

# **Ember.js front-end framework – SEO challenges and frameworks comparison**

Sunil Shrestha

Bachelor's Thesis  
Degree Program in BITe  
October 2015



## Abstract

<b>Author</b> Sunil Shrestha	<b>Group</b> BITE10S
<b>Title of report</b> Ember.js front-end framework – SEO challenges and frameworks comparison	<b>Report pages</b> 43+4
<b>Supervisor</b> Juhani Välimäki	
<p>IWA Labs Oy, a Finnish company with extensive experience in modern information technology provides professional service in Search Engine Optimization (SEO), online marketing as well as develop mobile and web applications for its clients. In order to provide smooth and better user experience with web applications, the company has adapted front-end dedicated frameworks such as AngularJS, Backbone.js, etc. Therefore, the company is interested in Ember.js– another emerging front-end framework that was first released in 2011.</p> <p>The paper aims to study over current development of Ember.js and compare it with alternative frameworks based on community popularity and their core features. Furthermore, it also addresses some of the most common SEO challenges from Ember.js standpoint.</p> <p>The comparison of frameworks is based on various metrics of each framework such as GitHub stars, commits history, available plugins, Stack Overflow questions, Google trend over time etc. Additionally, core features are also discussed in terms of Model-View-Controller (MVC) pattern. On the other hand, possible solutions for most common SEO challenges in Ember.js are studied and tested via a sample application.</p> <p>Based on the community popularity, AngularJS is clearly proven to be the most popular framework. Nonetheless, based on GitHub commits and other metrics, Ember.js can be taken equally an active and fast growing project. Ember.js community seems to be aware of SEO challenges in Single Page Application (SPA). An ember ad-on, ember-cli-fastboot is actively developed by Ember.js core team to make the application crawlable by search engines and improve performance. However, the ad-on is not ready for production yet and prerender.io service is used for our sample application to get it crawlable by search engines.</p>	
<b>Keywords</b> Ember.js, Front-end frameworks, Front-end JavaScript frameworks, Front-end frameworks comparisons, SEO challenges in Ember.js	

# Table of Contents

Abstract .....	i
Table of Contents .....	ii
Abbreviations and terms.....	iv
1 Introduction.....	1
1.1 Scope of the study.....	1
2 Study plan.....	2
2.1 Background theory.....	2
2.2 Objectives for the research.....	2
2.3 Research.....	2
2.4 Conclusions and Evaluation.....	2
2.5 Summary.....	2
3 Background theory.....	3
3.1 Web Application Framework (WAF) .....	4
3.2 MVC Pattern.....	5
3.3 Ember.js.....	8
3.4 Search Engine Optimization (SEO).....	10
4 Objectives for the research.....	11
4.1 Comparing Ember.js with similar frameworks.....	11
4.2 Addressing SEO challenges from Ember.js standpoint .....	11
5 Research.....	13
5.1 Comparing Ember.js to similar front-end frameworks on the market.....	13
5.2 SEO challenges from Ember.js standpoint .....	24
5.3 Applying SEO challenges solutions to sample application .....	35
6 Conclusions and Evaluation.....	38
6.1 Comparing results against the goals .....	39
6.2 Validity of the results.....	40
6.3 Evaluating used information sources.....	40
6.4 Learning during the thesis process .....	41
6.5 Further research .....	41
7 Summary.....	43

Appendix A – Landing page of sample application running at server .....	i
Appendix B – Google search result for sample application.....	ii
Appendix C – HTML document with Meta tags for each resource (AngularJS). .....	iii
Appendix D – HTML document with Meta tags for each resource (Backbone.js). .....	iv

## Abbreviations and terms

Terms	Description
AJAX	Asynchronous JavaScript and XML
API	Application Program Interface
App	Application
Assets	CSS, JavaScript, Images
Front-end frameworks	Framework dedicated to manage front-end components
Frontend JavaScript Framework	Front-end framework built upon JavaScript
Gist	Small piece of code or snippets hosted in GitHub for easy sharing
GitHub	Repositories hosting service where people can share and collaborate easily
HTML	Hyper Text Markup Language
JavaScript	Dynamic programming language also known as ECMAScript
JS	JavaScript
JSON	JavaScript Object Notation
MVC/MV*	Model-View-Controller / Model-View with optional components
Nginx	Open source HTTP server and reverse proxy, as well as an IMAP/POP3 proxy server
REST	Representational State Transfer is an architectural style, and an approach to communications that is often used in the development of Web services.
URL	Uniform Resource Identifier
WAF	Web Application Framework
XML	Extensible Markup Language

# 1 Introduction

With the advancement of information technology, the web technology has also evolved remarkably since early 1990s, when the Internet and World Wide Web (WWW) were publicly introduced. As the web applications have become the most common platform for sharing information, it is also very important to present content to users with intuitive design, great experience and smooth interactivity. This creates the need of proper client-side dedicated frameworks like AngularJS, Ember.js, Backbone.js etc. These frameworks offer not only great user interactivity, but also proper conventions, tools to keep the codes organized, as well as make web application development more efficient.

Iwa Labs Oy, a Finnish company focusing on human centric digital solutions, holds an extensive experience in developing popular web-services. While being close to modern web development technology, the company provides professional services on designing business models and service concepts, search engine optimization (SEO), online advertising, e-commerce and usage analytics. (Iwa Labs Oy, 2015). In order to serve clients with modern web technology, Iwa Labs has felt the need of study over current trend of front-end JavaScript frameworks on the Internet and see where Ember.js stands. Hence, this paper aims to compare currently available front-end frameworks and study where Ember.js stands. Additionally, we will also address some SEO challenges from Ember.js standpoint, find solution, and apply solutions to a sample application.

## 1.1 Scope of the study

The objective of this paper is to provide an overview on Ember.js, framework architecture as well as compare it with other alternative frameworks in terms of community popularity and their core features. The paper will also address Search Engine Optimization (SEO) in Ember.js, find solutions to SEO challenges, and test them via a sample application. However, it cannot be considered as a complete guide to SEO. As this paper will present some snippets as examples, readers are expected to be familiar with Ember.js to grasp the solutions for SEO challenges thoroughly.

## **2 Study plan**

The study in this paper consists of five different phases: background theory, objectives for research, research and, conclusions and summary.

### **2.1 Background theory**

This section will familiarize readers with the general idea and concepts around Ember.js such as front-end, backend, SEO, and other theoretical topics that will be required to understand the context and significance of content we will address ahead.

### **2.2 Objectives for the research**

In this section, we will clarify our main objective for research and the need for it. In addition, this section will also explain the means and methods that will be used to conduct the research and achieve our goals.

### **2.3 Research**

This section will address the objectives of the paper and will involve detailed research for our objectives. Here, we will collect data from different sources and find possible solutions and present the findings to readers. In addition, solutions to SEO challenges will be tested via a sample application. By the end of this section, the objectives of the study will be resolved and readers will have proper understanding on solution to research question.

### **2.4 Conclusions and Evaluation**

In this section, we will compare the result of study against our initial goal, discuss about the validity of our results and evaluate the sources from where we have extracted our solutions. In addition, readers will also be presented with uncovered topics as well as further research and development after this study.

### **2.5 Summary**

In this section, we will summarize the study in short and end the study part.

### 3 Background theory

The concept of web application is evolving continuously and demand for a better user experience has slowly reshaped its philosophy. Ember.js on the other hand is a front-end dedicated JavaScript framework contributed by well skilled open source community and holds some key concepts of a modern web technology. Hence, in this section, we will get into some of the theoretical topics and key concepts of web application that will help us to understand the context and significance of content we will address ahead.

As of a traditional web application, the basic structure of the application consists front-end and backend which keeps the development and maintenance process simpler. The figure below illustrates the basic components of a web application.

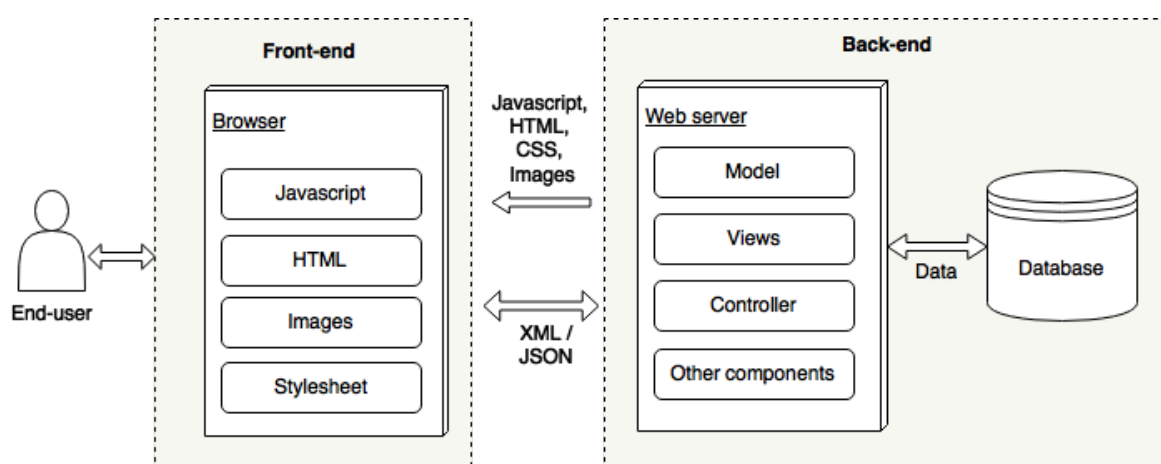


Figure 3.1 Basic components of a traditional web application

As shown in figure 3.1, the user interacts with the web application via browser where JavaScript is run and HTML, CSS, images, etc. are rendered. This presentation layer is known as front-end as well as client side. On the other hand, backend comprises of server with application, database and other components that stay behind the browser listening to users' requests. Backend handles the users' requests, apply business logics and serves HTML and other required assets. Hence, backend codes stay on the server side where as front-end acts as an interface between user and backend. (3nytechnology.com, 2015)



### 3.1 Web Application Framework (WAF)

“A web application framework (WAF) is a software framework that is designed to support the development of dynamic websites, web applications, web services and web resources. The framework aims to alleviate the overhead associated with common activities performed in web development.” (Wikipedia, 2015f)

As stated by Wikipedia, WAF is designed to reduce the complexity in application development. There are many frameworks, which are designed to handle the backend as well as render views for the front-end. Frameworks like Rails can handle different type of requests from clients and respond with appropriate format such as HTML, XML, and JSON, etc. Table below presents some of the famous web application Frameworks collected from Wikipedia. (Wikipedia, 2015a)

<b>Frameworks</b>	<b>Programming language</b>	<b>Runs at</b>
Laravel	PHP	Backend
Spring MVC	Java	Backend
Rails (Ruby on rails)	Ruby	Backend
Django	Python	Backend
Node.js	JavaScript	Backend
ASP.NET	C# (C Sharp)	Backend
AngularJS	JavaScript	Front-end
Ember.js	JavaScript	Front-end

Table 3.1 List of Web Application Frameworks (WAF)

The backend frameworks reside on the server whereas front-end frameworks run at the end-users' browser and takes care of the presentation of content and user interactions. As we have full control on the backend servers, it is common to have security layers and business layer on the backend and serve only assets and data to the front-end

## 3.2 MVC Pattern

Most of the famous web application frameworks follow the Model-view-controller (MVC) architectural pattern, which separates the data model with business rules from user interface. MVC pattern is sometimes also denoted with MV\* where asterisk (\*) could be an optional component. In addition, this concept helps to modularize the codes and allow us to render different views or interface for different type of requests. (Pastor, 2010) For example: we can render fully styled page for human requests; only JSON data for API requests or we can render sitemap.xml in XML format for crawlers like Googlebot. Some of the examples MVC based frameworks as follows. (Wikipedia, 2015a)

- Spring MVC built with Java
- Rails (Ruby on rails) built with Ruby
- Ember.js built with JavaScript

### Front-end frameworks

Front-end frameworks are meant to support the client-side presentation and maintain the front-end related codes and organized them. The term “Front-end frameworks” may refer to design oriented frameworks such as bootstrap, foundation, semantic UI etc. as well as the same term could meant the full-fledged JavaScript frameworks like Angular.js, Ember.js etc. Design oriented frameworks like bootstrap provide easy and more standards-compliant web-design along with different tool and modules like reset-stylesheet, grid and responsive web-design, styling for forms, buttons and navigation menus etc. On the other hand, full-fledged JavaScript frameworks are meant to handle JavaScript related interactivity in application by keeping them structured and organized. Most of the famous JavaScript frameworks follow the MVC pattern, which isolates the data model, user interface and business logic into different layers as Model-View-Controller. This makes the MVC pattern capable of controlling user requests and handle interaction to servers more intelligently and logically.

### 3.2.1 How do we know if we should change to JS MVC Frameworks?

For years, we have been using jQuery and other libraries. One might think, it was okay then why should we change to frameworks now? Sometimes, adapting such big frameworks for a simple application could be too heavy and overkill the application. However, we should always analyze the required resources, need of framework and competitive advantage of adapting them. These JavaScript MVC frameworks would be beneficial for sites such as Stock Market Ticker, Social networking app, Data visualizations apps etc. that involves maximum user interaction. We should use them under following circumstances (Kukreja, 2014).

- In an application with use of heavy AJAX interaction
- Same data is being rendered in various ways on the page
- Presence of highly reusable visual elements. E.g. Like, Comment or Share links on Facebook, which appears in every single Facebook post
- Page contains many but trivial interactions. E.g. Like buttons
- The server can serve JSON/XML bases interface

Sites with news, blog pages, weather forecasts etc. rarely have that much of interactivity on the page. So, sites of these typical types are not recommended or should at least consider following points before undertaking any frameworks.

- The server is not capable of handling too many requests. If we use JS MVC, each model operation will cost a server call and server will need to respond to several small requests.
- The application has minimum user interactions. For example a page with form, that has a "Submit" and a "Delete" button only.
- Data on the application does not change very often or is a static web app.

### 3.2.2 Single page application

A single page application is a trending approach of building web application where all necessary code or assets such as HTML, JavaScript, CSS etc. are retrieved on user's browser with a single page load or the appropriate resources are dynamically loaded and added to the page as necessary in response to user actions. (Wikipedia, 2015e). Figure below shows a typical example of a single page application built with Ember.js

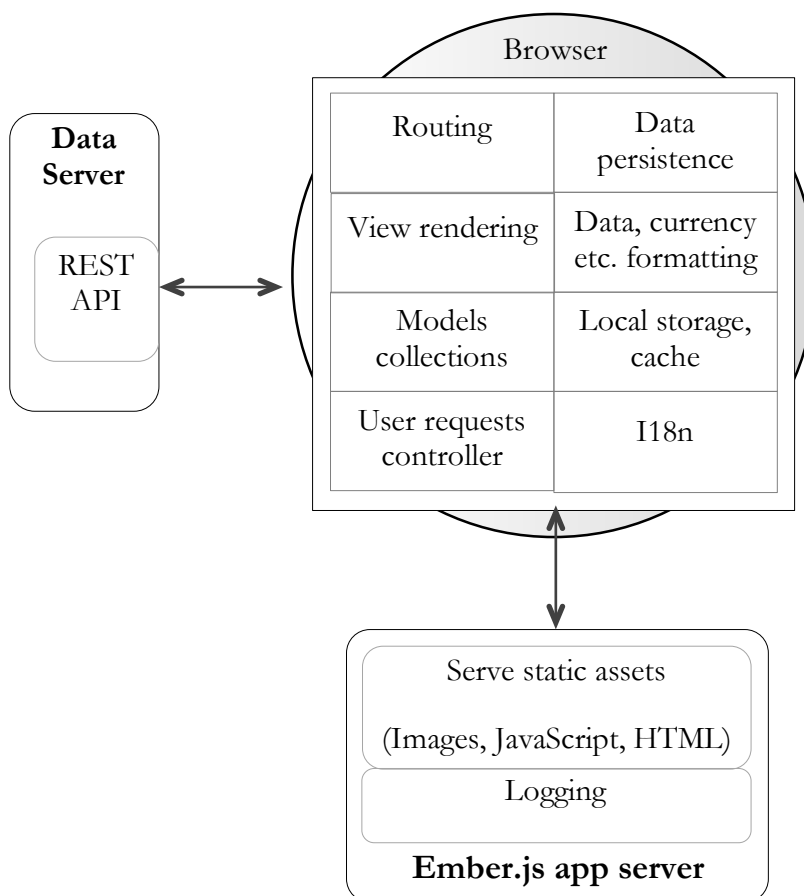


Figure 3.2 Ember.js application communicating with data server and application server. Adapted from Brehm's diagram (Brehm, 2012)

As shown in the figure 3.2, Ember.js application runs almost entirely in the browser as a single-page app. Routing, views rendering, model layer, handling user events, persistence, etc. all run in the client. The application is loosely coupled to the app server and data server, which may or may not be running on the same server. In this case, they are running in different servers. App server mostly serves static files whereas data server serves data in JSON or XML format via some RESTful API.

This approach is gaining popularity with Ember.js as well as with other similar frameworks like angular, backbone etc. However, this approach has two major drawbacks: performance and SEO. The performance flaws comes from the fact that the browser needs to download and evaluate all external and inline scripts before the browser can render any meaningful HTML. SEO is a vital aspect for content-heavy sites to be visible on Internet and absolutely need to allow search engines to crawl our application. But, keeping the application SEO friendly and crawlable has been big challenge while building single-page app like this. (Brehm, 2012)

### 3.3 Ember.js

‘Ember.js’ an MVC framework for creating ambitious web applications. Ember.js is one of the most famous and emerging open sourced JavaScript front-end frameworks with wide range of developers’ community. Many popular sites including Yahoo, Groupon, Discourse etc. currently use Ember.js. The framework is well designed to develop scalable single page application where all the templates and views resides within a page and takes care of rendering appropriate template based on state of URL or route. More than anything else, it provides a rich object model, declarative two way data binding, computed properties, HTMLBars templating engine (previously known as Handlebars.js) to automatically-updating views and a router for managing application state. (Ember.js, 2015a)

#### 3.3.1 Basic Concept

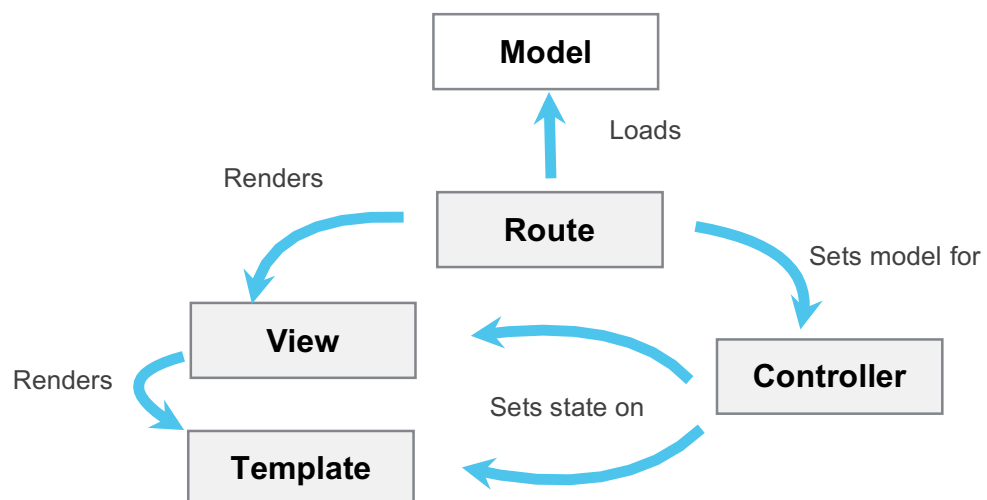


Figure 3.3 Ember.js architectural design. Adapted from Knebel’s diagram ( Knebel, 2013).

Ember emphasizes strongly on convention over configuration to incorporate common idioms and best practices among the developers' community. Figure 3.3 shows the architectural design and core concept of Ember.js app following the MVC pattern.

## **Router**

The router is a powerful concept of Ember, emphasizing the importance of the URL in managing application state. Ember also has route, which is different from the router. A route object corresponds to a URL and essentially serializes the application's current state whereas the router is just a synopsis of all of our routes. Tom Dale one of the lead developers of Ember quoted as follows emphasizing importance of URL.

“The URL is an important strength that the web has over native apps. In web apps, the URL goes beyond just being a static reference to the location of a document on a server. The best way to think of it is as the serialization of the application's current state. As the user interacts with the application, the URL should update to reflect what the user is seeing on their screen.” (Dale, 2012)

## **Model**

Every route has an associated model, containing the data associated with the current state of the application. Models may be retrieved and updated by implementing AJAX callbacks inside each route, or we can also rely on Ember-Data (a data-storage abstraction layer) to greatly simplify the retrieval, updating and persistence of models over a REST API. Despite of all great features in ember.js, it is important to realize that Ember-Data in rapid development and its good idea to keep an eye on its updates.

## **Controller**

A controller gets a model from a route and makes the bridge between the model and the view or template. It is the perfect place to handle user events or actions like submitting the form and apply the business logic before making request to API.

## **Template**

Templates are used to present the user interface, which are written with the HTMLBars templating language, a variation of handlebars template system. HTMLBars templates print the model data and automatically update it when the model changes.

In addition, it has the usual templating logic like, if-else, each-loops as well as formatting helpers to allow developers to build their own helper functions.

## **View**

A view is associated with a controller, a template and a route. The difference between views and templates can be tricky. Views are best when we want to handle events or handle some custom interactions of elements that are impossible to manage from templates or CSS. Views are used to add sophisticated handling of user events, JavaScript animations and custom graphics not made with CSS.

### **3.4 Search Engine Optimization (SEO)**

Search Engine Optimization (SEO) is the process of optimizing a website which might involve editing its associated codes, content, HTML or making it more relevant to specific keywords to increase visibility in search engines. The sites having higher rank on search engines get visible earlier and more frequently in search result list, which certainly affects the web traffic to the site. On the other hand, it also takes care of any behavior that cause any barrier for the site being indexed by search engines. The process of SEO considers on what people search for, how search engines work, relevancy of content to the search keyword etc. that makes a remarkable contribution for Internet marketing. (Wikipedia, 2015c)

## **4 Objectives for the research**

The web is evolving rapidly, new technologies arise, and old methodologies quickly become irrelevant. Having numbers of frameworks providing similar kind of features, choosing a single framework is one big challenge. Nevertheless, choosing Ember.js for a project is another huge decision. Hence, this report focuses on study over different aspect of Ember.js to achieve the following goals.

### **4.1 Comparing Ember.js with similar frameworks**

Choosing a right framework for our application may have a vital impact on ability to deliver on time, and ability to maintain application code in future. An application can easily get dirty when features grow, which requires a scalable framework to maintain it properly. As there are many frameworks with similar design and features, it can be confusing to choose one of them. Hence, we will try to find other alternative frameworks that can provide similar feature or closer to what Ember.js does and compare with them. Based on top JavaScript MVC frameworks presented by InfoQ (Synodinos, 2013), AngularJS, Backbone.js and Knockout.js seem to be possible alternative framework to Ember.js. Hence, we will go through an in-depth comparison of these three frameworks with the possible sources available on Internet.

### **4.2 Addressing SEO challenges from Ember.js standpoint**

SEO is a significant tool for a content-heavy web application, which enables its targeted users to reach the relevant content from our site when they search in any search engine. On the other hand, implementing SEO in an application along with a JavaScript framework has been a well-known issue. Obviously SEO is a broad topic and many factors such as content of application, response time by server etc., independent of framework we choose and thus, will not come under our scope of study. Hence, we will accomplish the following tasks to keep our Ember.js application SEO friendly.



- Use SEO friendly URLs or make it human readable instead of having some random number or text as URL.
- Update document title of page dynamically based on current route.
- Set Meta tags content dynamically depending on current route. For example: different route of the application will have different content for Meta tags. I.e. description, keywords etc.
- Set Facebook open graphs properties dynamically based on current route. For example, change in model or route will have different og:image, og:url, og:description etc.
- Make the site crawlable by search engines.

In order to achieve the above listed goals, we will create a sample application and host in a server. Before we can even check in search engine or in Facebook, we will need to make the site crawlable first. Then apply our solutions and make test on Google search and also post a link to Facebook to check if open graph properties are applied properly.

## 5 Research

In this section, we will address the research objectives mentioned above. For comparing Ember.js with other possible alternatives, data from different Internet sources will be collected. However, addressing SEO challenges from Ember.js standpoint will involve finding solutions on Internet, documenting them and testing with sample application.

### 5.1 Comparing Ember.js to similar front-end frameworks on the market

The Landscape of front-end web development consists of so many libraries, tools, and frameworks that it has become very difficult for anyone to keep up with everything. Developers these days are spoiled with choice when it comes to selecting MV\* framework for structuring and organizing their JavaScript web apps. If one does a quick Google search for “JavaScript MVC Frameworks”, it is really easy to become overwhelmed with all the choices that one can choose from. Table below lists the top four frameworks as presented by InfoQ (Synodinos, 2013).

Frameworks	Version	Size	Initial Release	Source
AngularJS	1.3.15	50 kB compressed and minified	2009	Angularjs.org
Backbone.js	1.1.2	6.5 kB, Packed and gzipped	13 October, 1010	Backbonejs.org
Ember.js	1.7.0	95 kB, minified & gzipped	2011	Emberjs.com
KnockoutJS	3.3.0	21 kB, minified & gzipped	5 July 2010	Knockoutjs.com

Table 5.1 List of popular MV\* frameworks on market

All these frameworks today have a lot in common. They are open-sourced, released under the permissive MIT license, and try to solve the problem of creating Single Page App with MV\* design pattern. Nevertheless, it is no easy job to decide which one to choose. So, we will try to compare them based on community popularity as well as main features they provide.

### 5.1.1 Community popularity

One of the biggest factors to consider when choosing a specific framework is size of the community and the support behind it. A large community means more questions answered, more third party modules, easy solutions as well as assertion of future development. So, we will compare each of these frameworks based on the following key points.

- Google trends for frameworks' keyword
- Third-party modules, GitHub and other metrics
- Commits history
- Top 10 contributors and their commits count

#### Google trends for frameworks' keyword

Google trends might be the easiest way to find the most popular framework on Internet. Google provides a tool, which helps us to compare keywords, which are being searched on web, and illustrates comparison among searched keywords over time. Hence, the figure 5.1 shows the trends for these four frameworks on Internet from 1 June 2010 till 31 March 2015 within computer and electronics category.

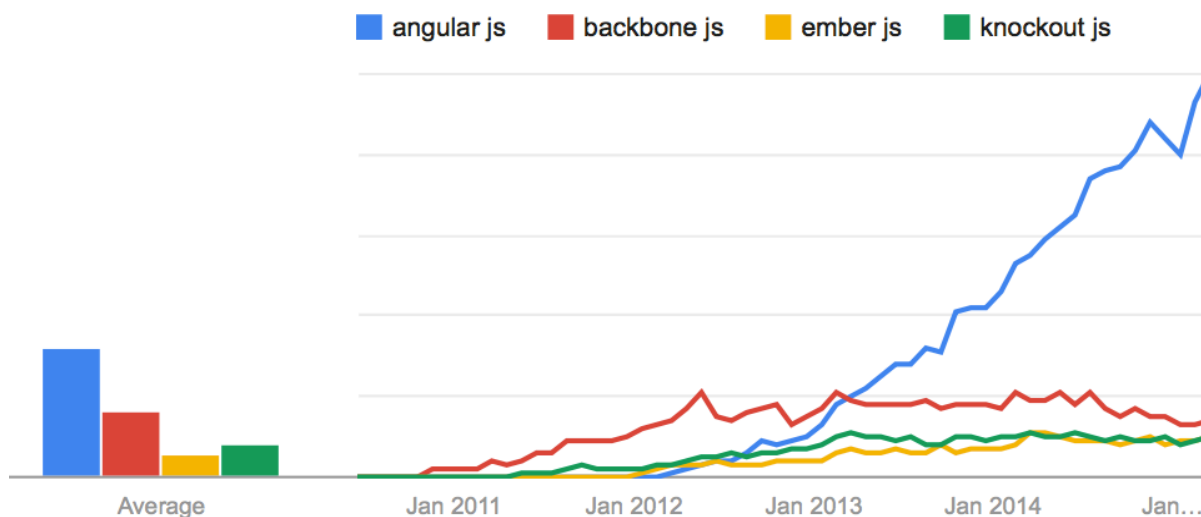


Figure 5.1 Worldwide Google trends over time on web search. June 2010 -March 2015

It becomes pretty clear that “angular js” is highly searched term on Internet while other three frameworks terms are almost not comparable. However, we should also remember that the graph compares only for the keywords searched regardless of what they mean. If a person searches for a tech company named “backbone” which not necessary means the Backbone.js framework, then that search is also accounted in this graph. On the other hand, the graph also does not include search count for framework if the framework is known with any other keywords. For example: Ember.js which was first known as SproutCore 2.0 and name as Amber.js then renamed it again to be Ember.js as now.

### Third-party modules, GitHub and other metrics

As all of these four repositories are open sourced and hosted in GitHub, it makes easy to compare them based on their commits, GitHub stars and contribution activities. Table 5.2 presents author-collected data as of 29 March 2015 from GitHub, Stack Overflow and other sources.

Metric	AngularJS	Backbone.js	Ember.js	Knockout.js
Third-Party Modules	1,312 (ngmodules.org, 2015)	257 (Backplug.io, 2015)	798 (Emberaddons.com, 2015)	63 (Steiner, 2015)
Stack Overflow questions	85,428 (Mitchell, 2015)	17,675 (Waite, 2015)	14,529 (Iwburk, 2015)	13,270 (Stack Overflow, 2015)
Stars on GitHub	36,818 (AngularJS, 2015)	21,202 (Ashkenas, 2015)	13,197 (Ember.js, 2015b)	6,201 (Knockout, 2015)
GitHub Contributors	1,212 (AngularJS, 2015)	253 (Ashkenas, 2015)	468 (Ember.js, 2015b)	50 (Knockout, 2015)

Table 5.2 Current metrics on popularity of each framework

In terms of third-party modules, AngularJS is clearly the richest framework with 1,312 modules. In the second place, Ember.js has around 800 modules whereas Backbone.js and Knockout.js have less than 300 modules. Besides, the highest number of questions on Stack Overflow also shows that it has become point of interest among many developers.

Contributors on GitHub help to understand the governance style of project whether it is managed by a small group of people or open to contributions from diverse audience. Angular is a clear winner here. However, we should not overlook the fact that Angular is the oldest (initial release 2009) of the three frameworks and Backbone.js initial release was in 2010 whereas Ember (initial release 2011) is the youngest. All these metrics, however, merely show the current state of each framework. It is also interesting to see which framework has a faster-growing popularity, which we can study from commits history and Google Trend.

### Commits history

Commit history of the project help us to learn about the trend and growth of particular framework. Commits history data can be retrieved from logging system of Git version control or from the GitHub API for any Git repository. Hence, author collected Git log for each framework is presented on the figure 5.2.

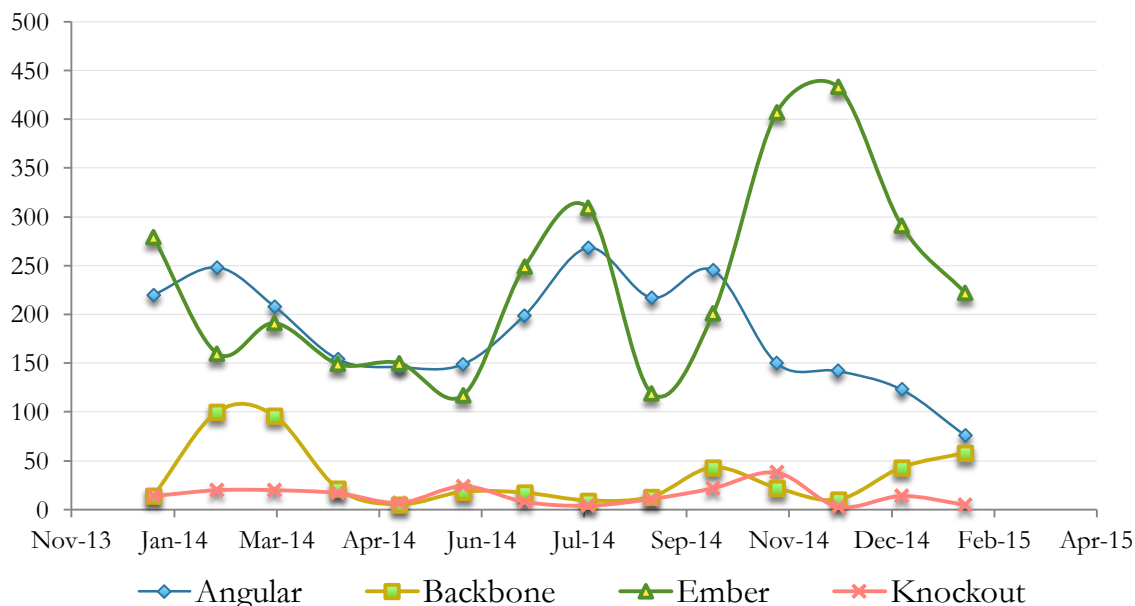


Figure 5.2 Commits history of each framework since 2014 till February 2015

As we can see from the graph above, Ember.js has a growing number of commits whereas Knockout.js and Backbone.js have very minimal number commits compared to AngularJS and Ember.js. In terms of commits history, AngularJS comes in second position but we should also consider that AngularJS community is moving to a complete re-write of framework as AngularJS 2 in a new repository which might have caused dropping number of commits.

## Contribution distributions

Besides the number of commits or contributors in any project, it's also important to know who is actually developing them. Contribution distribution in the chart below shows distribution of commits amongst top 10 contributors of each project. The data is taken from GitHub as of date 11 April 2015.

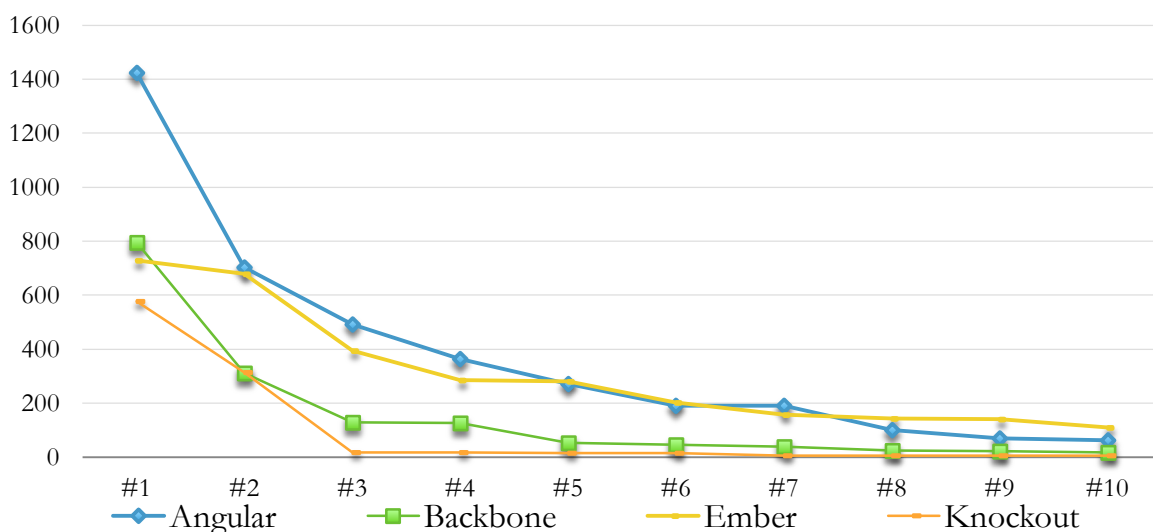


Figure 5.3 Contribution distributions of top 10 contributors in each framework till February 2015

The higher a line stays up before it drops down, a better indication that there is a healthy distribution of commits amongst contributors. From this measure, Ember.js and AngularJS seems have evenly distributed commits and share the credit equally among committers compare to others. Backbone.js seem to have only few contributors committed to project whereas Knockout.js seems a bit of a one-man show.

### 5.1.2 Concept and main features comparison

Based on the community popularity metrics and data, AngularJS seems to be highly anticipated framework for future development whereas Knockout.js has very few activities going on. From Ember.js standpoint, it would make more sense to compare it with AngularJS or Backbone.js to learn why we should move to these frameworks or should we just stick to Ember.js. Hence, we will compare Ember.js features with AngularJS and Backbone.js only, leaving out Knockout.js from now on. In order to understand the idea behind these frameworks, let's start by getting into core concepts and ideas behind them.

AngularJS's catchphrase "HTML Enhanced for Web Apps" explains pretty clearly its philosophy and how it is actually used. Angular is heavily driven by directives, which allows developer to extend the HTML with special tags and properties. This way of writing code allows us to express view logic using real markup code and thus feels like "enhanced HTML". Some describe AngularJS as a library; many believe that it's a framework, whereas lines below by Misko Hevery from AngularJS core development team claims that it's an html compiler rather than a framework.

"But a better way to think about angular is not to think about it as framework but as HTML compiler, which allows you to create your OWN DSL in HTML, by attaching your own behavior to any HTML element, attribute or text. And by any I mean that you can make up your own names (outside those of HTML spec)." (Hevery, How do Angular.js and Backbone.js compare?, 2011)

Backbone.js doesn't boldly have a catch phrase, but the creator Jeremy Ashkenas, likes to say, "There's more than one way to do it". As we can probably understand from that statement, Backbone.js was built to be very flexible. Even though Backbone.js does not offer many features out of the box, it does provide platform developers to extend via numbers of plugins.

Ember.js's catchphrase "A framework for creating ambitious web applications" from its home page presents itself very clearly and boldly as the framework for large web applications. Having "conventions over configuration" philosophy, Ember.js is a very holistic framework in the sense that it'll probably provide us with everything we need and more, but because of this it also tends to be one of the biggest frameworks in terms of size.

Furthermore, in order to understand these frameworks better, we will breakdown the core components of these frameworks in terms of MV\* Pattern as below.

- Routing
- Models
- Views/Templating
- Data binding

## Routing

The simplest routing infrastructure a MV\* JavaScript Framework can offer is mapping functions to URLs, similar to their server-side relatives (e.g. Grails, Spring, ASP.NET, CodeIgniter). Over the past couple years, we are seeing a big shift of the “backend” functionality happen more on the client side and routing is the major component of how these Single Page Apps are developed. With Single Page App (SPA), we can have one common UI and just request the specific data that’s needed. This is commonly achieved by adding a Hashtag (`#foo`) to the path (or might even be full path assuming HTML5 push-state is enabled) which tells the browser not to fetch a request to web server but allows the client side MV\* framework take care of the logic for those respective paths.

Routing in Ember.js is considered to be the core of the application, which serves, as almost the root file a developer would first start looking to understand the whole application. It is quite intuitive that Ember.js route figures out its model, view, and controller along with URL path just with a single line of code as long as we follow the naming convention. As shown in table 5.3, we can simply provide the View as first parameter while defining a route and Ember.js router will automatically look for these classes in application unless we are planning to change the default behavior.

```
App = Ember.Application.create();
App.Router.map(function() {
  //first paramater refers to the view/template
  this.route('home', { path: '/' });
  this.route('blogs');
});
//Defining data for route
App.BlogsRoute = Ember.Route.extend({
  model: function(){
    return {
      title : "Title for the blog post"
      body  : "Body for the blog post"
    };
  }
});
```

Table 5.3 Defining routes in Ember.js



By default, an Ember.js app expects the path to route, controller and templates as below.

- app/routes/blogs.js
- app/controllers/blogs.js
- app/templates/blogs.hbs

If we needed more granular customization, we can define different Controller, Template, Model. Besides, we can also change default behavior of data access on model and fetch data from API via Ajax in traditional way ( Knebel, 2013). For example in second part of code above, we are simply returning JSON object as data for BlogsRoute.

In contrast to Ember.js’s convention-over-configuration routing mechanism, AngularJS requires a template and even a controller to its router configuration. We have to manage our template and controller manually but we can do that almost without restrictions. Routing in Angular is very clean and simple. As shown on codes in table 5.4, Angular uses the “\$routeProvider” service to take care of the routing for our application. It uses a simple “object like” syntax of routing, for example the first route for “/home” simply declares that when user visits “/home” path, application should use “home.html” template backed by “HomeController” controller.

```
var routingExample = angular.module('Example.Routing', []);
routingExample.controller('HomeController', function ($scope) {});
routingExample.controller('BlogController', function ($scope) {});
routingExample.config(function ($routeProvider) {
    $routeProvider.when('/home', {
        templateUrl: 'home.html',
        controller: 'HomeController'
    }).when('/blogs', {
        templateUrl: 'blog.html',
        controller: 'BlogsController'
    }).otherwise({
        redirectTo: '/home'
    });
});
```

Table 5.4 Defining routes in AngularJS

Routing in Backbone is similar to AngularJS as it also uses similar “object like” syntax. However, it’s slightly different for the reason that it maps templates through functions instead of declaring them with in the route. For example in the code below in table 5.5, when the user visits the blogs page with URL ‘/blogs’, ‘blogsRoute’ function gets triggered. It is important to note that Backbone actually has no controller; the route is actually the closest thing to a controller in Backbone.js.

```
var AppRouter = Backbone.Router.extend({
  routes: {
    '': 'homeRoute',
    'home': 'homeRoute',
    'blogs': 'blogsRoute',
  },
  homeRoute: function () {},
  blogsRoute: function () {}
});
```

Table 5.5 Defining routes in Backbone.js

## Models

The Models are data objects that are stored in application to present on the UI. In an MV\* pattern, the model commonly serves two purposes. The first is to define the data object for storing data, and the second purpose is to be able to observe model changes and respond accordingly for state management.

In Angular the model is tied to the “\$scope” object of the controller and is used in a plain JavaScript object like style. Having dirty checking technique for data binding in AngularJS, there is no need to call any method to grab data or send data from and to the UI.

While AngularJS’s data binding takes care of the state management of the model, Backbone handles models a bit differently. Backbone models are like JavaScript objects too but with inherited methods to provide an API access and manipulate the data. Backbone allows us to “extend” a model to use `get ()` and `set ()` functions in addition to any other custom functions the developer creates.

The data model in Ember.js is a very similar concept to Backbone in the same style that we can “extend” the model and use getters and setters. However, it has a numerous way we can store, cache, and persist data. For example, we can also use Ember data to define data model, manage model data, store in data-store as well as persist data via adapters.

## Views/Templates

Templating is basically a presentation of HTML document with special parameters that eventually get replaced by arguments from a data model. There are two types of templating: string based and DOM based. String based templates are similar to the common jQuery `html (“”)` or the `innerHTML` method in which we can inject a string of HTML and the browser would parse it to HTML. For example, `$('#body').html("<p>Paragraph inside body</p>")`. In contrast DOM based templating will crawl the DOM tree to discover special elements in the tags (like angular directives) and manipulate the DOM tree to manufacture the final DOM tree.

Angular uses a DOM based templating approach that is very easy to use out of the box. As shown in table 5.6, it uses “mustache” like syntax with curly braces, and data for ‘title’ and ‘body’ is automatically mapped from ‘BlogCtrl’ Controller.

```
<body ng-app =“myApp”>
  <div ng-controller=“BlogCtrl”>
    <h1> {{ title }} </h1>
    <p> {{ body }} </p>
  </div>
</div>
```

Table 5.6 Templating in AngularJS

Backbone, on the other hand truly justifies, “There’s more than one way to do it” which actually allows us to use any third party templating system. Although Backbone has dependency to underscore.js for out-of-the-box templating system, Handlebars is also a very popular templating system, that is commonly used with backbone.js.

Ember.js uses string based Handlebars.js templating system and is going forward for HTMLBars templating engine that is built on top of Handlebars.js. HTMLBars is designed to compile Handlebars that builds a DOM rather than a String that can fake DOM in server and render server compiled templates, which not only improves performance, but also solve SEO issue, which is a major drawback in most of the front-end frameworks.

### **Data binding**

Data binding is a state of the current UI when it is in sync with the latest data model. In other words, it is a consistent snapshot of our data model presented in UI. For example, when a user fills out a form in a couple of input fields, if there's data binding in place the data model will be updated automatically to reflect the latest changes of the input fields.

AngularJS is well known for its two-way data binding. Having regular JavaScript objects as the 'model' in AngularJS, we don't even need to do anything to keep the data in sync because it does some "dirty checking" behind the scenes. This is done by checking if the property in that "\$scope" has been changed and if so, update the DOM and save the new value.

Backbone doesn't support data binding out of the box but there are numbers of third party plugins that you can integrate to support data binding.

Ember does support data binding but does so via "getters" and "setters". Ember.js provides ability to bind property by watching some other object, property and reflect changes immediately. In addition, it also provides the idea of observers to observe properties and trigger function.

Compared to AngularJS, 'getters' and 'setters' in Ember.js can get very verbose as we are constantly setting/getting, but the advantage of the ember way is that it is not dirty checking and thus gives more performance benefit. One of core members of the AngularJS team Misko Hevery points out that we can have up to 2000 properties bound at a time (Hevery, Databinding in AngularJS, 2012). The continuous dirty checking starts to be expensive when the collection of a specific \$scope has a ton of properties.

## 5.2 SEO challenges from Ember.js standpoint

Keeping a web application SEO friendly as well as harvesting the goodness of front-end JavaScript frameworks is one of the biggest challenges for developers. Hence, this section will address the SEO challenges in Ember.js. We will also create a sample application with a model name 'framework' and apply our findings and solution to this application. The application will be running at <http://seo.bitsocean.com> and source code will be available at GitHub repository: <https://github.com/sunil-shrestha/seo-friendly-ember.js> (Shrestha, 2015). Table 5.7 below represents the framework model in our sample application.

Framework	
name	String
description	String
source	String
initialRelease	String
liscence	String

Table 5.7 Attributes for 'framework' model in sample application

### 5.2.1 Make SEO friendly URLs

Most of the client side MVC frameworks default to hash-based URLs that take advantage of the fact that characters in a URL after an # are not passed through to the server. Once the JavaScript application boots up, it looks at hash data and figures out what it has to do. However, in modern browsers, we have better alternative to hash-based URLs. I.e. HTML5 History API that allows our JavaScript code to modify the URL without reloading the entire page. Thus, this approach helps us to maintain the traditional way of URL “<http://example.com/blogs/blog-title>” instead of using URL like “<http://example.com/#/blogs/blog-title>”. By default, `locationType` for routes is set to 'auto' which uses URLs based on HTML5 History API if possible and fallbacks to '#' based URLs. However, we can configure it manually as follows within `router.js` file.

```

// app/router.js
import Ember from 'ember';
var Router = Ember.Router.extend({
  location: 'auto'      //possible options: auto, history, hash,
  none
});
Router.map(function() {
  ...
});
export default Router;

```

Table 5.8 Configuring Ember.js router location type

So far, Ember.js seems to be pretty good in handling URLs by providing options to keep them clean with History API as well as fallback to '#' as per need. Nevertheless, our main topic of interest here is creating the human readable or SEO friendly URLs. Search engines do indexing on URLs and look for keywords in it. Both search engines and people prefer websites to have consistent, easy-to-read URLs whereas poorly structured URL can impair rankings and keep pages out of the search engine indexes. Therefore, we want to have page URLs as in first row in table 5.9 instead of a complicated URL in the second row.

<code>http://example.com/frameworks/emberjs</code>
<code>http://example.com/frameworks/Ember20%js</code>

Table 5.9 Example of URL structure

In order to achieve this human readable URL structure with Ember.js, we can use slug as the unique identifier to our resources. However, we also need to make sure that the property we use as slug is a unique attribute. Table 5.10 shows our model 'framework' with 'slug' as a computed property from name of framework and same property will be used for constructing a URL for each resource.

```

// app/model/framework.js
import DS from 'ember-data';

var Framework= DS.Model.extend({
  name          : DS.attr(),
  description    : DS.attr(),
  source        : DS.attr(),
  initialRelease : DS.attr(),
  liscence      : DS.attr(),
  slug: Ember.computed('name', {
    get: function () {
      return this.get('name').toLowerCase()
        .replace(/[\^\w]+/g, '')
        .replace(/ +/g, '-');
    }
  })
});

```

Table 5.10 Adding computed property 'slug' to use for resource URL

Now we can modify our router to use 'framework\_slug' instead of 'id' for the dynamic part of our framework route as shown in table 5.11.

```

// app/router.js
import Ember from 'ember';

export default Router.map(function() {
  this.resource('frameworks', function () {
    this.route('framework', {path:('/:framework_slug')});
  });
});

```

Table 5.11 Using slug as unique identifier instead of id

Once we have our slug property for the model, we need to tell our route to look for slug property instead of id of model. However, we also need to serialize the model so that it understands the dynamic part of route 'framework\_slug' as 'slug' property of the model. Table 5.12 presents the framework route after adding this serializer.

```

// app/routes/frameworks/framework.js
import Ember from 'ember';

export default Ember.Route.extend({
  model: function (params) {
    return this.modelFor('frameworks')
      .findBy('slug',params.framework_slug);
  },

  serialize: function(model){
    return {framework_slug: model.get('slug')};
  }
});

```

Table 5.12 Adding serializer to model for using slug as dynamic part of route

After having all these codes intact, we should be able to have a SEO friendly URL. For example, when we have a framework record with name ‘ember.js’, it should set our URL to ‘<http://example.com/frameworks/emberjs>’.

### 5.2.2 Create dynamic document title based on current route of page

The document title of a page is defined by `<title>` tag, which is obligatory in all HTML documents. For any HTML document, document title defines the title in browser toolbar, provides a title for the page when it is added to favorites, displays a title for the page in search-engine results as well as plays a vital role for the page being indexed by search engines. (W3Schools, 2015). There are many ways to set our document title for page and Ember.js has couple of add-ons just to do that. We could write our own code also, but it can also be good idea to install an add-on and harvest the best possible solution as well as benefit from Ember.js community. Thus, we will install an add-on ‘ember-cli-document-title’ to our sample application from <https://github.com/kimroen/ember-cli-document-title> which is quite well documented and contributed by some of the developers who are also in Ember.js core development team.



The add-on simply adds two new keys `'title'` and `'titleToken'` to our routes, which can be string or a function and sets the document title every time when the application transition to any route. With installation of the add-on, Ember.js collects the `'titleTokens'` from the leaf most route and bubble them up until it hits a route that has title defined and finally, returned value from the title function is used as the document title. For example: we can create the `'title'` function in application route as shown in table 5.13 and set `'titleToken'` in child routes as in the tables 5.14 and 5.15.

```
// app/routes/application.js

import Ember from 'ember';

export default Ember.Route.extend({

  title: function(tokens) {
    if (tokens.length > 0) {
      return tokens.join(' | ') + ' - BitsOcean';
    }else{
      return 'BitsOcean';
    }
  }
});
```

Table 5.13 Adding title function in application route

```
// app/routes/frameworks.js

import Ember from 'ember';

export default Ember.Route.extend({
  ...
  titleToken: 'Front-end frameworks'
  ...
});
```

Table 5.14 Setting titleToken for frameworks route

```

// app/routes/frameworks/framework.js

import Ember from 'ember';

export default Ember.Route.extend({
  model: function (params) {
    return this.modelFor('frameworks')
      .findBy('slug', params.framework_slug);
  },
  ...
  titleToken: function (model) {
    return model.get('name');
  }
});

```

Table 5.15 Setting titleToken for framework route (leaf most route)

Thus, this will generate different document title for every record in ‘framework’ model based on their name. For example, When record name is ‘AngularJS’, document title will be “Front-end frameworks | AngularJS – BitsOcean” and when record name is ‘Knockout.js’, document title will be “Front-end frameworks | Knockout.js – BitsOcean”.

### 5.2.3 Set Meta tags dynamically depending on current route

The <meta> tag provides metadata about the HTML document which is machine parsable, but not displayed on the page. Meta elements are usually used to specify page description, keywords, and author of the document, last modified, and other metadata. Meta tags can help describe any page in a more convenient machine-readable format and more suited to search engines. They are best used when accurately describing the page, and helping search engines to shortcut to information about our page.

“Meta tags are a great way for webmasters to provide search engines with information about their sites. Meta tags can be used to provide information to all sorts of clients, and each system processes only the meta tags they understand and ignores the rest.” (Google Inc., 2015b)

As quoted above, Meta tags are great way to quickly tell search engines about the Meta data of any HTML document. Besides, meta tags are also used by search engines while indexing as well as presenting the page as matching search results. Figure 5.4 below shows the search result for “prerender.io for ember.js” in Google search. When we follow the link and look into the HTML document of the site, we can see the Meta tag with name ‘description’ as on the table 5.16

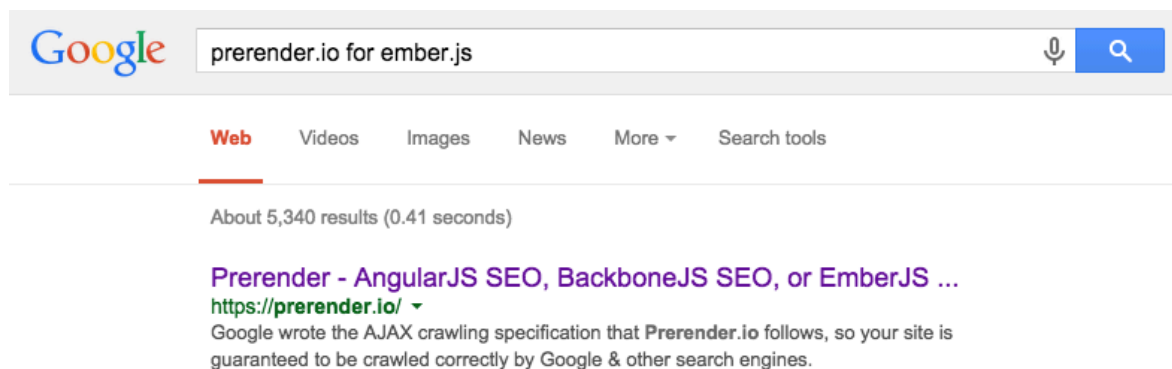


Figure 5.4 Meta tags content used by Google search while listing search result

```
<meta name="description" content="Allow your AngularJS, BackboneJS, or  
EmberJS apps to be crawled perfectly by search engines. View on  
Github.">
```

Table 5.16 Meta tag from the HTML document of the site

As shown in the result above, Google presents description below the URL of each search result, which is extracted from the content of Meta tag name ‘description’ from HTML document. Meta tags not only benefit sites with SEO but also help people to get quick information about the site immediately from result page. Hence, we will add Meta tags to our sample application so that the content of Meta tags changes depending on the route we visit. In addition to basic Meta tags, we will also implement Meta tags for Facebook open graph.

## Adding basic Meta tags

Adding dynamic Meta tags in Ember.js application is pretty straight forward with an Ember add-on “ember-cli-meta-tags”. With this add-on, we can have those Meta tags populated when entering individual routes. So, we can simply follow documentation of the add-on in GitHub and setup our application. (ember-cli-meta-tags, 2015) . Once the add-on is installed, we can import the mixin provided by add-on and extend our route from it. Table 5.17 below shows the setup for our sample application where we have extended our ‘app/routes/frameworks/framework.js’ route.

```
// app/routes/frameworks/framework.js
import Ember from 'ember';
import RouteMetaMixin from 'ember-cli-meta-tags/mixins/route-meta';

export default Ember.Route.extend(RouteMetaMixin, {
  model: function(){
    ...
  },

  meta: function (model) {
    var model = this.get('currentModel');
    return {
      'name': {
        'description' : model.metaDescription.substring(0,150)+ '...',
        'keywords'    : model.title
      }
    };
  }
}
```

Table 5.17 Adding Meta tags for framework route

By extending the mixin from add-on, we are provided a property ‘meta’ inside the route, which can be set as a function or a JSON object. The add-on will then nicely populate the Meta tag within header tag of document when user visits this route. In the snippet above, we are simply accessing the current model (record) and setting the description and keywords meta tags based on the current model for now but we can also set additional meta tags name if needed.

## Adding Meta tags for Facebook open graph

'Ember-cli-meta-tags' add-on also provides ability to set meta tags' property which is very useful when we want to set our meta tags targeting to precise social networks and sharing platforms. In this case, we will setup Meta tags for Facebook open graph by modifying our previous route file and adding additional key-values as required by Facebook open graph such as og:url, og:image, og:description etc. Table 5.18 below shows our route after modification (Facebook Inc, 2015b).

```
// app/routes/frameworks/framework.js
...
...
meta: function (model) {
  var model = this.get('currentModel');
  return {
    'name': {
      'description' : model.metaDescription.substring(0,150)+ '...',
      'keywords'    : model.title
    },
    'property': {
      'og:name'      : model.title + ' - BitsOcean',
      'og:url'       : document.location.href,
      'og:image'     : 'http://i.imgur.com/ilmbIh1.png',
      'og:description' : model.metaDescription.substring(0,150)+
'...'
    }
  };
}
```

Table 5.18 Adding Meta tags for Facebook Open Graph

Hence, dynamic Meta tags in an Ember.js app can be achieved pretty easily once we have the add-on installed. With this setup in our sample application, we should be able to see the Meta description under the URL of search result when, our application is presented in Google search result. In addition, above settings for Facebook Open Graph will suggest Facebook to apply the particular name, URL, image and description to show when someone shares or posts the link to our application.

#### 5.2.4 Make the site crawlable by search engines.

Single page applications with client side frameworks have been challenging for search engines to process because the browser can execute JavaScript and produce content on the fly but the crawler cannot. AJAX content is produced dynamically by the browser and thus not visible to crawlers. To make the crawler see what a user sees, the server needs to give a crawler an HTML snapshot, the result of executing the JavaScript on the page. While there are existing methods for dealing with this problem, they involve regular manual maintenance to keep the content up-to-date. This is a real trade off developers have to consider before deciding to go with an MV\* framework instead of an application that does its rendering on the server side. (Google Inc., 2015a)

“If some of your content is created dynamically, the page source will not include all the content you will want the crawler to see. In other words, "View Page Source" is exactly what the crawler gets. Why is this important? It is important because search results are based in part on the words that the crawler finds on the page. In other words, if the crawler can't find your content, it's not searchable.” (Google Inc., 2015a)

Thus, Keeping Crawlers happy with client side frameworks one of the challenging job for developers. Google has set some conditions to be agreed between our application and Google crawler so that the crawler can reach out to our AJAX enabled page content and get it indexed, which can be summarized as follows.

- The site should adapt AJAX crawling scheme.
- The server should provide an HTML snapshot for each AJAX URL, which is the content a user (with a browser) sees.
- The search engine indexes the HTML snapshot and serves original AJAX URLs in its search results.

## **Prerender.io**

There are numbers of open source projects, which are trying to solve this issue while only few of them are able to give satisfactory result. However, paid service like Prerender.io seems to be quite a promising option for many businesses. Prerender.io simply renders JavaScript in a browser, save the static HTML, and our application/server returns that to the crawlers. If a crawler ever tries to access a page that isn't cached, Prerender.io will render it on the fly and cache it afterwards. Furthermore, Prerender.io also provides free service for saving up to 250 pages. (Prerender.io, 2015) Thus, we can sign up for free service in Prerender.io and setup our server so that it will request for static HTML from Prerender.io and return it whenever we get request from a crawler. Once, our sample app is ready and running at server, we can simply follow the documentation, which provides numbers of options and snippets depending on platform we use. In our case, the sample application will be running in nginx server, so we will configure it similar to Gist snippets provided by Prerender.io (<https://gist.github.com/thoop/8165802>).

## **Ember-prerender and ember-cli-fastboot**

In order to solve the particular issue in an Ember.js application, there is also a project in GitHub named 'ember-prerender' (<https://github.com/zipfworks/ember-prerender>), which makes use of Node.js and JSDOM, PhantomJS, or WebDriverJS depending on our preference. However, we will need to run this service in our own server unlike Prerender.io doing this for us. Unlike the Prerender.io Service, the goal of ember-prerender is to reduce rendering times by utilizing a long-lived instance of an app instead of reloading it on every request. In addition, we also get the flexibility of using JSDOM or WebDriverJs instead of only PhantomJS. (Ember-prerender, 2015)

Main reason why we chose Prerender.io over ember-prerender is that the ember-prerender project seems to be not very active project and, if we check commits history, there is not much going at the moment. On the other hand, Ember.js core team is actively developing an add-on 'ember-cli-fastboot' which has promised to solve not only this SEO issue but also increase the efficiency of initial page load. There are still several major restrictions we should be aware of and only the bravest should even consider deploying this to production.

While looking for the best options to get our sample application crawled by search engines, ember-cli-fastboot has also been experimented. Though, it could not provide us a satisfactory result at this phase, we should definitely keep an eye on this project so that we can get this functionality out of the box with Ember.js app and not require Prerender.io.

### **5.3 Applying SEO challenges solutions to sample application**

Our sample application is built upon Ember.js 1.12.0, Ember-data 1.0.0-beta.19.1 along with numbers of other ember-add-ons. The application contains a model ‘framework’ representing any front-end frameworks such as Ember.js, AngularJS etc. The application is running at <http://seo.bitsocean.com> and repository is hosted at GitHub (<https://github.com/sunil-shrestha/seo-friendly-ember.js>). When user visits our site, the application takes user to ‘<http://seo.bitsocean.com/frameworks>’ URL, which shows the list of frameworks and some description. Clicking on any framework item will take user to particular framework’s page. Appendix A demonstrates the current look of landing page in our application. Now we have our application ready and running on server, we can start to test if we have achieved our goal. Hence, for each of our goals mentioned above, we will evaluate the result from the sample application.

#### **5.3.1 Make SEO friendly URLs**

In order to have SEO friendly URLs, each of the record should have URL with slug text instead of ids or meaningless strings. I.e. URL should look something like ‘<http://example.com/frameworks/backbonejs>’. Appendices C and D are the snapshots taken from our sample application, which show the page with URLs for two different resource with SEO friendly URLs. Comparing the images, we can conclude that we have successfully achieved our SEO friendly URL for each resource.



### 5.3.2 Create dynamic document title based on current route

We can test the document title of application by inspecting the HTML source code of running application and also verify that Search Engines are showing the correct title in search result. As shown in appendices C and D, we are able to get contextual document title for different records. I.e. document title tag for the framework AngularJS is “<title>Front-end frameworks | AngularJS - BitsOcean</title>” whereas for Backbone.js is “<title> Front-end frameworks | Backbone.js - BitsOcean </title>”. Similarly, the appendix B shows the search result for the term ‘seo bitsocean’ in Google search and result quite nicely shows the title of each result item mapped to the document title of each record.

### 5.3.3 Set Meta tags dynamically based on current route

The main objective of having dynamic Meta tags is to be able to set different Meta tag names and properties for different records. As we can see from the appendix B, search results in Google already present the description of every result quite nicely. Another way of verifying that we have dynamic Meta tags for page of each record is to look into the HTML document. As shown in the appendix C and D, Meta tags for name such as description and keywords are different in different pages of records. Additionally, we need to make sure that Meta tags for Facebook Open Graph also work as expected. Therefore, a link from sample application is shared to Facebook and figure 5.5 below demonstrates how nicely Facebook pulls the Meta tag properties we have set for Open Graph and shows while sharing a link.

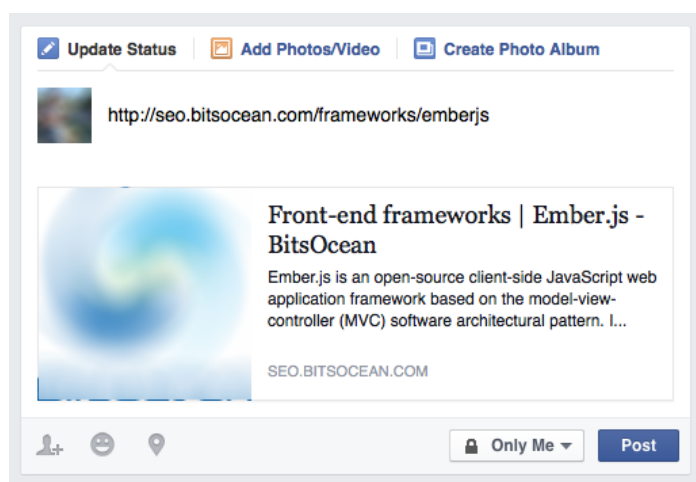


Figure 5.5 Meta tags for Facebook Open Graph displayed while sharing a link from sample application on Facebook

#### **5.3.4 Make our application crawlable**

As we can see, Facebook is able to pull information from our site, and show it while sharing or posting a link. Additionally, Google search results in appendix B also gives proper information about our application, which clarifies that we are able to make our site crawlable.

## 6 Conclusions and Evaluation

It's very overwhelming to imagine the future of web application with all these fast evolving tools and frameworks. Frameworks like AngularJS, Backbone.js, Ember.js etc. brings structure to our JavaScript code and keep them organized, but they might also overkill if used for small applications. Thus, many times, it's a good idea to think if the application really needs any framework. All these frameworks have their own kind of benefits and scenarios where they fit the best. Instead of having competitive comparisons of these frameworks, one good approach on picking the framework would be to know the requirement of our application and choose the one that fits the best.

After our study about community popularity, AngularJS seems to be highly anticipated framework for the future development of the single page application. Some of the best features like two-way data binding, modularity, dependency injection, etc. have made it highly popular amongst the developer community. However, two-way data binding can be pretty expensive when the application grows with numerous components because it has to make dirty checking continuously for every binding. Many times, developers claim that, integrating AngularJS with other libraries and API are quite a struggling job. Especially when a developer with jQuery background enters to AngularJS world, where DOM manipulation is designed with a totally different approach.

In terms of integration and customization, Backbone.js is considered as the best option. Besides, it is also very modular, plays quite well with other libraries and quite easy to get solutions from Internet in case we get stuck at some point. On the other hand, it is very minimal, requires third party plugins to get data bindings working and memory management has to be done manually for destroying events and views. In spite of being a popular framework, commit frequency seems to be slowing down and not many contributors seem to be active. However, Backbone.js can be best options for small or medium type of applications with the possibility of integrating with many other libraries.

Backbone.js sets many options and flexibility, which also means that there are many choices to be made while creating an application. Deciding which plugins or libraries are the best for our project requires research, which in turn takes time, and one of the main purposes of framework is to save our time. Thus, Backbone.js standalone can hardly be considered a complete framework, but can be taken as great tool that can be used to create structure, leaving it up to the developer to decide how to structure the application.

Contrasting with Backbone.js, Ember.js is a well-structured and very opinionated framework with lots of granular configurations and structures. Ember.js embraces an excellent routing mechanism, an optional data layer called ember-data and many other modules out of the box. Thus, it makes the size of Ember.js comparatively larger than others frameworks. Ember.js's philosophy on Convention over Configuration binds developers to follow best practice and keep the application standardized whereas, it might also narrow down the options on what developers can do. Ember.js is creating its own eco-system with a rapid growth of add-ons and development tools like ember-inspector, ember-cli etc. Learning Ember.js would mean also to learn its methods, tools, development environment, conventions as well as getting involved with whole eco-system which might be quite a steep learning curve for beginners. Nevertheless, Ember.js is still in rapid development which will require developers to keep an eye on latest updates and builds, but it's quite an exciting to see how Ember.js evolves in future.

## **6.1 Comparing results against the goals**

Looking back to our initial goal of study at this point, results are fairly satisfactory. While comparing community popularity of Ember.js with other front-end frameworks, we were able to get some concrete data and know the current trend. In addition, we are also able to find fairly good solutions to our all SEO challenges as well as get our sample application running on server. Hence, it can be considered that we have successfully accomplished our goal.

On the other hand, there are definitely some inadequacies, which are not covered. Comparison of front-end frameworks could go more deeply on architectural design and internal elements, which are only softly touched. In order to make our application SEO friendly and crawlable by search engines, the initial solution was to use ‘ember-cli-fastboot’ or ‘ember-prerender’ add-ons, which are meant to solve the particular problem. After testing our sample application with these add-ons, it became clear that they are not production ready yet. Thus, it led us to adapt limited service of Prerender.io, which might not be a preferred solution for many other cases.

## **6.2 Validity of the results**

Due to rapid development of web technologies, any report or information about it can be outdated pretty easily. Thus, the information reported here best fits with the current context of web technology and expected to change in near future. Comparison of front-end framework is done based on currently available alternatives on Internet whereas the data collected may not stay relevant for long in the future. There are many similar tools and libraries like canJS, montageJS, etc. that we could not include in our comparison, but it can still be interesting to see how they evolve in near future. Furthermore, our sample application is built upon current release of Ember.js and the SEO challenges are handled with currently available methods, tools and add-ons. Hence, it is important to realize that the solutions applied in this sample application can still be optimized or get some better alternative solutions in the future.

## **6.3 Evaluating used information sources**

Having study on information technology, specially related to Internet and web technology makes more sense to rely on Internet sources to access the most recent information. On the other hand, some information presented in this report relies on Stack Overflow, Quora and some personal blogs, which provide more of personal opinion than any authentic solution. Nevertheless, these sources still hold information from expert in particular field. Even though the URLs or contents of some blogs and forums might change, sources like Google, Facebook, and Wikipedia are expected to stay consistent for longer period.

## 6.4 Learning during the thesis process

After going through numbers of frameworks and features they provide, it can be seen that one of the major challenge for any single page application is to keep it SEO friendly. In addition, most of these frameworks are trying to solve more or less the same issues with web technology, which are as follows

- Improve the end-user experience and interactivity between user and application
- Improve the performance by providing all views layer and interactive components of application right on users' browser and keeping only data on the backend.
- Making the development of application more smooth, fast and organized.

During the time of this study, it can be seen that Ember.js framework has evolved quite fast and continuously getting better and better. The Ember.js team eventually released 'ember-data' which has been in beta testing for years, as well as Ember.js 2.0 along with its development tools. Although Ember.js is one of the best tools to develop an ambitious web application, continuous API changes, needing to keep track of latest updates, etc. has not been good experience for me personally. Thus, it can easily create lots of outdated documents, solutions and tutorials on the Internet, which confuses the developers as well as makes the learning more difficult.

## 6.5 Further research

Tools, techniques and data presented in this study provide an overview of available front-end frameworks as well as solve some of the SEO issues in Ember.js application. With the release of ember-cli-fastboot add-on, Ember.js community is trying to render virtual DOM at node server which will not only solve the SEO issue but also planned to improve the initial page load so that, it could show information on page immediately after the request is completed instead of waiting for all assets to load and then load any meaningful data via Ajax call. Nevertheless, it is also a good idea to look for other alternatives and choices available. For example: React.js powered by Facebook, which has gained a great deal of popularity lately for its ability to render virtual DOM at node server and provides solid solution to make an application crawlable by search engine. React.js is simply a UI component, which acts as 'V' in a 'MVC' framework.

Thus, we did not include React in our frameworks comparison, but it definitely worth taking a look.

“React abstracts away the DOM from you, giving a simpler programming model and better performance. React can also render on the server using Node.js, and it can power native apps using React Native.” (Facebook Inc., 2015a)

On the other hand, AngularJS community is working on AngularJS 2.0, which is not a complex major update, but a whole rewrite of framework. It is expected to have numbers of performance improvements along with solid architecture for future development. While attraction to AngularJS is growing, many AngularJS developers also suggest beginners to wait until the release of AngularJS 2.0 so that they could learn only one AngularJS framework. However, AngularJS team is trying to keep the community calm and providing an overview of future AngularJS at its preview site (<http://ng-learn.org/2014/03/AngularJS-2-Status-Preview/>).

As these applications run fully on users' browser, UI layer is loosely coupled with data servers and the communication between UI and server requires certain level of security. It is important to be aware of security issues such as CORS (Cross Origin Resource Sharing) as well as proper way of user authentication and authorizations.

## 7 Summary

Keeping track of all the latest releases and changes on front-end frameworks is obviously a challenging job. However, this study gives an overview of currently available front-end frameworks, compares Ember.js with them as well as addresses some of the commonly known SEO issues in Ember.js application. With an active developer community, Ember.js along with its add-ons, development tools as well as whole eco-system is growing very fast. On the other hand, AngularJS is equally an active project and proven to be the most popular in developer community based on our comparison data. Thus, AngularJS can be the best choice for the one looking for an alternative to Ember.js.

As many front-end frameworks are still facing a common challenge of making the application crawlable by search engines, Ember.js is no exception. Even though, we had to adapt the Prerender.io service, we have been able to make our sample application crawlable by search engines and handled all our planned SEO goals. However, with future version of Ember.js and ember-cli-fastboot, we expect to get our application crawlable by search engines without needing to adapt Prerender.io services.

Hence, all these tools, frameworks and active developers community has given great opportunities for any beginner to learn and build single page application in no time. On the other hand, framework like AngularJS, Ember.js, Backbone.js etc. are continuously evolving to provide better interactivity and user experience, which also makes clear that the web applications these days are being more user experience centric.



## Bibliography

Ashkenas, J. (2015, March 29). Backbone. Retrieved March 29, 2015, from Github:  
<http://github.com/jashkenas/backbone>

Knebel, J. (2013, November 7). An In-Depth Introduction To Ember.js . Retrieved April 15, 2015, from [www.smashingmagazine.com](http://www.smashingmagazine.com):  
<http://www.smashingmagazine.com/2013/11/an-in-depth-introduction-to-ember-js/>

Steiner, J. (2015, March 29). Plugins. Retrieved March 29, 2015, from Github:  
<https://github.com/knockout/knockout/wiki/plugins>

3nytechnology.com. (2015). Website Front-end and Back-end. Retrieved August 2, 2015, from 3nytechnology.com: <http://www.3nytechnology.com/website-frontend-and-backend/>

AngularJS. (2015, March 29). Angular.js. Retrieved March 29, 2015, from Github:  
<http://github.com/angular/angular.js>

Backplug.io. (2015, March 29). Backbone plugins. Retrieved March 29, 2015, from backplug.io: <http://backplug.io/>

Brehm, S. (2012, October 1). Building Single-Page Apps. Retrieved April 16, 2015, from Airbnb: <http://nerds.airbnb.com/slides-and-video-from-spike-brehms-tech-talk/>

Dale, T. (2012, May 14). Our approach to routing in Ember. Retrieved March 02, 2015, from [tomdale.net](http://tomdale.net): <http://tomdale.net/2012/05/ember-routing>

Ember.js. (2015a). A framework for creating ambitious web applications. Retrieved April 13, 2015, from Ember.js: <http://emberjs.com/>

Ember.js. (2015b, March 29). Github. Retrieved March 29, 2015, from Ember.js:  
<https://github.com/emberjs/ember.js>

Emberaddons.com. (2015, March 29). Ember addons. Retrieved March 29, 2015, from [Emberaddons.com](http://www.emberaddons.com/): <http://www.emberaddons.com/>

ember-cli-meta-tags. (2015, May 25). Set Meta Tags From Ember Routes. Retrieved May 28, 2015, from Github: <https://github.com/ronco/ember-cli-meta-tags>

Ember-prerender. (2015, May 8). Ember-prerender. Retrieved June 11, 2015, from Github.com: <https://github.com/zipfworks/ember-prerender>

Facebook Inc. (2015b, June 5). Open Graph Stories - Object Properties. Retrieved June 6, 2015, from Facebook.com: <https://developers.facebook.com/docs/sharing/opengraph/object-properties>

Facebook Inc. (2015a, July 18). A JavaScript library for building user interfaces. Retrieved July 18, 2015, from github.io: <https://facebook.github.io/react/>

Google Inc. (2015a, May 25). Making AJAX applications crawlable. Retrieved June 11, 2015, from developers.google.com: <https://developers.google.com/webmasters/ajax-crawling/docs/learn-more>

Google Inc. (2015b, June 7). Meta tags that Google understands. Retrieved June 8, 2015, from support.google.com: <https://support.google.com/webmasters/answer/79812?hl=en>

Hevery, M. (2012, March 13). Databinding in AngularJS. (N. Lafferty, Editor) Retrieved April 18, 2015, from Stack Overflow: <http://stackoverflow.com/a/9693933>

Hevery, M. (2011, January 20). How do Angular.js and Backbone.js compare? Retrieved April 18, 2015, from Quora: <http://www.quora.com/How-do-Angular-js-and-Backbone-js-compare>

Iwa Labs Oy. (2015). Human centric digital solutions. Retrieved August 2, 2015, from Iwa Labs: <http://www.iwa.fi/en>

Iwburk. (2015, March 29). About ember.js. Retrieved March 29, 2015, from Stack Overflow: <http://stackoverflow.com/tags/ember.js/info>

Knockout. (2015, March 29). Knockout. Retrieved March 29, 2015, from Github: <https://github.com/knockout/knockout>

Kukreja, N. (2014, February 22). When does it make sense to use an MVC framework for JavaScript? Retrieved April 20, 2015, from Quora: <http://www.quora.com/When-does-it-make-sense-to-use-an-MVC-framework-for-JavaScript>

Mitchell , D. (2015, March 29). About angularjs. Retrieved March 29, 2015, from Stack Overflow: <http://stackoverflow.com/tags/angularjs/info>

ngmodules.org. (2015, March 28). Angular modules. Retrieved March 29, 2015, from ngmodules.org: <http://ngmodules.org/>

Pastor, P. (2010, March 24). MVC for Noobs. Retrieved March 14, 2015, from Tutsplus: <http://code.tutsplus.com/tutorials/mvc-for-noobs--net-10488>

Prerender.io. (2015, June 10). Prerender - AngularJS SEO, BackboneJS SEO, or EmberJS SEO. Retrieved June 11, 2015, from <https://prerender.io>:  
<https://prerender.io>

Schachinger , K. (2012, May 1). How to use html meta tags. Retrieved April 19, 2015, from searchenginewatch.com: <http://searchenginewatch.com/sew/how-to/2067564/how-to-use-html-meta-tags>

Shrestha, S. (2015, August 16). Seo friendly ember.js. Retrieved August 20, 2015, from Github: <https://github.com/sunil-shrestha/seo-friendly-ember.js>

Stack Overflow. (2015, March 29). About knockout.js. Retrieved March 29, 2015, from Stack Overflow: <http://stackoverflow.com/tags/knockout.js/info>

Synodinos, D. (2013, February 5). Top JavaScript MVC Frameworks. Retrieved May 3, 2015, from InfoQ: <http://www.infoq.com/research/top-javascript-mvc-frameworks>

TodoMVC. (2015). Helping you select an MV\* framework. Retrieved 03 25, 2015, from TodoMVC: <http://todomvc.com>

W3Schools. (2015, May 30). HTML <title> Tag. Retrieved May 30, 2015, from w3schools.com: [http://www.w3schools.com/tags/tag\\_title.asp](http://www.w3schools.com/tags/tag_title.asp)

Waite, P. (2015, March 29). About backbone.js. Retrieved March 29, 2015, from Stack Overflow: <http://stackoverflow.com/tags/backbone.js/info>

Wikipedia. (2015a, August 20). Comparison of web application frameworks. Retrieved August 26, 2015, from Wikipedia:

[https://en.wikipedia.org/wiki/Comparison\\_of\\_web\\_application\\_frameworks](https://en.wikipedia.org/wiki/Comparison_of_web_application_frameworks)

Wikipedia. (2015b, March 17). Comparison of JavaScript frameworks. Retrieved March 21, 2015, from Wikipedia:

[http://en.wikipedia.org/wiki/Comparison\\_of\\_JavaScript\\_frameworks](http://en.wikipedia.org/wiki/Comparison_of_JavaScript_frameworks)

Wikipedia. (2015c, March 22). Search Engine Optimization. Retrieved March 22, 2015, from Wikipedia: [http://en.wikipedia.org/wiki/Search\\_engine\\_optimization](http://en.wikipedia.org/wiki/Search_engine_optimization)

Wikipedia. (2015d, May 16). Semantic URL. Retrieved May 24, 2015, from Wikipedia: [http://en.wikipedia.org/wiki/Semantic\\_URL](http://en.wikipedia.org/wiki/Semantic_URL)

Wikipedia. (2015e, August 11). Single-page application. Retrieved August 15, 2015, from Wikipedia: [http://en.wikipedia.org/wiki/Single-page\\_application](http://en.wikipedia.org/wiki/Single-page_application)

Wikipedia. (2015f, July 29). Web application framework. Retrieved July 24, 2015, from Wikipedia: [http://en.wikipedia.org/wiki/Web\\_application\\_framework](http://en.wikipedia.org/wiki/Web_application_framework)

# Appendix A – Landing page of sample application running at server

The screenshot shows a web browser window with the URL `seo.bitsocan.com/frameworks`. The page has a blue header with the title "Front-end JavaScript Frameworks". On the left, there is a sidebar menu with the following items: "Frameworks", "AngularJS", "Backbone.js", "Ember.js", and "Knockout.js". The main content area contains the following text:

Front-end frameworks are meant to support the client-side presentation and maintain the front-end related codes simple and organized. The term "Front-end frameworks" may refer to design oriented frameworks such as bootstrap, foundation, semantic UI etc. as well as the same term could mean the full-fledged JavaScript frameworks like Angular.js, Ember.js etc. Design oriented frameworks like bootstrap provide easy and more standards-complaint web-design along with different tool and modules like reset-stylesheet, grid and responsive web-design, styling for forms, buttons and navigation menus etc. (Wikipedia, 2015). On the other hand, full-fledged JavaScript frameworks are meant to handle JavaScript related interactivity in application by keeping them structured and organized. Most of the famous JavaScript frameworks follow the MVC pattern, which isolates the data model, user interface and business logic into different layers as Model-View-Controller. This makes the MVC pattern capable of controlling user requests and handle interaction to servers more intelligently and logically.

The Landscape of front-end web development consists of so many libraries, tools, and frameworks that it has become very difficult for anyone to keep up with everything. Developers these days are spoiled with choice when it comes to selecting MV\* framework for structuring and organizing their JavaScript web apps. If one does a quick Google search for "JavaScript MVC Frameworks", it is really easy to become overwhelmed with all the choices that one can choose from. In order to solve this problem, a project "TodoMVC" is available in the Internet which offers the same To-do application implemented using MV\* concepts in most of the popular JavaScript MV\* frameworks of today.

### Front-end javascript frameworks comparision based on Google trend

Interest over time. Web Search. Worldwide, 2010-2015, Computers & Electronics.

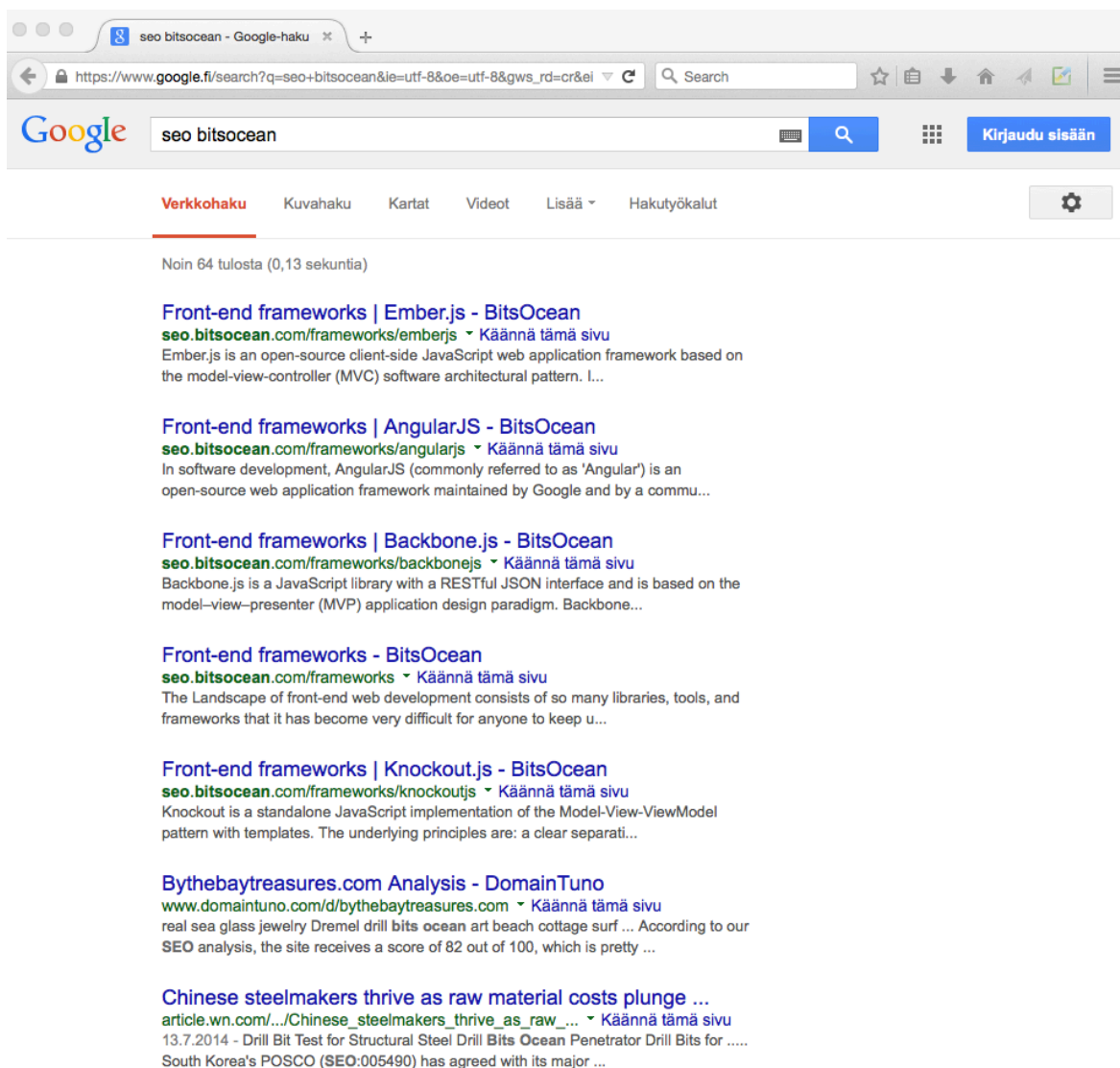
The chart shows search interest over time for four JavaScript frameworks. The x-axis represents time from January 2011 to January 2015, with an 'Average' bar for the period 2010-2011. The y-axis represents relative search interest. The legend indicates: angular.js (blue), backbone.js (red), ember.js (yellow), and knockout.js (green). Angular.js shows a significant upward trend, starting at a low level in 2011 and rising sharply to become the most popular framework by 2015. Backbone.js shows a steady, moderate increase. Ember.js and knockout.js show very low and relatively flat interest throughout the period.

Year	angular.js	backbone.js	ember.js	knockout.js
Average (2010-2011)	High	Medium	Low	Low
Jan 2011	Low	Low	Very Low	Very Low
Jan 2012	Low	Low	Very Low	Very Low
Jan 2013	Low	Low	Very Low	Very Low
Jan 2014	High	Low	Very Low	Very Low
Jan 2015	Very High	Low	Very Low	Very Low

Google

[View full report in Google Trends](#)

## Appendix B – Google search result for sample application

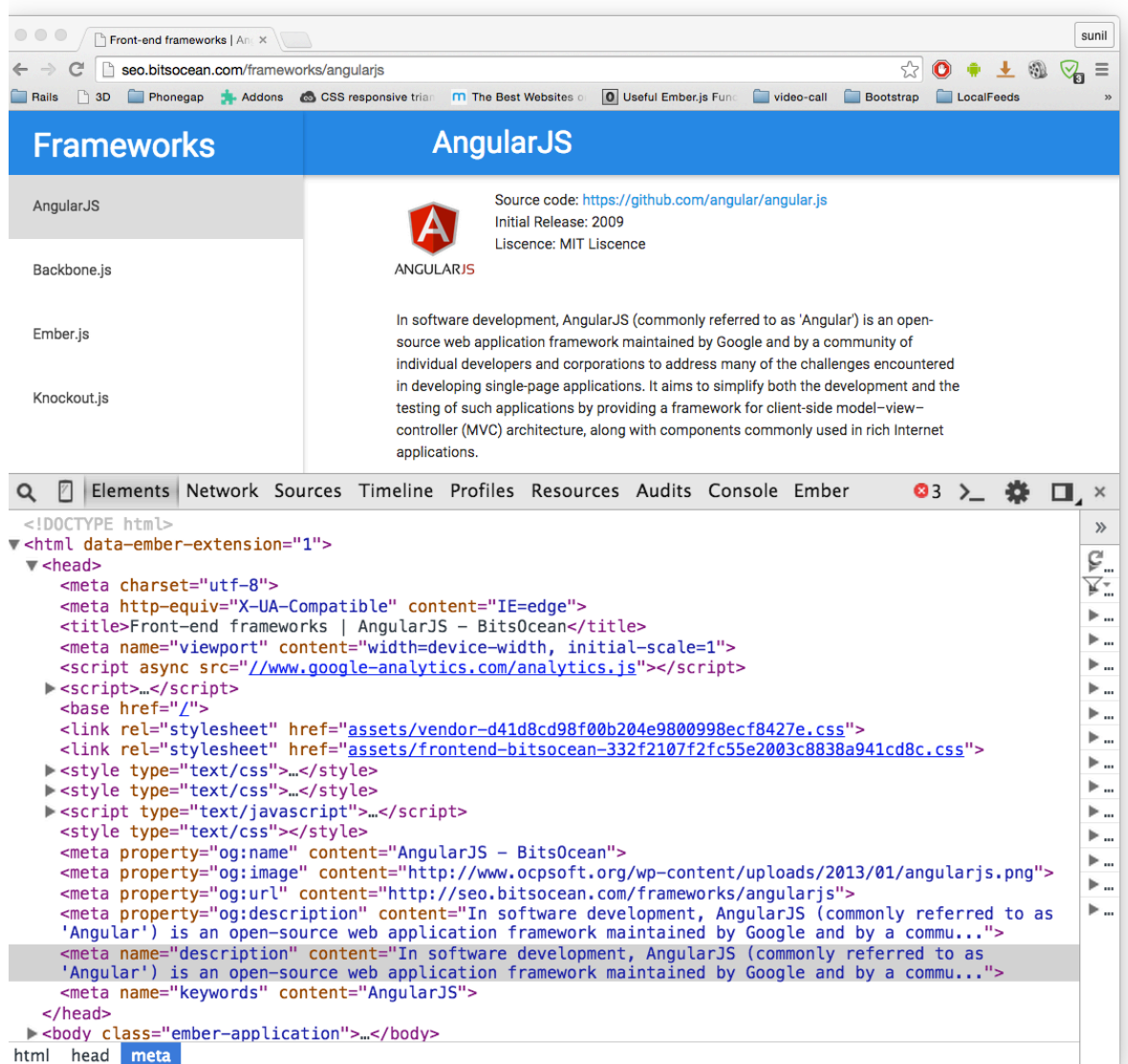


The screenshot shows a Google search page for the query "seo bitsocean". The search results are as follows:

Noin 64 tulosta (0,13 sekuntia)

- Front-end frameworks | Ember.js - BitsOcean**  
[seo.bitsocean.com/frameworks/emberjs](http://seo.bitsocean.com/frameworks/emberjs) ▾ Käännä tämä sivu  
Ember.js is an open-source client-side JavaScript web application framework based on the model-view-controller (MVC) software architectural pattern. I...
- Front-end frameworks | AngularJS - BitsOcean**  
[seo.bitsocean.com/frameworks/angularjs](http://seo.bitsocean.com/frameworks/angularjs) ▾ Käännä tämä sivu  
In software development, AngularJS (commonly referred to as 'Angular') is an open-source web application framework maintained by Google and by a commu...
- Front-end frameworks | Backbone.js - BitsOcean**  
[seo.bitsocean.com/frameworks/backbonejs](http://seo.bitsocean.com/frameworks/backbonejs) ▾ Käännä tämä sivu  
Backbone.js is a JavaScript library with a RESTful JSON interface and is based on the model-view-presenter (MVP) application design paradigm. Backbone...
- Front-end frameworks - BitsOcean**  
[seo.bitsocean.com/frameworks](http://seo.bitsocean.com/frameworks) ▾ Käännä tämä sivu  
The Landscape of front-end web development consists of so many libraries, tools, and frameworks that it has become very difficult for anyone to keep u...
- Front-end frameworks | Knockout.js - BitsOcean**  
[seo.bitsocean.com/frameworks/knockoutjs](http://seo.bitsocean.com/frameworks/knockoutjs) ▾ Käännä tämä sivu  
Knockout is a standalone JavaScript implementation of the Model-View-ViewModel pattern with templates. The underlying principles are: a clear separati...
- Bythebaytreasures.com Analysis - DomainTuno**  
[www.domaintuno.com/d/bythebaytreasures.com](http://www.domaintuno.com/d/bythebaytreasures.com) ▾ Käännä tämä sivu  
real sea glass jewelry Dremel drill bits ocean art beach cottage surf ... According to our SEO analysis, the site receives a score of 82 out of 100, which is pretty ...
- Chinese steelmakers thrive as raw material costs plunge ...**  
[article.wn.com/.../Chinese\\_steelmakers\\_thrive\\_as\\_raw\\_...](http://article.wn.com/.../Chinese_steelmakers_thrive_as_raw_...) ▾ Käännä tämä sivu  
13.7.2014 - Drill Bit Test for Structural Steel Drill Bits Ocean Penetrator Drill Bits for .... South Korea's POSCO (SEO:005490) has agreed with its major ...

## Appendix C – HTML document with Meta tags for each resource (AngularJS).

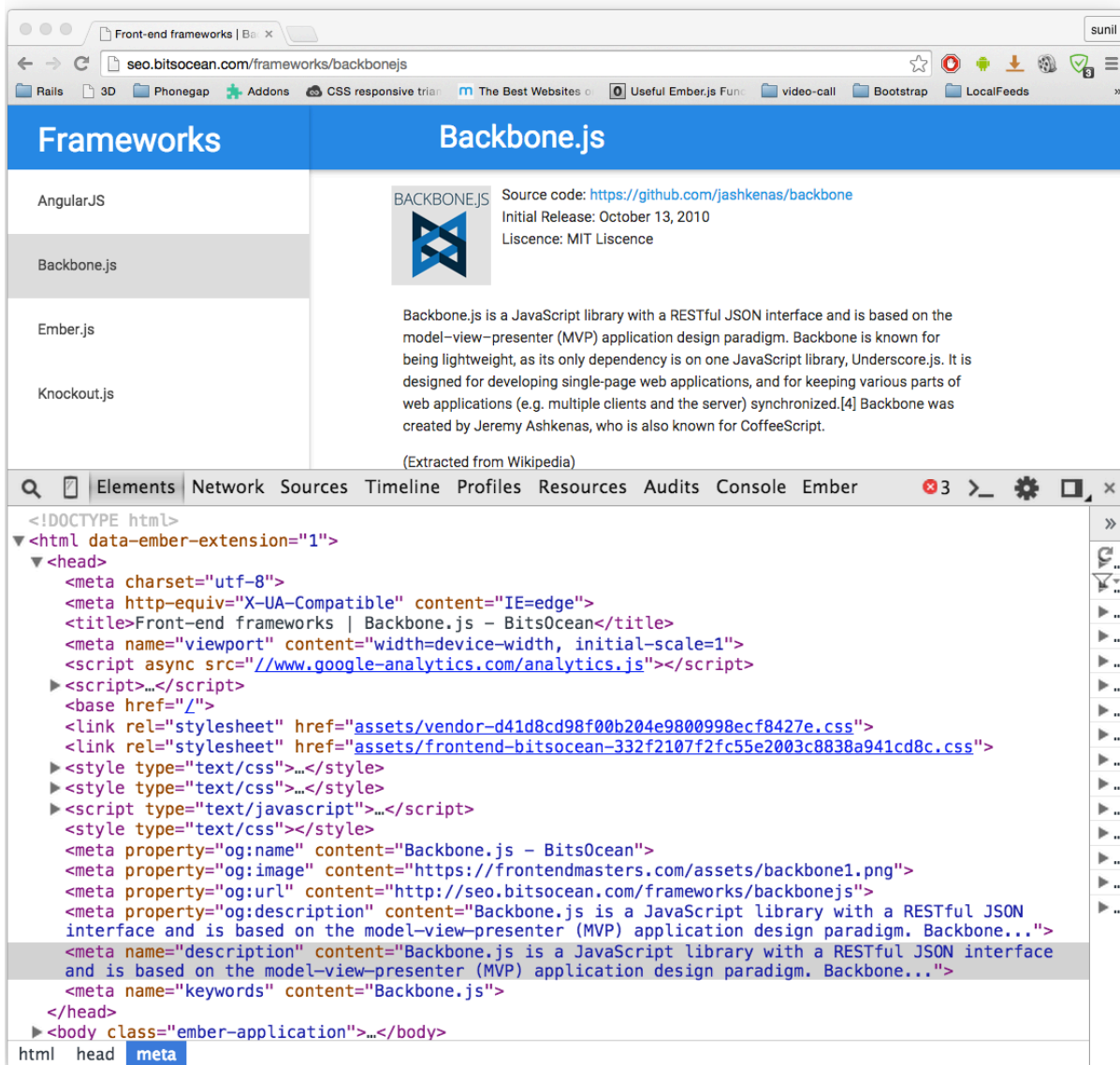


The screenshot shows a web browser displaying the AngularJS page on [seo.bitsocean.com/frameworks/angularjs](http://seo.bitsocean.com/frameworks/angularjs). The page features a blue header with "Frameworks" and "AngularJS". A sidebar on the left lists frameworks: AngularJS, Backbone.js, Ember.js, and Knockout.js. The main content area displays the AngularJS logo, source code link (<https://github.com/angular/angular.js>), initial release (2009), and license (MIT). A paragraph describes AngularJS as an open-source web application framework maintained by Google and a community of developers, used for developing single-page applications with an MVC architecture.

The browser's developer tools are open to the "Elements" tab, showing the HTML structure. The `<head>` section contains several meta tags, including Open Graph tags for name, image, url, and description, and a `<meta name="description">` tag that matches the text on the page. The `<meta name="keywords">` tag contains the text "AngularJS".

```
<!DOCTYPE html>
<html data-ember-extension="1">
  <head>
    <meta charset="utf-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <title>Front-end frameworks | AngularJS - BitsOcean</title>
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <script async src="//www.google-analytics.com/analytics.js"></script>
    <script>...</script>
    <base href="/">
    <link rel="stylesheet" href="assets/vendor-d41d8cd98f0b204e9800998ecf8427e.css">
    <link rel="stylesheet" href="assets/frontend-bitsocean-332f2107f2fc55e2003c8838a941cd8c.css">
    <style type="text/css">...</style>
    <style type="text/css">...</style>
    <script type="text/javascript">...</script>
    <style type="text/css">...</style>
    <meta property="og:name" content="AngularJS - BitsOcean">
    <meta property="og:image" content="http://www.ocpsoft.org/wp-content/uploads/2013/01/angularjs.png">
    <meta property="og:url" content="http://seo.bitsocean.com/frameworks/angularjs">
    <meta property="og:description" content="In software development, AngularJS (commonly referred to as 'Angular') is an open-source web application framework maintained by Google and by a commu...">
    <meta name="description" content="In software development, AngularJS (commonly referred to as 'Angular') is an open-source web application framework maintained by Google and by a commu...">
    <meta name="keywords" content="AngularJS">
  </head>
  <body class="ember-application">...</body>
</html>
```

## Appendix D – HTML document with Meta tags for each resource (Backbone.js).



The screenshot shows a web browser displaying the Backbone.js page on [seo.bitsocean.com/frameworks/backbonejs](http://seo.bitsocean.com/frameworks/backbonejs). The page features a navigation menu with 'Frameworks' and 'Backbone.js'. The main content area includes the Backbone.js logo, source code link (<https://github.com/jashkenas/backbone>), initial release date (October 13, 2010), and license (MIT License). A paragraph describes Backbone.js as a JavaScript library with a RESTful JSON interface, based on the model-view-presenter (MVP) application design paradigm. The developer tools are open to the 'Elements' tab, showing the HTML document structure. The meta tags in the head section are highlighted, including charset, http-equiv, viewport, analytics script, base href, stylesheets, and Open Graph (og) tags for name, image, url, description, and keywords.

```
<!DOCTYPE html>
<html data-ember-extension="1">
  <head>
    <meta charset="utf-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <title>Front-end frameworks | Backbone.js - Bits0cean</title>
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <script async src="//www.google-analytics.com/analytics.js"></script>
    <script>...</script>
    <base href="/">
    <link rel="stylesheet" href="assets/vendor-d41d8cd98f00b204e9800998ecf8427e.css">
    <link rel="stylesheet" href="assets/frontend-bitsocean-332f2107f2fc55e2003c8838a941cd8c.css">
    <style type="text/css">...</style>
    <style type="text/css">...</style>
    <script type="text/javascript">...</script>
    <style type="text/css"></style>
    <meta property="og:name" content="Backbone.js - Bits0cean">
    <meta property="og:image" content="https://frontendmasters.com/assets/backbone1.png">
    <meta property="og:url" content="http://seo.bitsocean.com/frameworks/backbonejs">
    <meta property="og:description" content="Backbone.js is a JavaScript library with a RESTful JSON interface and is based on the model-view-presenter (MVP) application design paradigm. Backbone...">
    <meta name="description" content="Backbone.js is a JavaScript library with a RESTful JSON interface and is based on the model-view-presenter (MVP) application design paradigm. Backbone...">
    <meta name="keywords" content="Backbone.js">
  </head>
  <body class="ember-application">...</body>
```