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

Ngo Thanh Vu

December 5, 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 [1–3] . 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.

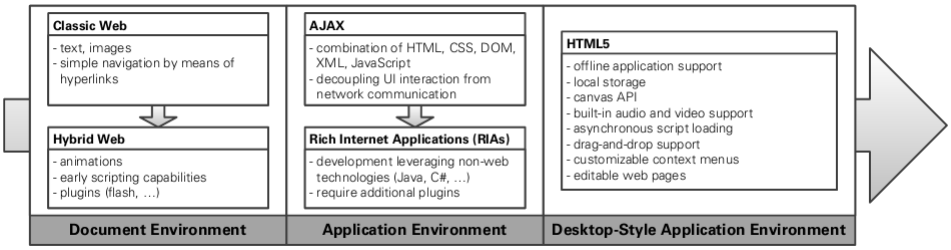

Figure 2.1: Major evolution steps of the Web [13]

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

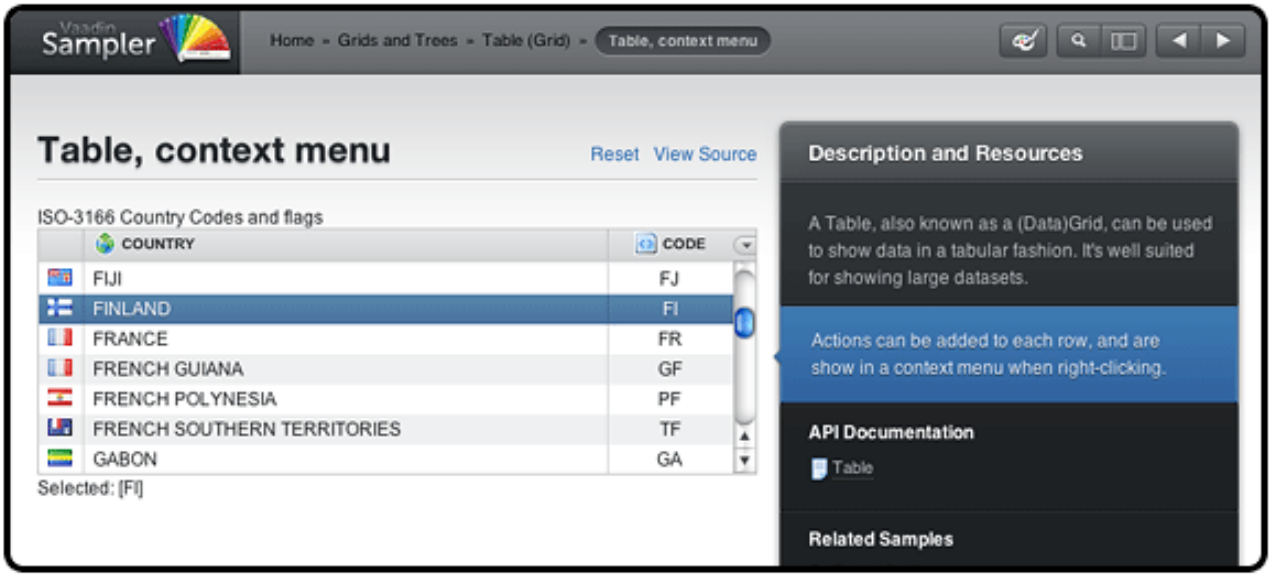

Figure 2.3: An example created by Vaadin

  - 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.

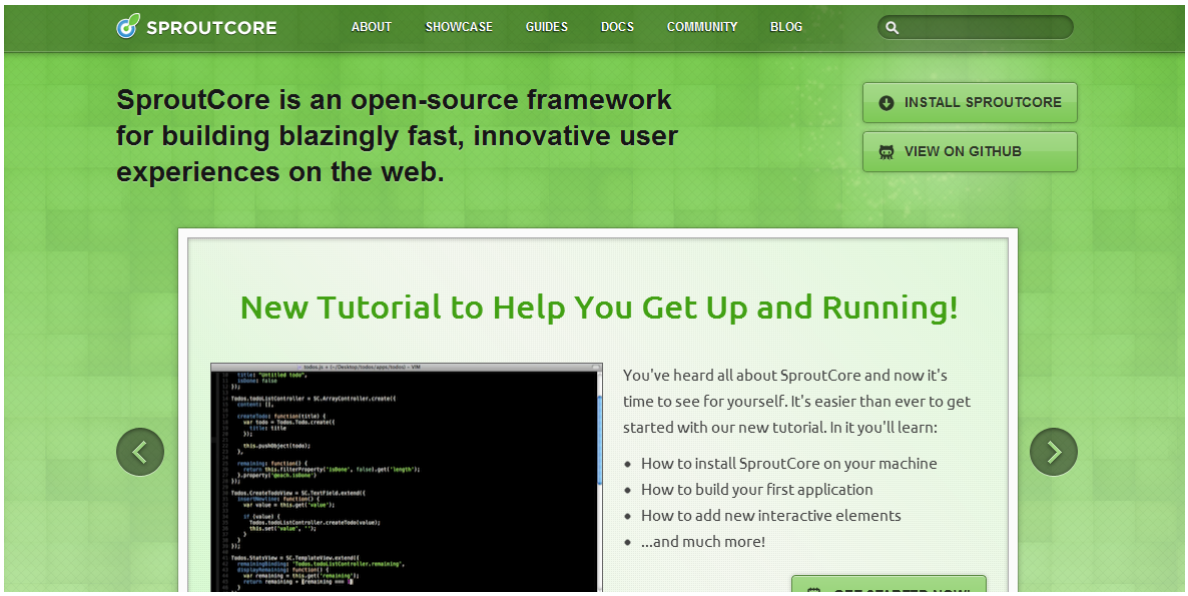

Figure 2.4: An example created by SproutCore

- 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

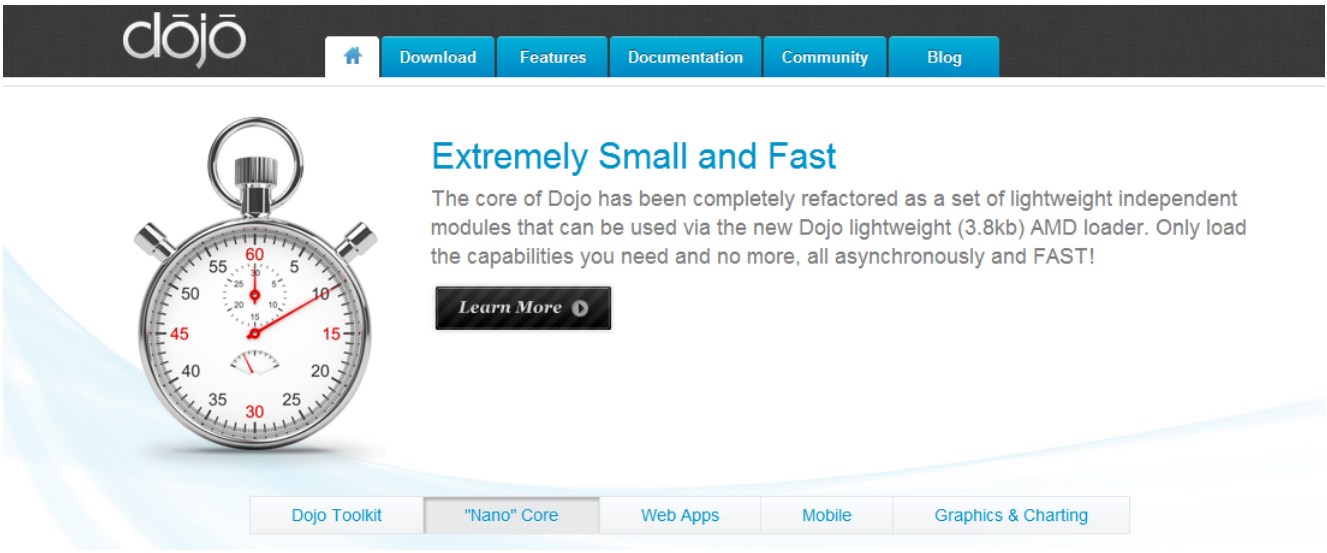

Figure 2.5: An example created by Dojo

- 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

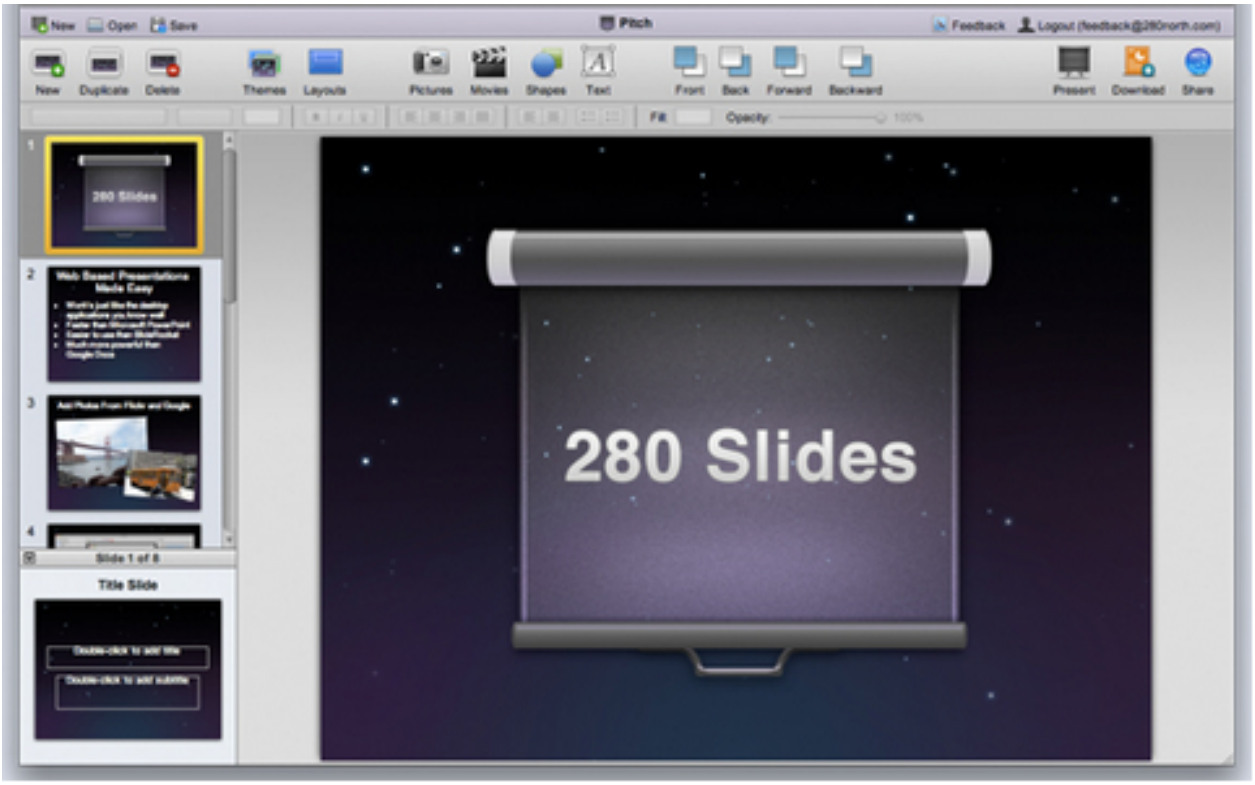

Figure 2.7: An example created by cappucino

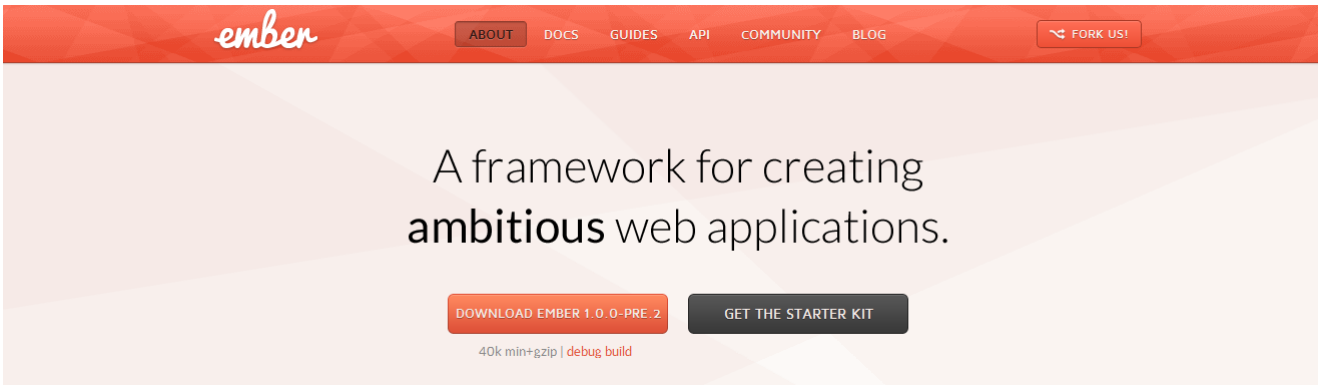

Figure 2.8: An example created by ember

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.

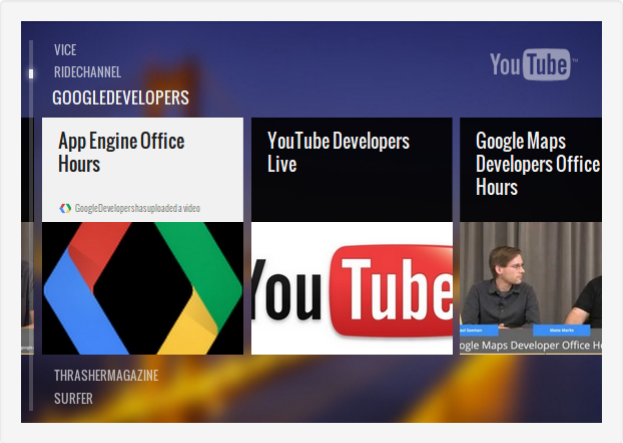

Figure 2.9: An example created by AngularJS

- 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.

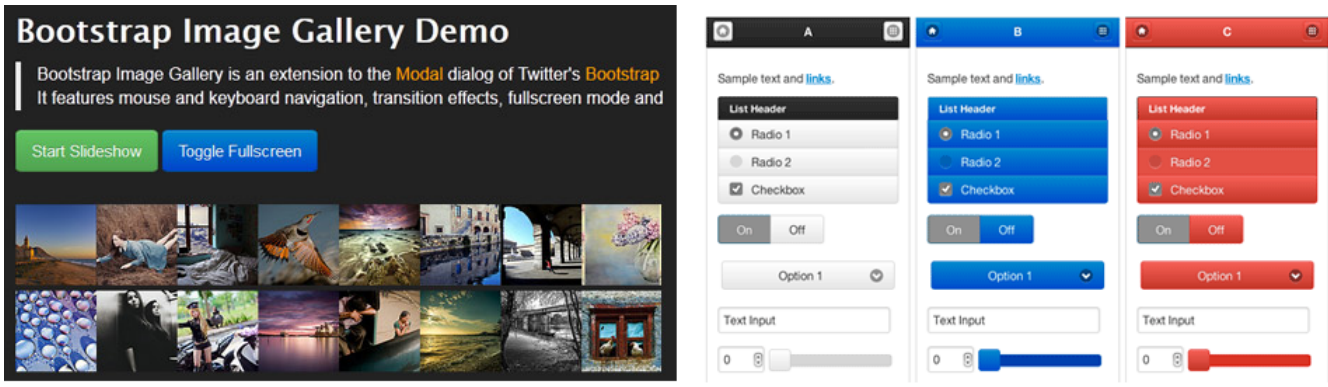

Figure 2.10: An example that created by Twitter Bootstrap

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

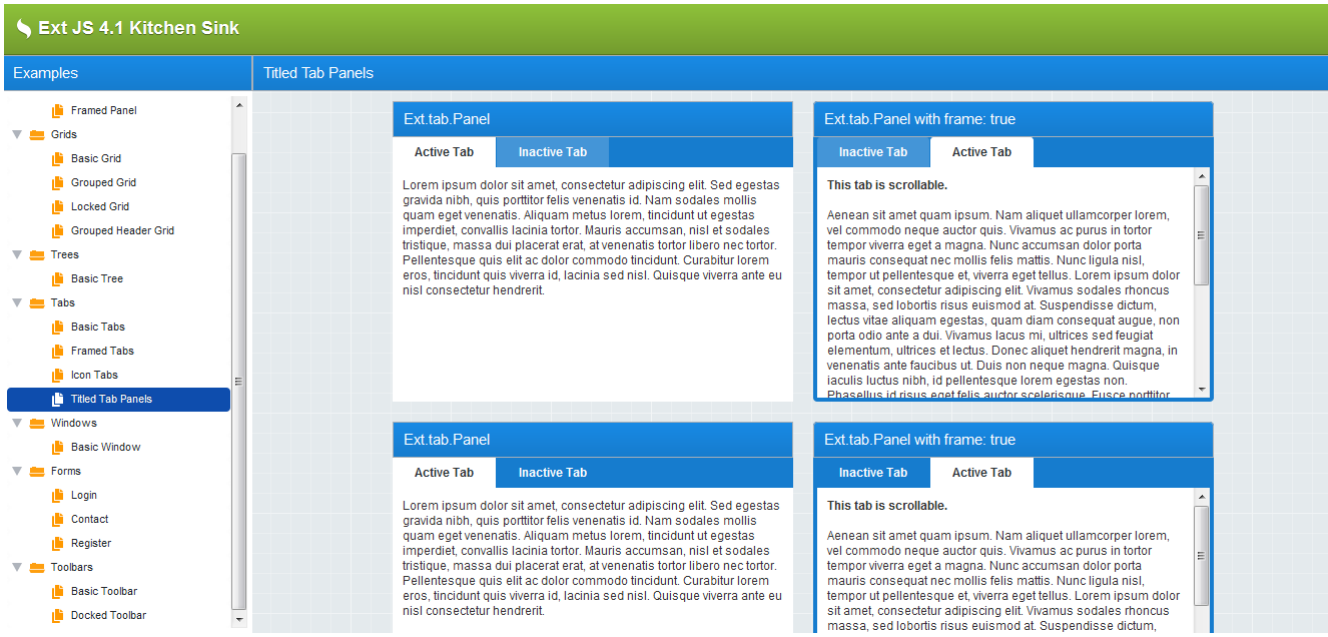

Figure 2.11: An example that created by sencha

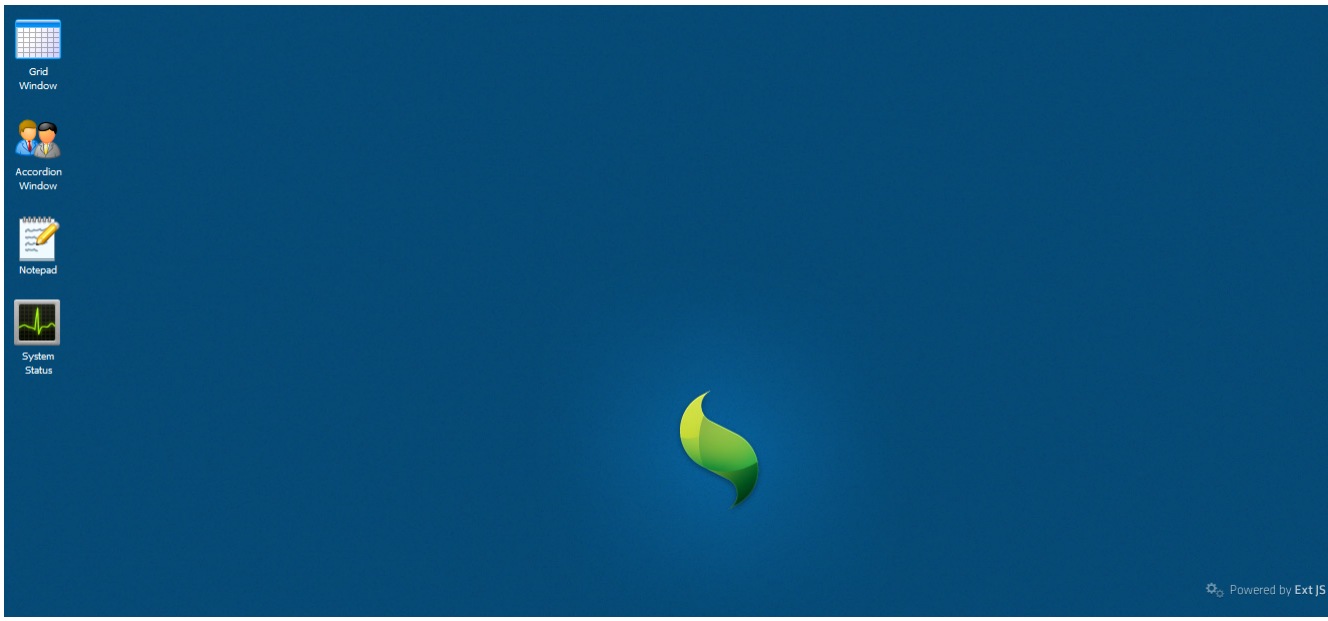

Figure 2.12: Another example that created by sencha

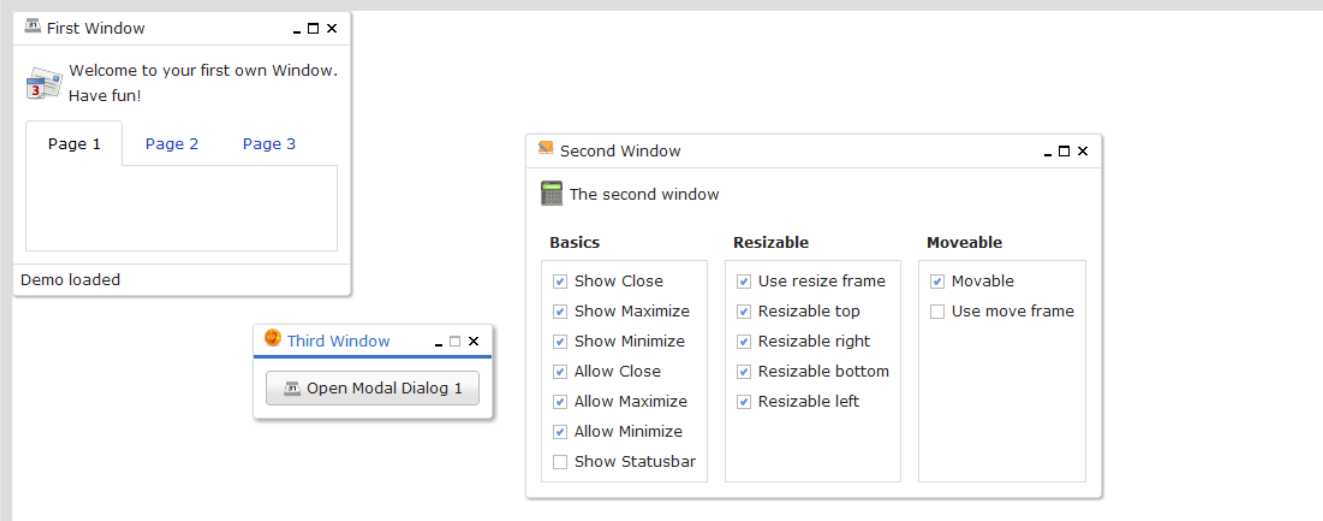

Figure 2.13: An example that created by qooxdoo

  - 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.

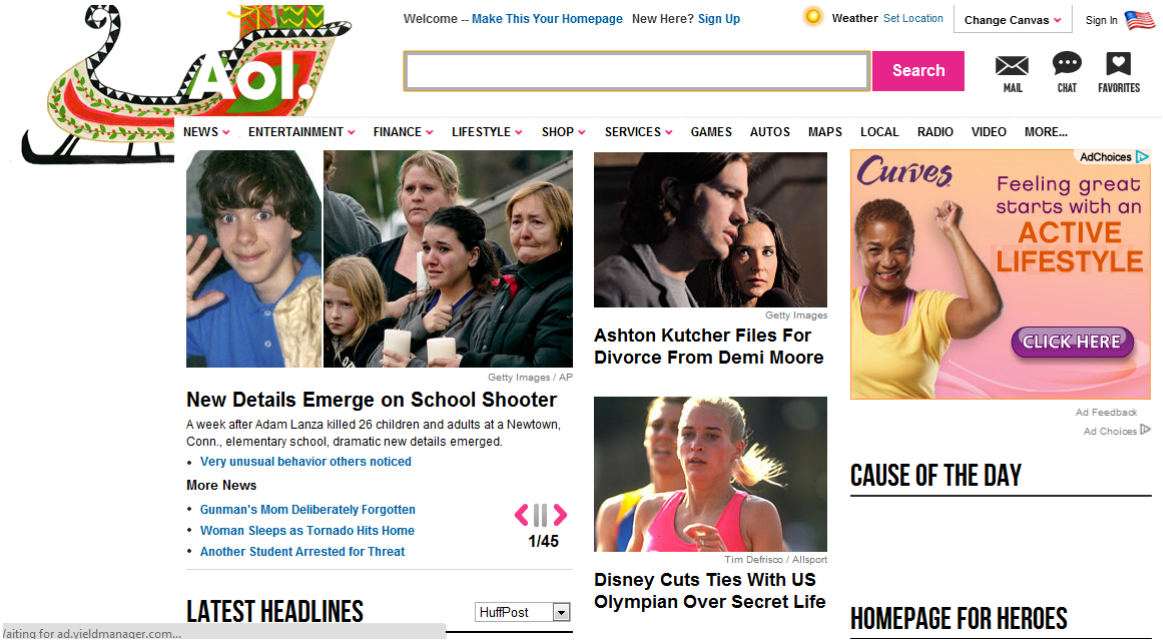

Figure 2.14: An example that created by jqueryui

  - 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.

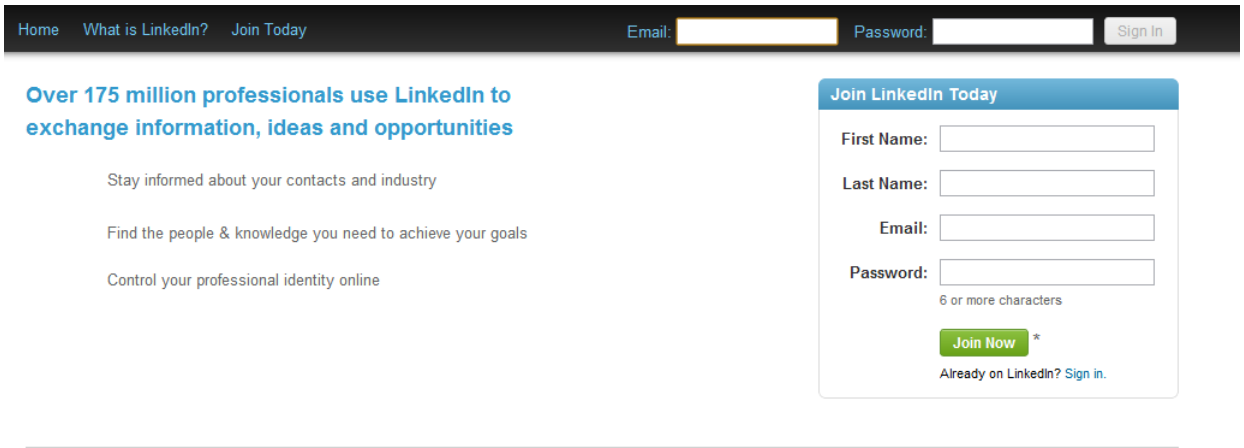

Figure 2.15: An example that created by YUI

  - 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.

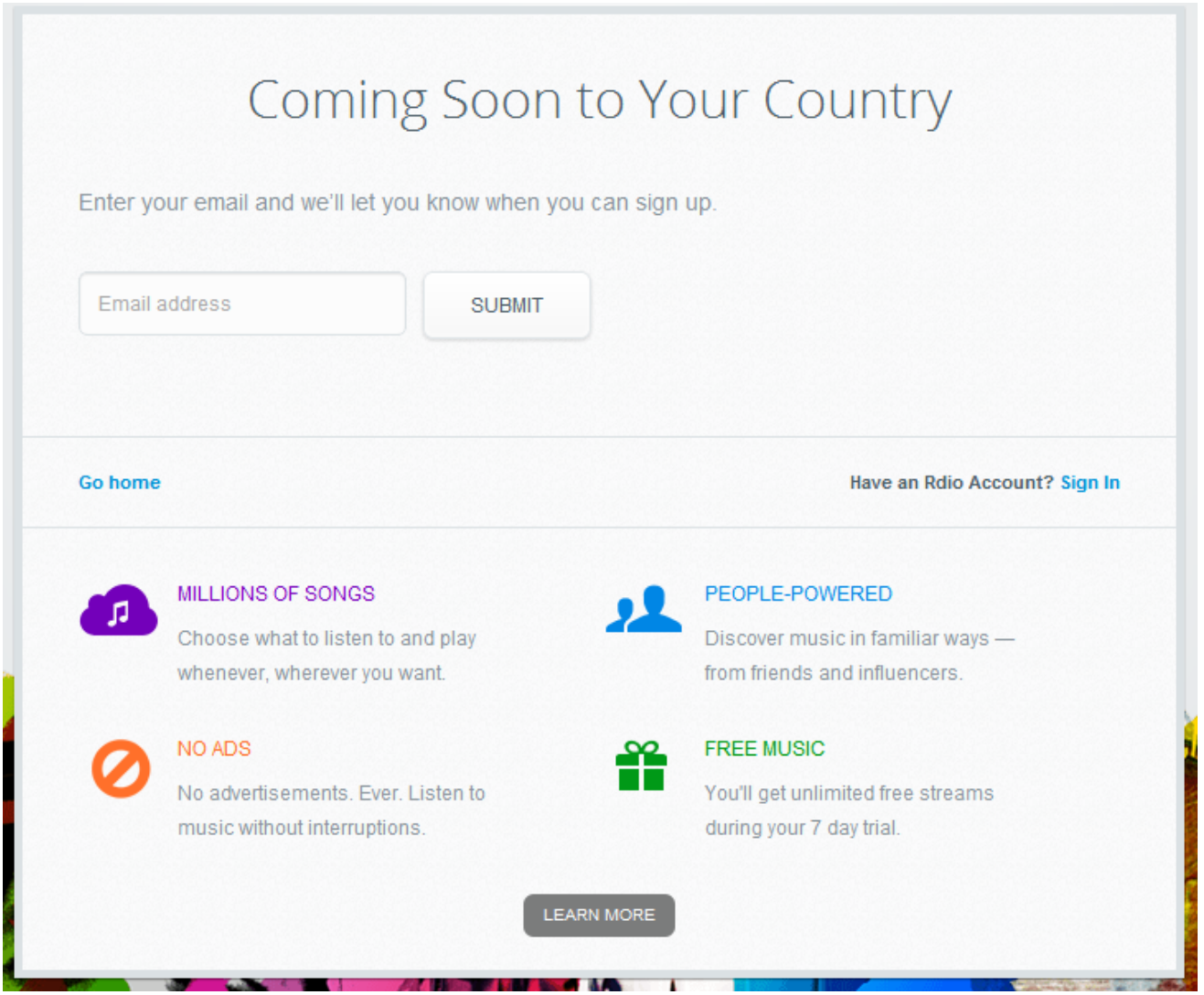

Figure 2.16: An example that created by backbone

  - 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.

	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

	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 witn 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

	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

	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/

	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

	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: Some of JavaScript frameworks in the survey

2.3 JavaScript Graph Libraries Selection

2.3.1 Overview of some javaScript Graph libraries

2.3.2 JavaScript Graph Libraries Selection Characteristics

2.3.3 Survey of Existing graph libraries

2.3.4 Selected graph library

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?

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

function SomeClass(greeter)

this.greeter = greeter;

SomeClass.prototype.doSomething = function(name)

this.greeter.greet(name);

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.
4. // Teach the injector how to build a 'greeter'
5. // Notice that greeter itself is dependent on '$window'
6. factory('greeter', function($window)
7. // This is a factory function, and is responsible for
8. // creating the 'greet' service.
9. return
10. greet: function(text)
11. $window.alert(text);
12.
13. ;
14. );
15.
16. // New injector is created from the module.
17. // (This is usually done automatically by angular bootstrap)
18. var injector = angular.injector(['myModule', 'ng']);
19.
20. // Request any dependency from the injector
21. var greeter = injector.get('greeter');
```

3.1.4 Separation of Concern

3.1.5 Inversion of Control

3.1.6 Angular 's key features

Directives

Two-way DataBinding

Filters

MVC

Service vs Factory vs provider

Testing

3.2 GoJS

3.2.1 Overview

3.2.2 Why Gojs?

3.2.3 Features

drag-and-drop

transactional state and undo management

palettes and overviews

data-bound models

event handlers and an extensible tool system for custom operations

3.3 Mongodb

3.3.1 Overview

3.3.2 Why Mongodb?

3.3.3 Introduction to MongoLab

3.4 Angular Bootstrap

3.5 Jquery

3.6 Semantic-UI

Chapter 4

BUILDING AN APPLICATION USING SELECTED TECHNOLOGIES

Chapter 5

DISCUSSION & CONCLUSION

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.