same direction between any pair of ports, nor may the user draw a link connecting a node with itself.

Aligned, sized and grouped

Users can align a set of selected nodes, resize one or multiple nodes, and grouping one or many nodes. Adding more node to the group just by dragging.

4.1.3 Sidebar

The user can view information of a single selected node by open the sidebar as in figure 4.4. They can view the type, width, height, position and change color of that node as in figure 4.5

4.1.4 Menu & Toolbar

There are several categories on the Menu such as : file, Edit, View, Arrange, Tool, Align, Rotate with corresponding functions. On the toolbar there are some buttons for zooming, Change size of the node, text manipulation, copy, paste, delete

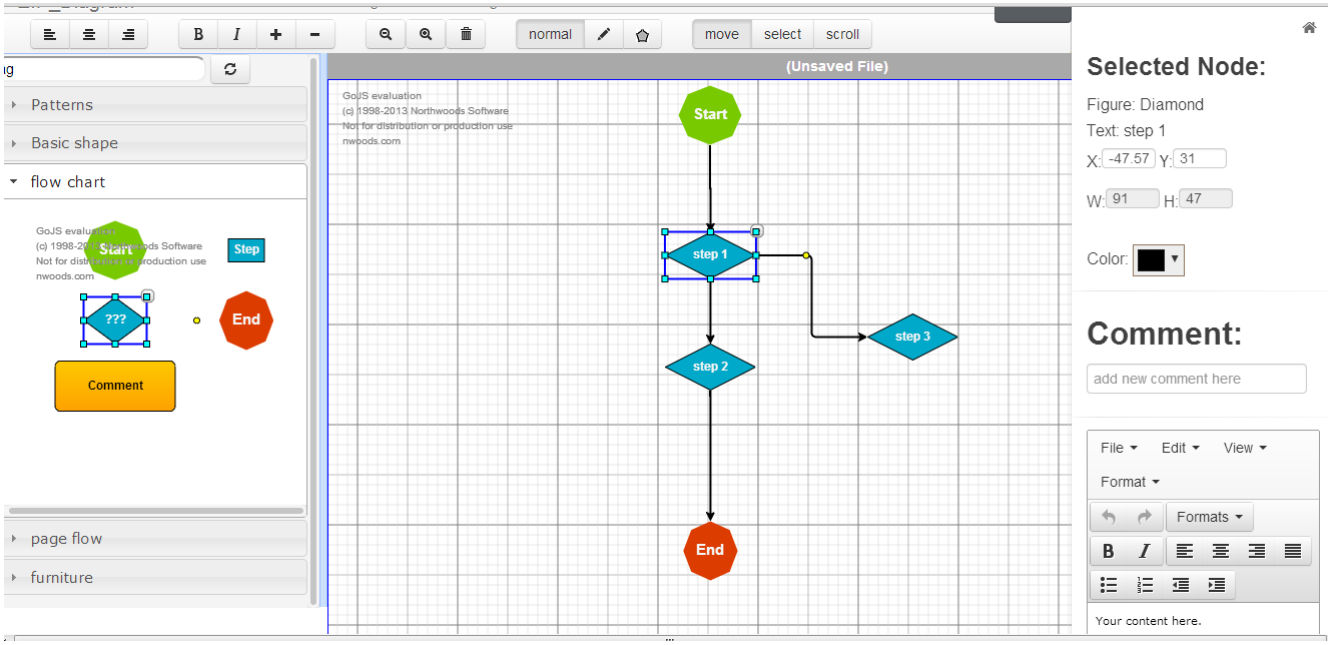

Figure 4.4: sidebar

4.2 Database

MongoLab is a fully-managed cloud database service featuring highly-available MongoDB database. It's a NoSQL database which provides a mechanism for storage and retrieval of data that is modeled in means other than the tabular relations used in relational databases. In mongolab, documents and collections are used instead of tables and rows. This application uses two collections. They are Node (all the shape storage) and template (for templates storage).

4.3 Front-end technologies

4.3.1 Angular Bootstrap

In this application, buttons, font and menu are the widget that used from Bootstrap. This library also provide a set of glyph icon that is used in the toolbar.

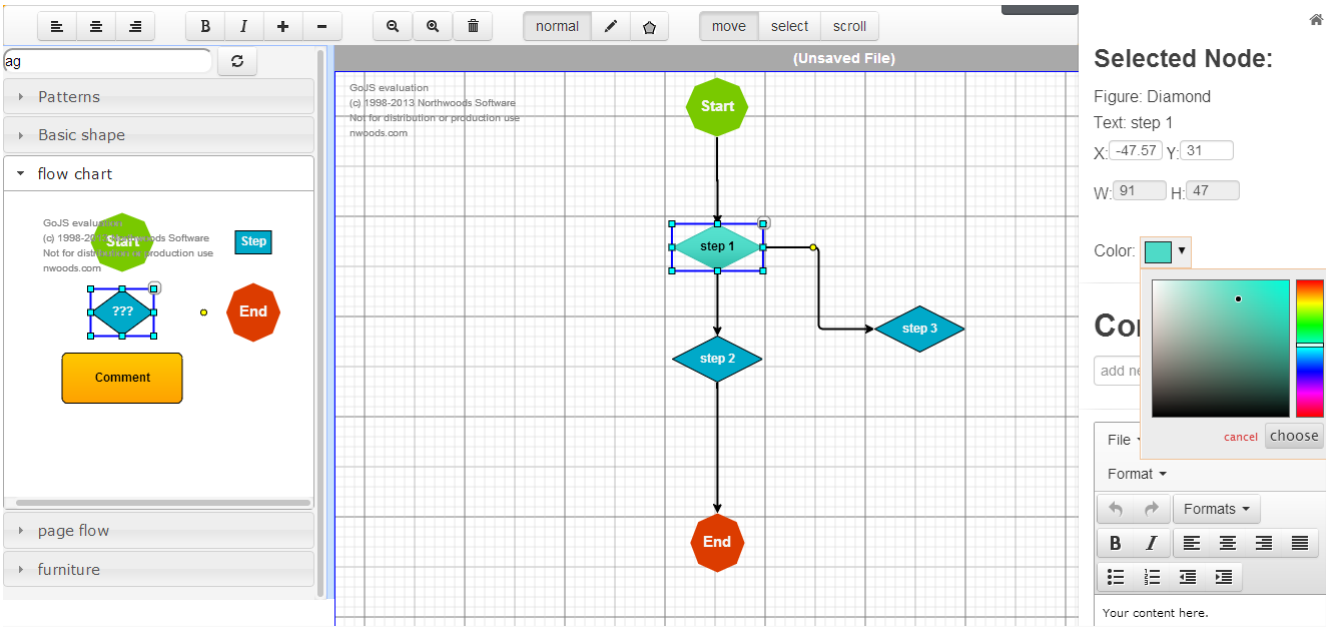

Figure 4.5: change color of a node in sidebar

4.3.2 JqueryUI

JqueryUI 's widgets are also integrated in this application. The palettes are made by using accordion component of this library. Besides, the splitter is also based on Jquery library that help to make nice animation.

4.3.3 Semantic-UI

The sidebar is a great widget using from semantic-ui library. It is a menu hidden beside page content and when activated it floats over the content of the page. It makes the theme clean and delicate with smooth animation.

4.4 Back-end technologies

4.4.1 AngularJS

application 's structure

The first thing AngularJS brought to the application is the structure. It should be clear already that Angular applications are structured very differently than similar applications were in the past.It has the services layer like in figure

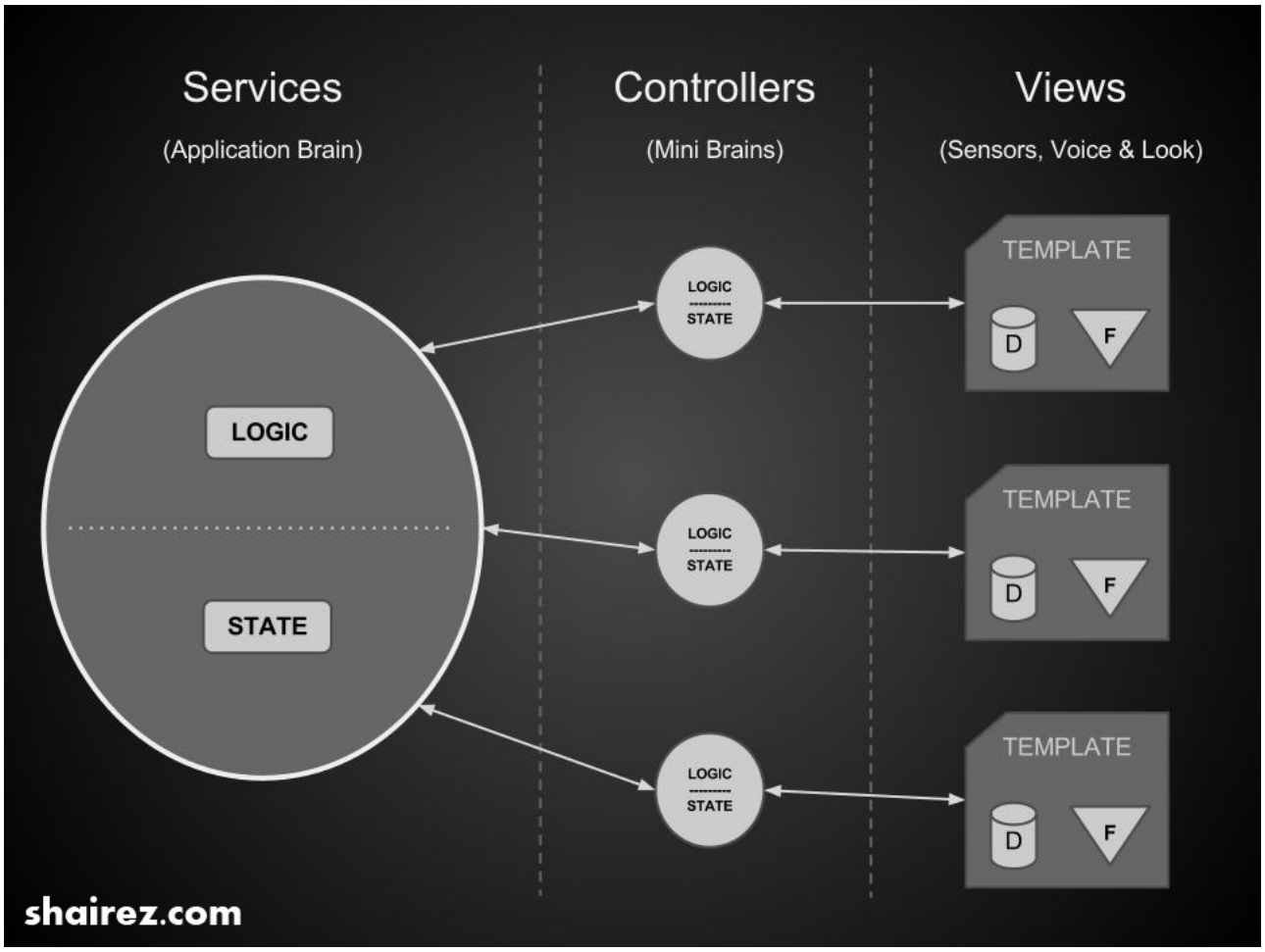

Figure 4.6: Angular application 's structure

4.6, which is like a giant application brain that handles all of the state coordination, data persistence and the business logic. The angular controllers are like mini brains, less capable and are used to set up the initial state of the $scope object or add behavior to the $scope object. Templates and directives are like sensors and speech, which are responsible for displaying or "speaking" data to the user, and to take in outside information (like user events or data) and pass it to the controller. The application folders are also organized as in figure 4.7.

Figure 4.7: Angular application 's folder organization

MVC

Data Binding

Before AJAX single-page apps were common, platforms like Rails, PHP, or JSP helped us create the user interface (UI) by merging strings of HTML with data before sending it to the users to display it. Here with data binding i just declare which parts of the UI map to which JavaScript properties and have them sync automatically. It works great with MVC to eliminate code when writing view and model. Most of the work in moving data from one to the other just happens automatically. From now on the view will be automatically updated when the model changed . how great is this?

Changing the DOM with Directives

Directives extend HTML syntax, and are the way to associate behavior and DOM transformations with custom elements and attributes. Through them, it's very easy to create reusable UI components, configure the application, and do almost anything else ones can imagine wanting to do in the UI template. Writing apps with the built-in directives that come with Angular, but someteims, it's likely to run into situations where we want to write our own. Its time to break into directives when dealing with browser events or modify the DOM in a way that isnt already supported by the built-in directives. This

code belongs in a directive that you write, and not in a controller, service, or any other place in your app. From now on instead of dealing with such long html code like this

```
<div class="dropdown">
  <!-- Link or button to toggle dropdown -->
  <ul class="dropdown-menu" role="menu" aria-labelledby="dLabel">
    <li><a tabindex="-1" href="#">Action</a></li>
    <li><a tabindex="-1" href="#">Another action</a></li>
    <li><a tabindex="-1" href="#">Something else here</a></li>
    <li class="divider"></li>
    <li><a tabindex="-1" href="#">Separated link</a></li>
  </ul>
</div>
```

we totally can write like this:

```
<div dropdown-menu></div>
```

Angular supports the Model View Controller style of application design. Though it has a lot of flexibility in designing an Angular application, a brief description of what MVC in figure 4.8 can bring into this :

- A model containing data that represents the current state of the application.
- Views that display this data.
- Controllers that manage the relationship between models and views

4.4.2 GoJS

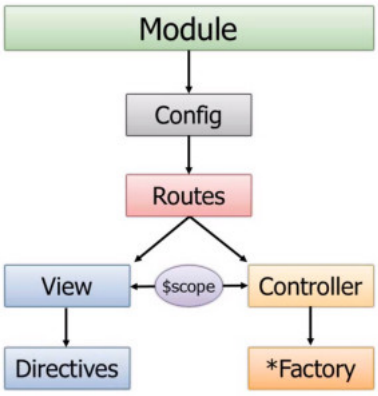

Figure 4.8: MVC structure in Angular

Chapter 5

DISCUSSION & CONCLUSION

In this chapter, discussion about all the explored technologies and evaluation about the application has just been built using and integrating such technologies. Besides, i will discuss further about the future work to improve performance as well as adding more features and improve code organization

5.1 Discussions of key technologies and concepts

5.1.1 Front-end

This application uses some libraries that support and provide very beautiful and delicate widget such as accordion and splitter of JqueryUI, font, theme, color, menu of Twitter Bootstrap, Sidebar and modal of semantic-UI. They have very specific advantages that each library provide a nice set of Widgets and easily to integrate with other libraries. The combination of all these libraries gives the application a very delicate and nice look. Without such widgets, create an nice application takes a lot more times and effort and may cause errors.

5.1.2 Back-end

AngularJS

AngularJS is an open-source JavaScript framework, maintained by Google, that assists with running single-page applications. Its goal is to augment browser-based applications with modelviewcontroller (MVC) capability, in an effort to make both development and testing easier. It give the RIA a nice structure with using many software enginering concepts like dependencies injection ,separation of concern and Inversion of control. Utilizing such advantages, this application use Angular.js to structure and support the development.However, i cannot utilize all the advantages of angular like testing and some other services.

GoJs

GoJS is a feature-rich JavaScript library for implementing interactive diagrams across modern browsers and platforms. GoJS makes constructing diagrams of complex Nodes, Links, and Groups easy with customizable templates and layouts. GoJS offers many advanced features for user interactivity such as drag-and-drop, copy-and-paste, transactional state and undo management, palettes, overviews, data-bound models, event handlers, and an extensible tool system for custom operations. However, this library lack support of UML component and sometimes there are still several errors happen.

5.1.3 Integration into one single RIA

It is such a nice work of integrating many frameworks and libraries together into one single RIA. This means this RIA is a combination of all the advantages of all the technologies which has been used. However, before the integration , a serious survey need to be conducted to choose such libraries that work together.

5.1.4 Software engineering concepts in RIA development

Two important software engineering concepts used in this RIA is Dependencies injection(DI) and Separation of concern. DI help the Reduction of boilerplate code which is included in many places without any alteration and the programmers have to write more code to do minimum jobs in the application objects since all work to initialize or set up dependencies is handled by a provider component. Separation of Concerns (SoC) is a standard practice of dividing up major components of an application and segregating them from the rest of the application. There are many different ways to accomplish SoC and one of them is to utilize the magic of Dependency Injection (DI). SoC via DI is the practice of splitting up the application into components, and then resolving any dependencies those components have on other components through the use of injection.

5.2 Discussion of demo application

5.2.1 Features and benefits

This RIA support design interactive diagram and collaboration with others.

5.2.2 Comparisons with other online diagram editors

There are some other diagram tool online such as draw.io, creately.com, griffy.com built by some big companies with stunning features. This application somehow can also provide similar features.

5.2.3 Drawbacks

Due to lack of experience and this is a one-person thesis so the scope is not big and the features cannot be compared with other tools built by large companies.

5.3 Future works and Conclusions

I have explore and apply many technology into an RIA development. I also present in-details about each technologies as well as the feature of this application. In the future, this application need to improve more about features as well as how to organize the function.

Bibliography

[1] ”Surveying the digital future” UCLA Internet Report, http://www.digitalcenter.org/pdf/InternetReportYearThree.pdf

[2] Gvu 10th www user survey, GVU, http://www.cc.gatech.edu/gvu/user_surveys/survey-1998-10/

[3] Center for the digital future: 2008 digital future report, 2009. University of Southern California (USC) Annenberg School, C.F.T.D.F.

[4] L. Rainee, Internet, broadband and cellphone statistics, 2010. http://www.pewinternet.org/Reports/2010/Internet-broadband-and-cell-phone-statistics.aspx.

[5] P. R. Center, Online activities, http://www.pewinternet.org/StaticPages/Trend-Data/Online-Activites-Total.aspx.

[6] ”Rich Internet application ” http://en.wikipedia.org/wiki/Rich_Internet_applicatior

[7] ”Web Application” http://en.wikipedia.org/wiki/Web_application

[8] Franz Josef Grneberger , ”REAL-TIME COLLABORATION SUPPORT FOR JAVASCRIPT FRAMEWORKS”

[9] Tommi Mikkonen and Antero Taivalsaari. Web Applications Spaghetti Code for the 21st Century. In SERA, pages 319328, 2008.

[10] Trygve M. H. Reenskaug. Models - Views - Controllers. http://heim.ifi.uio. no/ trygver/1979/mvc-2/1979-12-MVC.pdf, December 1979.

[11] Trygve M. H. Reenskaug. Thing-Model-View-Editor - an Example from a Planningsys-tem. http://heim.ifi.uio.no/ trygver/1979/mvc-1/1979-05-MVC. pdf, 1979.

[12] Tim Berners-Lee. WorldWideWeb, the first Web Client. http://www.w3.org/People/Berners-Lee/WorldWideWeb.html, 2012.

[13] Antero Taivalsaari and Tommi Mikkonen. The Web as an Application Platform: The Saga Continues. In EUROMICRO-SEAA, pages 170174, 2011.

[14] Jesse James Garrett. AJAX: A New Approach to Web Applications. http://www.adaptivepath.com/ideas/ajax-new-approach-web-applications, 2005.