Master Thesis

Derpa derp with the derpaline in the derp

by

William Lundin Forssén

KTH-CSC/KTH—DERP–DERPA

November 20, 2012

Supervisors: Emil Birgersson (Valtech), Alexander Baltatzis (KTH)

Examiner: Olle Bälter (KTH)

Abstract

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nunc in urna ipsum. Maecenas dignissim urna tristique ipsum tempus fringilla. Fusce dui metus, ultrices et condimentum vel, eleifend at elit. Suspendisse pellentesque dolor ac tellus egestas lobortis. Sed sed ipsum erat, sit amet aliquam velit. Donec tempus fermentum dui at convallis. Nulla facilisi. Integer nec neque sed quam auctor iaculis a consequat erat. Pellentesque ullamcorper dignissim arcu semper sollicitudin. Vestibulum ante ipsum primis in faucibus orci luctus et ultrices posuere cubilia Curae; Aenean hendrerit condimentum magna a mollis.

Nulla eget mi tellus, a convallis mauris. Suspendisse sit amet venenatis justo. Aliquam quis justo a dui faucibus euismod at blandit augue. Suspendisse potenti. Quisque ac sapien eleifend libero rhoncus vestibulum eu vel nibh. Praesent at massa in tortor ultrices ornare. In elementum fringilla felis, et blandit neque rhoncus et.

Acknowledgments

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nunc in urna ipsum. Maecenas dignissim urna tristique ipsum tempus fringilla. Fusce dui metus, ultrices et condimentum vel, eleifend at elit. Suspendisse pellentesque dolor ac tellus egestas lobortis. Sed sed ipsum erat, sit amet aliquam velit. Donec tempus fermentum dui at convallis. Nulla facilisi. Integer nec neque sed quam auctor iaculis a consequat erat. Pellentesque ullamcorper dignissim arcu semper sollicitudin. Vestibulum ante ipsum primis in faucibus orci luctus et ultrices posuere cubilia Curae; Aenean hendrerit condimentum magna a mollis.

Nulla eget mi tellus, a convallis mauris. Suspendisse sit amet venenatis justo. Aliquam quis justo a dui faucibus euismod at blandit augue. Suspendisse potenti. Quisque ac sapien eleifend libero rhoncus vestibulum eu vel nibh. Praesent at massa in tortor ultrices ornare. In elementum fringilla felis, et blandit neque rhoncus et.

Glossary

Term	**Description**
AJAX	Asynchronous JavaScript and XML, a technique used to let clients asynchronously do API-requests after the page has been rendered.
API	Application Programming Interface, an interface for a computer program.
Backbone.js	A lightweight Javascript framework.
CDN	Content Delivery Network, a network of servers that serves content with high availability and performance.
Crawler	A program that browses the web, used by search engines to index websites.
DOM	Document Object Model, a tree representation of the objects on a web page.
DOM-element	A node in the DOM-tree.
DOM-selector	A method of retrieving nodes from the DOM-tree.
Ember.js	A Javascript framework with two-way data-bindings.
GET parameter	A parameter sent as a GET request in HTTP.
HTML	Hypertext Markup Language, a markup language used to describe objects on a website
HTTP	Hypertext Transfer Protocol, an application protocol used to distribute websites in a network.
Hashbang	A name for the character sequence #!
jQuery	A popular Javascript library commonly used in web applications.
JSON	JavaScript Object Notation, a text-based standard to format data in a compact way.
MySQL	An open source relational database manager.
PHP	A server-side scripting language commonly used for web development.
PhantomJS	A headless Webkit browser.
REST	REpresentational State Transfer, a simple software architecture style for distributed systems. Commonly used for web services.
SEO	Search-Engine Optimization, when optimizing a website with the goal to get a higher page rank.
SOAP	Simple Object Access Protocol, a software architecture style for distributed systems.
SPA	Single-Page Application, a web application that fits all content on one single web page.
SPI	A synonym for SPA.
Truthy	A value that is evaluated as true in a logical context. In Javascript, any value except false, NaN, null, undefined, 0 and "".
URL	Uniform Resource Locator, a string that refers to an Internet resource.
XML	Extensible Markup Language, a text-based standard to format data.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Pellentesque et nisi nibh. Nunc augue risus, pellentesque ut egestas et, rhoncus ac arcu. Etiam nunc velit, gravida vel luctus a, ornare et est. Proin vehicula elementum libero, in faucibus velit condimentum vel. Vestibulum ante ipsum primis in faucibus orci luctus et ultrices posuere cubilia Curae; Ut tempus elit vitae quam pellentesque posuere. Nulla id mauris eget nisl vehicula faucibus. Aliquam viverra lacus ac neque fringilla sit amet bibendum orci vestibulum. Maecenas et turpis in dui dictum volutpat. Quisque in enim sit amet metus euismod fringilla consectetur non urna. Fusce vitae dui ut dolor lobortis dictum sed non metus. Vivamus rutrum nisi eget ante euismod volutpat dignissim sem tempor.

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nunc in urna ipsum. Maecenas dignissim urna tristique ipsum tempus fringilla. Fusce dui metus, ultrices et condimentum vel, eleifend at elit. Suspendisse pellentesque dolor ac tellus egestas lobortis. Sed sed ipsum erat, sit amet aliquam velit. Donec tempus fermentum dui at convallis. Nulla facilisi. Integer nec neque sed quam auctor iaculis a consequat erat. Pellentesque ullamcorper dignissim arcu semper sollicitudin. Vestibulum ante ipsum primis in faucibus orci luctus et ultrices posuere cubilia Curae; Aenean hendrerit condimentum magna a mollis.

Nulla eget mi tellus, a convallis mauris. Suspendisse sit amet venenatis justo. Aliquam quis justo a dui faucibus euismod at blandit augue. Suspendisse potenti. Quisque ac sapien eleifend libero rhoncus vestibulum eu vel nibh. Praesent at massa in tortor ultrices ornare. In elementum fringilla felis, et blandit neque rhoncus et.

Sed leo tellus, pulvinar a vehicula non, dapibus id justo. Fusce a sapien sapien, vitae lobortis ante. Morbi convallis libero sit amet sem auctor tristique. Curabitur ut porttitor nibh. Lorem ipsum dolor sit amet, consectetur adipiscing elit. Vestibulum id diam risus, id fermentum dui. Nullam malesuada commodo rhoncus. Duis eu hendrerit mauris. Aenean sodales nibh ac ante egestas quis ornare massa vehicula. Pellentesque sed mi ut sapien sodales tempor non vel enim.

Pellentesque et nisi nibh. Nunc augue risus, pellentesque ut egestas et, rhoncus ac arcu. Etiam nunc velit, gravida vel luctus a, ornare et est. Proin vehicula elementum libero, in faucibus velit condimentum vel. Vestibulum ante ipsum primis in faucibus orci luctus et ultrices posuere cubilia Curae; Ut tempus elit vitae quam pellentesque posuere. Nulla id mauris eget nisl vehicula faucibus. Aliquam viverra lacus ac neque fringilla sit amet bibendum orci vestibulum. Maecenas et turpis in dui dictum volutpat. Quisque in enim sit amet metus euismod fringilla consectetur non urna. Fusce vitae dui ut dolor lobortis dictum sed non metus. Vivamus rutrum nisi eget ante euismod volutpat dignissim sem tempor.

Proin aliquam tristique aliquam. Donec tincidunt ullamcorper massa, in ultrices velit iaculis et. Quisque malesuada tortor non nisi suscipit eu lacinia felis elementum. Cras eu libero quis risus porta sodales. Morbi elit est, vestibulum in tempus eget, hendrerit ut eros. Nam sed pharetra risus. Praesent commodo justo id neque pulvinar nec elementum lorem gravida. Cras egestas malesuada dolor, id auctor mauris interdum sed.

Pellentesque et nisi nibh. Nunc augue risus, pellentesque ut egestas et, rhoncus ac arcu. Etiam nunc velit, gravida vel luctus a, ornare et est. Proin vehicula elementum libero, in faucibus velit condimentum vel. Vestibulum ante ipsum primis in faucibus orci luctus et ultrices posuere cubilia Curae; Ut tempus elit vitae quam pellentesque posuere. Nulla id mauris eget nisl vehicula faucibus. Aliquam viverra lacus ac neque fringilla sit amet bibendum orci vestibulum. Maecenas et turpis in dui dictum volutpat. Quisque in enim sit amet metus euismod fringilla consectetur non urna. Fusce vitae dui ut dolor lobortis dictum sed non metus. Vivamus rutrum nisi eget ante euismod volutpat dignissim sem tempor.

Proin aliquam tristique aliquam. Donec tincidunt ullamcorper massa, in ultrices velit iaculis et. Quisque malesuada tortor non nisi suscipit eu lacinia felis elementum. Cras eu libero quis risus porta sodales. Morbi elit est, vestibulum in tempus eget, hendrerit ut eros. Nam sed pharetra risus. Praesent commodo justo id neque pulvinar nec elementum lorem gravida. Cras egestas malesuada dolor, id auctor mauris interdum sed.

Fusce eget tincidunt nibh. Quisque sed velit felis. Suspendisse in sagittis lorem. Fusce posuere ipsum id leo facilisis sit amet sollicitudin enim egestas. In augue massa, tincidunt ac blandit faucibus, ultricies nec tortor. Fusce et dui diam. Cras nec eros id dui laoreet lacinia. Proin malesuada est vel erat pharetra rutrum faucibus nulla porta.

Curabitur dui mi, consequat et varius sed, laoreet id neque. Fusce ac ante non ligula porta viverra vitae a quam. Morbi eget nulla sit amet nunc vehicula blandit id et nibh. Duis est arcu, lobortis a lobortis non, adipiscing id ante. Etiam ut massa sit amet mi malesuada facilisis vitae nec neque. Nulla sodales massa metus. Morbi molestie faucibus velit, nec elementum enim vehicula vel. Vivamus ut risus pellentesque risus vehicula ornare. Integer quis neque lobortis libero tempor consequat ut vitae tortor. Maecenas pulvinar laoreet rhoncus.

The rest of the thesis is structured as follows:

- Chapter two describes the technical background behind SPAs.
- Chapter three discusses design patterns commonly used in graphical interfaces.
- Chapter four presents the architecture behind the framework and how search engine optimization problems can be approached.
- Chapter five describes the implementation of the framework.
- Chapter six presents measurements of the framework regarding testability and initial loading time. It also includes comparisons with other frameworks on the market.
- The seventh and last chapter discusses when SPAs are suitable to use and how the future of the framework might look like.

Chapter 2

Technical background

2.1 Single-page applications

As previously described SPAs introduce new demands regarding architecture and structure on the client side. To understand how these problems can be approached it is important to understand the basics behind how a single-page application works. There is yet no formal definition that describes what a SPA is. Many developers have their own view of what it exactly means, but Ali Mesbah and Arie van Deursen [4] have stated a quite clear definition in their paper about SPAs:

> "The single-page web interface is composed of individual components which can be updated/replaced independently, so that the entire page does not need to be reloaded on each user action."

The definition includes a number of key attributes that help to define a SPA:

- **Web interface** - Used on the web, focuses on user interfaces.
- **Individual components** - It's divided into smaller components that coop with each other.
- **Updates and replaces** - A component can at any time be changed or replaced with another component.
- **Reloading** - The entire page is never reloaded even though new content may be loaded into some sections.
- **User actions** - The SPA is responsible for handling user actions such as input from mouse and keyboard.

When running a Javascript SPA the browser first makes a request to the web server. The web server will then respond with the Javascript client including the resources needed to run it. Once the client is transferred it will be initialized and it will be ready to be run. When the user interacts with the client, such as clicking on graphical elements, this may require new data to be fetched from the server. Instead of reloading the entire page the client will instead make a small request to the API on the web server. The API is simply an interface that allows the clients to communicate with the server in a data format that is easy to understand for both the client and server [5]. A typical flow of communication for a SPA can be seen in figure 2.1.

2.2 URLs, routes and states

The URL structure of a website is an fundamental aspect to consider. In order to provide linkable and shareable URLs it is important that the URL is unique for the application's state. If the same URL would be used for several states they would not be accessible directly from the address bar, which would become impractical if a user stores or shares the current URL. The Javascript application needs to be able to map a URL into a certain state, this

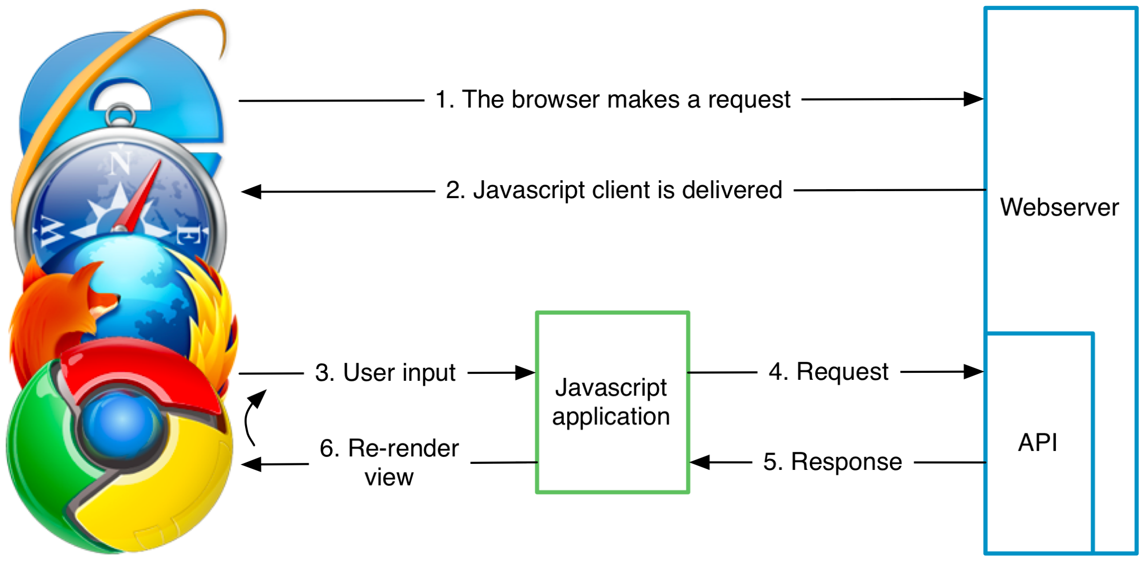

Figure 2.1: The flow of communication between a browser and the web server.

is often done by a routing component. It takes the URL that the user provides and decodes it into a state so that the correct components can be initialized and presented to the user. The routing component also needs to be able to update the URL if the application's state is changed. The HTML5 draft provides a history API to allow a Javascript client to change the URL during runtime without reloading the page [14]. To support older browsers that do not support the history API the hashbang technique can instead be used [15]. The idea is to add *#!/state* to the URL, the characters that comes after the hashmark (#) will not be sent to the server and can instead be used by the Javascript client to detect a certain state of the application.

Both the HTML5 history API and the hashbang technique will keep the back-button behavior intact. When the user hits the back-button in the browser the URL will simply change to the last visited URL. The Javascript application will recognize a change of URL and will load the content once again. By having a system like this it becomes possible to use URLs just as if the pages were rendered by a backend system instead by a Javascript application.

2.3 API communication

In order for the client to dynamically fetch data from the server an API is needed in between. The API describes how data is being transferred, what data format is being used and what methods are available.

One of the most common ways of letting the client asynchronously communicate with the server is via AJAX (Asynchronous Javascript and XML). AJAX allows the client to asynchronously fetch and post data to the server after the page has been loaded. However, AJAX has no native support for letting the server push data to the client. The client must always initialize a new connection and then poll the server for new data [16, p. 491]. To enable the server to push data to the client other techniques can instead be used, e.g. the websocket API in the HTML5 draft which supports bi-directional communication. This will ease the load of the server and allows a faster and more responsive system.

When using a HTTP-based data transportation some kind of architectural structure is needed. The architecture describes how messages are formed so that both parties understand each other, e.g. what methods that are available and how data is transferred. It also often includes additional information about how authentication is handled, as well as the flow of communication. Common architectural structures are REST and SOAP.

The data format of a message describes the syntactical form of the objects that are transferred. Since HTTP messages are based on ASCII-characters [20] it is also commonly used when describing data objects. Common data formats used today are JSON, XML and plain text [5].

2.4 Templates

In order to separate the presentation layer from the business logic, templates are often used. A template engine compiles HTML from a description written in a template language together with a set of data, it's commonly used for rendering the views in a Javascript application. When the data is changed the view can simply be re-rendered so that the user sees the correct version of it. A template engine is however a quite complex system, to generate HTML from a description an interpretation of the template language is required. Since these interpretations are done during runtime the performance is a crucial aspect to consider. Some frameworks have their own template engines while others use generic libraries in order to support template rendering.

2.5 Data bindings

Data bindings are used to bind a set of data to a corresponding view. One-way data bindings allow a view to be automatically re-rendered when its data is changed. Some frameworks, like Angular JS, support a two-way data bindings that also allow the data to be updated corresponding to changes in the view [21]. This is quite a powerful technique since all logic behind a view now is managed by the framework which makes it easier for the developer to get things working. Important to note is that no logic or complexity is removed, it is just moved to the framework. However, a framework must support a wide range of HTML-elements even though all of these are not used in the application. This often affects the performance of the application and two-way data bindings can sometimes become overkill, especially for smaller applications.

2.6 Testing a Javascript application

Testing comes in many forms and shapes during a project. During the development phase unit tests are commonly used to drive the development. It's used as a tool by the developers to ensure that all the functionality is kept intact. The use of a testdriven approach to develop software is becoming more and more popular [9]. However, developing Javascript applications with a testdriven approach can be tricky. The functionality behind a Javascript application tends to be tightly coupled with the user interface. This is simply the case since one of the most essential purposes of a Javascript applications is to interact with the user, which of course is done via the user interface. To write unit tests that interacts with the user interface is not trivial. It requires the unit test to trigger events on DOM-elements and then traverse the DOM and validate its content. If a view is even slightly changed this can break the test even if an acceptance test would still pass. Some architectures allow developers to instead write unit tests for what is underneath the graphical interface, making the tests easier to write and the result easier to validate. In the end unit testing will only ensure that the code is doing the right thing, not that the system is working as expected. High-level testing, such as acceptance testing, will always be needed to validate the functionality from a user perspective [10].

2.7 Search engine optimization

Search engine optimization is an important aspect of today's web services. Search engines such as Google, Bing and Yahoo indexes webpages every day to make it easier for users to find what they are looking for. In order to get a high page rank it is important that the search engines can index the content of the web service to understand what it is about.

The indexing is done by a crawler. The crawler downloads a snapshot of the webpage and analyzes the HTML to find keywords, links and more. Even though the crawler itself is quite a complex system the support for Javascript is limited. Google crawler won't run any Javascript at all [6], even though there are other reports stating that it actually can run a limited amount of Javascript [17]. Nevertheless, the limited support is a major issue when building Javascript-based sites that are to be indexed by a crawler.

In terms of the crawlers from Google, Bing and Yahoo is this solved by translating the URLs for AJAX-based services, this allows the webserver to serve static HTML snapshots instead of dynamic Javascript versions. When the crawler visits a website that has an URL that contains a hashbang (#!) the site will be treated as an AJAX-based site. All characters after the hashbang are then sent as the GET parameter *_escaped_fragment_* [18]. The reason for this is due to the HTTP specification stating that characters after a hashmark should not be included in the request to the webserver [19]. The webserver can check if the GET parameter *_escaped_fragment_* is set, if so is the client probably a crawler and a static HTML version is to be served. Figure 2.2 shows how a typical crawler treats AJAX-based websites.

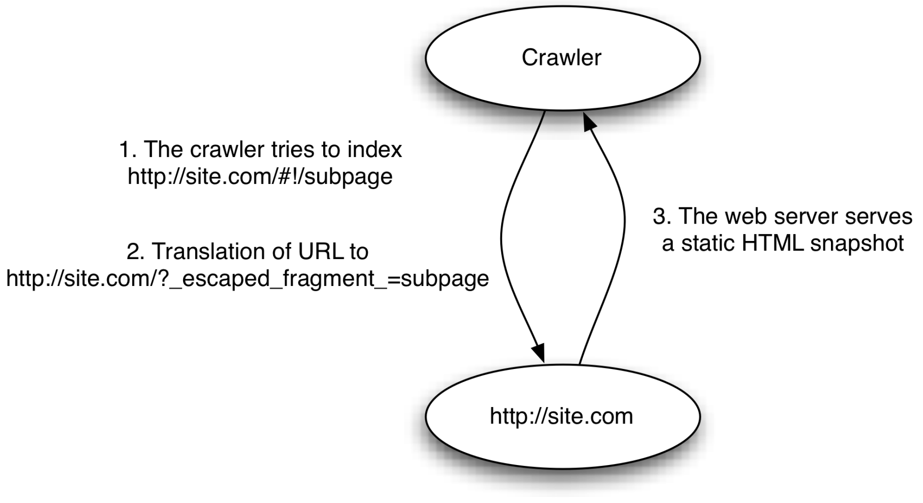
Crawler
1. The crawler tries to index http://site.com/#!/subpage
2. Translation of URL to http://site.com/?_escaped_fragment_=subpage
3. The web server serves a static HTML snapshot
http://site.com

Figure 2.2: The flow of how a typical crawler communicates with an AJAX-based website.

Chapter 3

Discussion

7.1 Test-driven development in Javascript

Sed leo tellus, pulvinar a vehicula non, dapibus id justo. Fusce a sapien sapien, vitae lobortis ante. Morbi convallis libero sit amet sem auctor tristique. Curabitur ut porttitor nibh. Lorem ipsum dolor sit amet, consectetur adipiscing elit. Vestibulum id diam risus, id fermentum dui. Nullam malesuada commodo rhoncus. Duis eu hendrerit mauris. Aenean sodales nibh ac ante egestas quis ornare massa vehicula. Pellentesque sed mi ut sapien sodales tempor non vel enim.

7.2 The importance of delays

Sed leo tellus, pulvinar a vehicula non, dapibus id justo. Fusce a sapien sapien, vitae lobortis ante. Morbi convallis libero sit amet sem auctor tristique. Curabitur ut porttitor nibh. Lorem ipsum dolor sit amet, consectetur adipiscing elit. Vestibulum id diam risus, id fermentum dui. Nullam malesuada commodo rhoncus. Duis eu hendrerit mauris. Aenean sodales nibh ac ante egestas quis ornare massa vehicula. Pellentesque sed mi ut sapien sodales tempor non vel enim.

7.3 When to use SPAs

Sed leo tellus, pulvinar a vehicula non, dapibus id justo. Fusce a sapien sapien, vitae lobortis ante. Morbi convallis libero sit amet sem auctor tristique. Curabitur ut porttitor nibh. Lorem ipsum dolor sit amet, consectetur adipiscing elit. Vestibulum id diam risus, id fermentum dui. Nullam malesuada commodo rhoncus. Duis eu hendrerit mauris. Aenean sodales nibh ac ante egestas quis ornare massa vehicula. Pellentesque sed mi ut sapien sodales tempor non vel enim.

Sed leo tellus, pulvinar a vehicula non, dapibus id justo. Fusce a sapien sapien, vitae lobortis ante. Morbi convallis libero sit amet sem auctor tristique. Curabitur ut porttitor nibh. Lorem ipsum dolor sit amet, consectetur adipiscing elit. Vestibulum id diam risus, id fermentum dui. Nullam malesuada commodo rhoncus. Duis eu hendrerit mauris. Aenean sodales nibh ac ante egestas quis ornare massa vehicula. Pellentesque sed mi ut sapien sodales tempor non vel enim.

7.4 Future work

Pellentesque et nisi nibh. Nunc augue risus, pellentesque ut egestas et, rhoncus ac arcu. Etiam nunc velit, gravida vel luctus a, ornare et est. Proin vehicula elementum libero, in faucibus velit condimentum vel. Vestibulum ante ipsum primis in faucibus orci luctus et ultrices posuere cubilia Curae; Ut tempus elit vitae quam pellentesque posuere. Nulla id mauris eget nisl vehicula faucibus. Aliquam viverra lacus ac neque fringilla sit amet bibendum orci vestibulum. Maecenas et turpis in dui dictum volutpat. Quisque in enim sit amet metus euismod fringilla consectetur non urna. Fusce vitae dui ut dolor lobortis dictum sed non metus. Vivamus rutrum nisi eget ante euismod volutpat dignissim sem tempor.

Proin aliquam tristique aliquam. Donec tincidunt ullamcorper massa, in ultrices velit iaculis et. Quisque malesuada tortor non nisi suscipit eu lacinia felis elementum. Cras eu libero quis risus porta sodales. Morbi elit est, vestibulum in tempus eget, hendrerit ut eros. Nam sed pharetra risus. Praesent commodo justo id neque pulvinar nec elementum lorem gravida. Cras egestas malesuada dolor, id auctor mauris interdum sed.

Bibliography

[1] Dan Webb, Twitter, 2012-05-29, *Twitter engineering blog* http://engineering.twitter.com/2012/05/improving-performance-on-twittercom.html [2012-06-11]

[2] Sean Work, 2011, *How loading time affects your bottom line* http://blog.kissmetrics.com/loading-time/ [2012-07-06]

[3] Nicholas C. Zakas, Yahoo, 2010-10-13, *How many users have JavaScript disabled?* http://developer.yahoo.com/blogs/ydn/posts/2010/10/how-many-users-have-javascript-disabled [2012-08-15]

[4] Ali Mesbah, Arie van Deursen, Delft University of Technology, 2006, *Migrating Multi-page Web Applications to Single-page AJAX Interfaces, 2nd revision*, Software Engineering Research Group, Delft University of Technology

[5] Valerio De Lucaa, Italo EpicocoDaniele Lezzi, Giovanni Aloisio, 2012, *GRB_ WAPI, a RESTful Framework for Grid Portals*, p. 2, DOI: 10.1016/j.procs.2012.04.049

[6] Google, 2012-02-17, *Making AJAX Applications Crawlable - Learn more*, What the user sees, what the crawler sees https://developers.google.com/webmasters/ajax-crawling/docs/learn-more [2012-06-28]

[7] Tom Dale, Yehuda Katz, 2011-03-01, *Sproutcore, Getting started* http://guides.sproutcore.com/getting_started.html [2012-06-25]

[8] Angular JS, *Angular JS - What is a Module?* http://docs.angularjs.org/guide/module [2012-06-25]

[9] Ron Jeffries, Grigori Melnik, 2007, *TDD: The art of fearless programming*, p. 29

[10] Roy W. Miller, Christopher T. Collins, 2001 *Acceptance testing*, p. 1, XP Universe

[11] Addy Osmani, 2011, *Writing Modular JavaScript With AMD, CommonJS & ES Harmony* http://addyosmani.com/writing-modular-js/ [2012-06-25]

[12] Addy Osmani, 2012, *Learning JavaScript Design Patterns*, O'Reilly media, Inc.

[13] Ross Harmes, Dustin Diaz, 2008, *Pro Javascript Design Patterns*, p. 12, Springer-verlag New York, Inc.

[14] W3C, 2012-03-29, *HTML5 - A vocabulary and associated APIs for HTML and XHTML* http://www.w3.org/TR/2012/WD-html5-20120329/history.html#history [2012-06-25]

[15] Matthew MacDonald, 2011, *HTML5 - The missing manual*, p. 373, O'Reilly media, Inc.

[16] David Fanagan, 2011, *Javascript - The definitive guide, 6th edition*, O'Reilly media, Inc.

[17] Stone Tapes, 2012-05-25, *Google now searches Javacript* http://www.i-programmer.info/news/81-web-general/4248-google-now-searches-javascript.html [2012-07-10]

[18] Google, 2012-02-17, *Making AJAX Applications Crawlable - Full specification* https://developers.google.com/webmasters/ajax-crawling/docs/specification [2012-06-28]

[19] T. Berners-Lee, 1994, *RFC 1630*, p. 13 http://www.ietf.org/rfc/rfc1630.txt [2012-06-28]

[20] R. Fielding, UC Irvine, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, T. Berners-Lee, 1999, *RFC 2616*, p. 31 http://www.ietf.org/rfc/rfc2616.txt [2012-06-28]

[21] Angular JS, *Angular JS - Data Binding in Angular* http://docs.angularjs.org/guide/dev_guide.templates.databinding [2012-07-02]

[22] Erich Gamma, Richard Helm, Ralph Johnson, John Vlissides, 1994, *Design Patterns - Elements of Reusable Object-oriented Software*, Addison-Wesley.

[23] Eric Freeman, Elisabeth Freeman, 2004, *Head first - Design patterns*, p. 530, O'Reilly media, Inc.

[24] John Smith, 2009, *WPF Apps With The Model-View-ViewModel Design Pattern* http://msdn.microsoft.com/en-us/magazine/dd419663.aspx [2012-07-05]

[25] Addy Osmani, 2012, *Understanding MVVM - A guide for Javascript developers* http://addyosmani.com/blog/understanding-mvvm-a-guide-for-javascript-developers/ [2012-08-30]

[26] Davy Brion, 2010-07-21, *The MVVM Pattern is highly overrated* http://davybrion.com/blog/2010/07/the-mvvm-pattern-is-highly-overrated/ [2012-07-05]

[27] Mustache, *Mustache - Logic-less templates* http://mustache.github.com/ [2012-07-06]

[28] Behrouz A. Forouzan, 2010, *TCP/IP Protocol Suite, fourth edition*, Connection establishment, p. 442, McGraw-Hill Companies, Inc.

[29] Behrouz A. Forouzan, 2010, *TCP/IP Protocol Suite, fourth edition*, Congestion control, p. 473, McGraw-Hill Companies, Inc.

[30] Nicholas Zakas, 2012, *Maintainable Javascript*, p. 145, O'Reilly media, Inc.

[31] Steve Souders, 2007, *High Performance Web Sites: Essential Knowledge for Front-End Engineers*, p. 31, O'Reilly media, Inc.

[32] L. Peter Deutsch, 1996, *RFC 1951*, p. 13 http://www.ietf.org/rfc/rfc1951.txt [2012-08-30]

[33] Addy Osmani, Sindre Sorhus *TodoMVC* http://addyosmani.github.com/todomvc/ [2012-08-29]

[34] John Resig, 2008-03-20, *Simple JavaScript Inheritance*
http://ejohn.org/blog/simple-javascript-inheritance/ [2012-07-30]

[35] Ian Hickson, W3, 2012-08-09, *The Websocket API*
http://www.w3.org/TR/2012/WD-websockets-20120809 [2012-08-15]

[36] Douglas Crockford, 2010-11-18, *JSON in Javascript*
https://github.com/douglascrockford/JSON-js/ [2012-08-17]

[37] Nina Bhatti, Anna Bouch, Allan Kuchinsky, 2000, *Integrating user-perceived quality into Web server design*, Elsevier Science B.V.