Node.js in Open Source projects on Github

A literature study and exploratory case study

Kenneth Lewenhagen, Anders Åkesson

Faculty of Computing
Blekinge Institute of Technology
SE-371 79 Karlskrona Sweden

Contact information:
Kenneth Lewenhagen lewenhagen@live.com
Anders Åkesson andy_akesson@me.com

University Advisor:
Krzysztof Wnuk
DIPT Department of Software Engineering

Faculty of Computing
Blekinge Institute of Technology
SE-371 79 Karlskrona Sweden

Internet: www.bth.se
Phone: +46 455 38 50 00

Abstract

This study has been performed with an aim to provide an insight into how Node.js is used and the Node.js technology adaptation in the open source community. This research displays the diversity of Node.js and can inspire the reader to further development or continued research.

Studies into different usages of Node.js have been missing in academic research and therefore this study gives a new, important insight into this technology.

The authors used the exploratory case study methodology. For data collection, the authors created a JQuery and HTML script that fetched the desired dataset from Github and that were used as a static base for the study. Based on the usage areas extracted from the literature study, the authors specified different categories of usage. The dataset was manually investigated and placed into the categories, if they were relevant.

The results show that web applications is by far the most well represented category with over 50% of all usages falling into this category. Network applications and Web servers come in at second and third position with 14% and 13% respectively.

This study provided further categories and the authors could generate a set of diagrams, showing a trend on how the different usage areas changed from 2010 to 2015.

Keywords: Node.js, open source, application, framework

Acknowledgments

We would like to extend our gratitude to our supervisor Krzysztof Wnuk for his invaluable support and feedback from the idea initiation until the conclusion of the thesis. This thesis would not have been possible without Wnuk endless guidance throughout the research. His perpetual energy and enthusiasm in research extremely motivated us in both personal and professional level. The authors would like to mention that it has been a great pleasure for us to have Wnuk as our supervisor since his guidance, valuable suggestions, constructive criticism and incredible patience helped us to successfully complete the thesis in a smooth way.

Table of contents

1.0 Introduction

Javascript is the most popular client-side programming language on the Web today and also the second fastest growing server-side programming language, according to W3Techs[1]. This makes research into Javascript libraries and their adaptation important as this kind of research provides an insight into how the Web evolves and in what ways developers are pushing the boundaries of what is possible.

Node.js was primarily created for real-time web applications employing push technology over websockets. Thanks to the capabilities of Node.js we now have web applications with real-time, two-way connections, where both the client and server can initiate communication, allowing them to exchange data freely. This is in stark contrast to the typical web response paradigm, where the client always initiates communication[4].

One rationale for doing this research is that W3Tech[1] lists Node.js as a web server technology in its rankings. It does not list Node.js in its rankings[2] for most used Javascript libraries[3]. The Node.js usage is however much more diverse than just as a web server[4]. Therefore the authors believe it is relevant to measure the usage of Node.js by other metrics and look at different usage areas. Doing so in a limited environment such as Github will provide a different insight into the actual usage and the adoption of Node.js in the tech community.

Node.js is a diverse technology with many different usages. We believe it is therefore important to explore the way Node.js is used in an open source environment, which is reflective of the entire tech community[15].

The aim and expected outcome of this thesis is to provide the reader with an overview and an insight into the different ways that Node.js is used in open source projects on GitHub. The intended reader is anyone with an interest in knowing the different ways in which Node.js can be used, for example developers. This study provides the reader with trends and diagrams based on our findings regarding Node.js usage spanning from 2010 to early 2015. This is of value for further investigation to some readers as well as the whole Node.js community. The target group is large; all levels of developers can find related projects and scientists as well as the Node.js community may want to see the trend and diagrams for further investigation.

2.0 Background

Node.js was created by Ryan Dahl in 2009 [5], sponsored by Joyent, the firm where Dahl worked. Node.js is an open source, cross-platform runtime environment for server-side and networking applications. Node.js applications are written in JavaScript, and can be run within the Node.js runtime on OS X, Microsoft Windows, Linux and FreeBSD.
Node.js provides an event-driven architecture and a non-blocking I/O API that optimizes an application's throughput and scalability. These technologies are commonly used for real-time applications[5].
According to its own website, Node.js is a platform built on Chrome's JavaScript runtime for easily building fast, scalable network applications. Node.js uses an event-driven, non-blocking I/O model that makes it lightweight and efficient, perfect for data-intensive real-time applications that run across distributed devices.[6].
Node.js allows the creation of web servers and networking tools, using JavaScript and a collection of "modules" that handle various core functionality. Modules handle File system I/O, Networking (HTTP, TCP, UDP, DNS, or TLS/SSL), Binary data (buffers), Cryptography functions, Data streams, and other core functions. Node's modules have a simple and elegant API, reducing the complexity of writing server applications[43].
Node.js has grown to be one of the most used and talked about web technologies in recent years[7]. A simple search on StackOverflow shows over 13 000 questions related to Node.js[8] and the question "How to get started with Node.js" has over 1200 upvotes and an answer with over 2700 upvotes[9] (An upvote is the action of giving an article, post or video on an internet forum or website a positive rating, such as a "thumbs up"). A large number of corporations and organizations use Node.js as part of their core IT infrastructure, these corporations include Yahoo, Uber, New York Times, Dow Jones, PayPal, Microsoft, eBay[10] and Walmart[11].

Originally founded by Tom Preston-Werner, Chris Wanstrath, and PJ Hyett to simplify sharing code, GitHub has grown into the world's largest code host and is now the most used open source community on the web[12]. To date Github has over 6.7 million users and over 1.1 million projects[13]. Open source software has a massive impact on innovation in information technology[14] and is therefore a reflection on the entire tech community[15].
Node.js and GitHub are unified and the development of Node.js is dependent on the GitHub community as all development takes place on the Node.js GitHub repository[16][17].
Projects on GitHub can be accessed and manipulated using the standard git command-line interface and all of the standard git commands work with it. GitHub also allows registered and unregistered users to browse public repositories on the site. Multiple desktop clients and git plugins have also been created by GitHub and other third parties which integrate with the platform.
The site provides social networking-like functions such as feeds, followers, wikis (using wiki software called gollum) and a social network graph to display how developers work on their versions ("forks") of a repository and which fork (and branch within that fork) is newest.
A user must create an account in order to contribute content to the site, but public repositories can be browsed and downloaded by anyone. With a registered user account, users are able to discuss, manage, create repositories, submit contributions to others' repositories, and review changes to code[87].

To summarize the related research work that has been performed into Node.js, there are some work done on performance evaluations, security assessments and communications and networks, however research into the adaptation and usage of the technology in the open source space is not covered. Also the current number of GitHub projects tells us that there are a multitude of ways in which Node.js is being used by the open source community. Therefore, studying Node.js is important and relevant.

3.0 Literature review

3.1 Purpose

The purpose of the literature review is to provide the context for our research and to justify its relevance. Performing a literature review also ensures that this research has not already been conducted and provides some new insight for the research community. The review also shows where our research fits into the existing body of knowledge and enables us to learn from previous experiences. It illustrates how our topic has been studied previously, highlights the potential research gaps and outlines gaps in said previous research.
Furthermore, the review will show that the work is adding to the understanding and knowledge of the field and help to refine and refocus the subject at hand.

3.2 Process & Method

3.2.1 Process

In "Guidelines for Snowballing in Systematic Literature Studies and a Replication in Software Engineering", Wohlin presented guidelines for conducting literature reviews using a snowballing approach. We based our literature review on these guidelines. According to Wohlin, a good start set of papers to review should have the following characteristics [18]:

- Targeting diversity of papers which should cover different communities, years, publishers and authors. To achieve this, it is important to have these covered in the start set.
- Number of papers or size of the start set depends on the size of the area being studied. For example, more specific focus requires fewer papers than a broad area.
- If too many papers resulted due to the general search string formulation, then papers having highly cited and more relevant references may be an alternative to obtain perfect start papers.
- To mitigate the risk of missing relevant papers using slightly different terminology, synonyms can be preferred with keywords derived from research questions.

3.2.2 Inclusion and exclusion criteria

The inclusion and exclusion criteria used in this literature review are outlined below:
A. Exclusion Criteria:
Non-peer reviewed,
Duplicate authors,
Duplicate studies,
Out of scope (i.e. Node.js) and

B. Inclusion Criteria:
Time Frame: 2010 to 2015
Language: English
Title - Is it tentatively a paper to include?
If yes:
Abstract, Conclusion and Keywords are reviewed to answer the questions -
Is Node.js used as an integral part of the performed research? or
Is Node.js described in a comprehensive way in the article?

3.2.3 Search process

We searched four different scientific databases, using the same search string, simply “node.js”.
We used this search string in order to capture as many papers as possible on our chosen subject.
The databases used in our search were Engineering Village, IEEE xplore, Scopus and Web of science.

Engineering Village is a powerful search platform that is essential for researchers, students, and faculty. Using a wide range of resources, including journals, conference proceedings, trade publications, patents, government reports and more. [72]

The IEEE Xplore digital library is a powerful resource for discovery and access to scientific and technical content published by the IEEE (Institute of Electrical and Electronics Engineers) and its publishing partners. IEEE Xplore provides Web access to more than 3-million full-text documents from some of the world's most highly cited publications in electrical engineering, computer science and electronics. [73]

Scopus is the largest abstract and citation database of peer-reviewed literature: scientific journals, books and conference proceedings. Delivering a comprehensive overview of the world's research output in the fields of science, technology, medicine, social sciences, and arts and humanities, Scopus features smart tools to track, analyze and visualize research. [74]

The Web of Science (formerly Web of Knowledge) is today's premier research platform, helping you quickly find, analyze, and share information in the sciences, social sciences, arts, and humanities. You get integrated access to high quality literature through a unified platform that links a wide variety of content with one seamless search. [75]

These databases were chosen based on recommendations by our supervisor. Thanks to the various search functionality of each of these databases we were able to filter out certain attributes from the search and thereby automatically ensure that our exclusion criteria were met to some extent. The reason for excluding papers from before 2010 is that Node.js was released in 2009. A summary of the search process can be found in section 3.2.4 Search results, found below.

3.2.4 Search results

Database	Number of results	Added criteria
Engineering Village	27	> 2009, English
IEEE xplore	22	Standards, Conference Publications, Early access articles, > 2009
Scopus	9	> 2009, Article
Web of science	12	> 2009

Table 1: Database search results

3.2.5 Deriving the Start set of papers

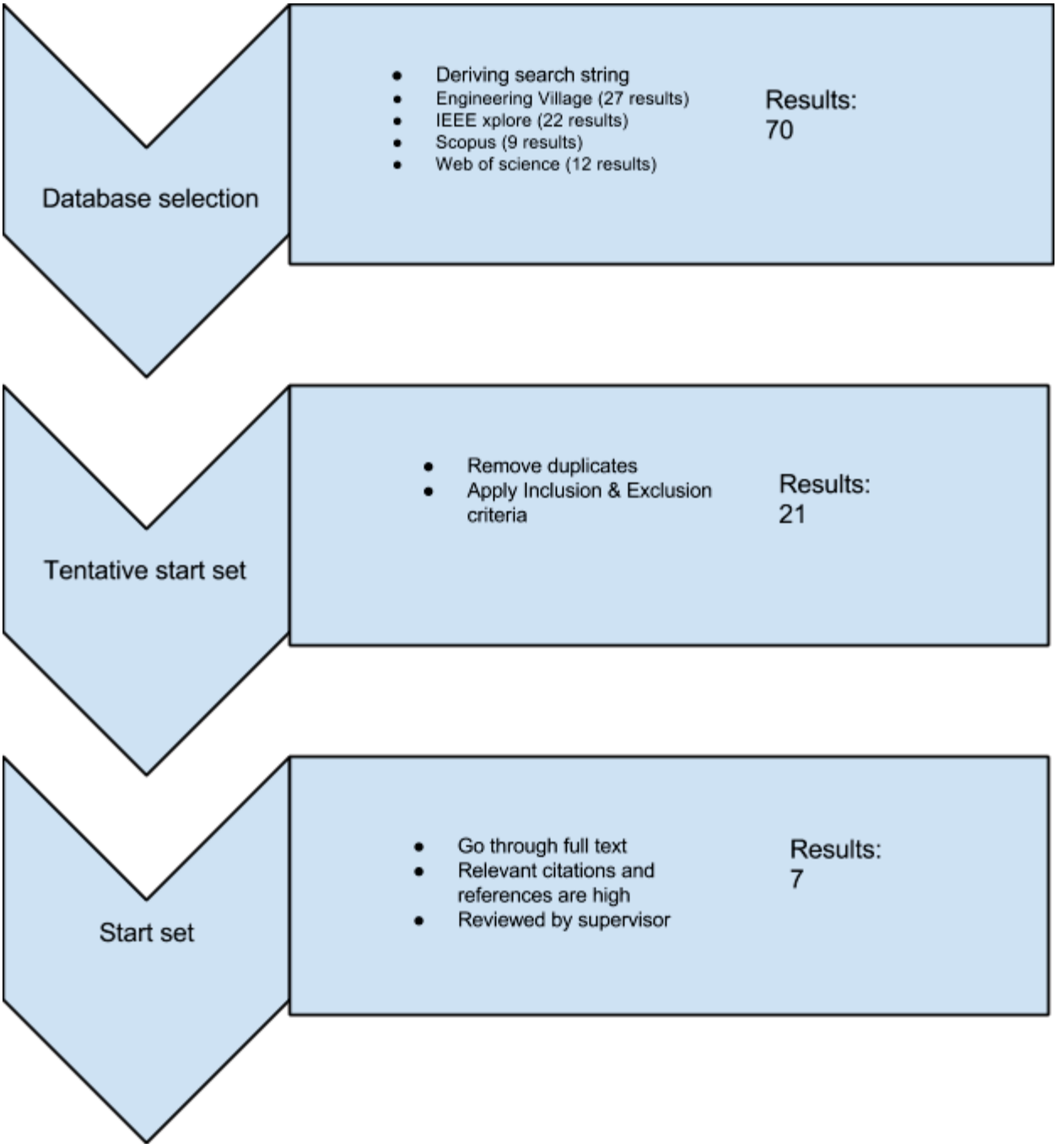

Figure 1: Start set procedure

The total number of results was 70 papers. Out of these, we removed 7 duplicates as per our inclusion and exclusion criteria. We went through the results by reading the title and abstracts. The results that matched our criteria were combined into an Excel document were all duplicates were subsequently filtered out. After this first iteration of filtering we were able to identify 21 unique papers with the potential of being included in our review. On these 21 papers we performed a deeper analysis and review by going through the full text of each paper and out of these 21 papers we identified 9 papers

to be included in our review. If there were studies that could not be decided by abstract and conclusion, then we would go through the full paper and make a decision on inclusion or exclusion.

In the next phase we performed additional filtration by looking at the number of citations and references of each paper candidate. This is because some of the papers were found cited among the identified tentative papers. In this case it is sufficient to take the paper into consideration that has more relevant references and highly cited since snowballing is driven using citations and references. Here it is important to target diversity so after reviewing the full text, a paper that had more relevant citations and/or references was included when compared to papers which have less citations and/or references. This is also done because of the motivation of snowballing that more relevant citations and references in a paper gives the possibility of more coverage of relevant studies[18]. If the citations or references are relevant or not, is justified by reading the title and abstract.

After consulting our supervisor and having him review our selection, we identified 2 more papers for exclusion based on our criteria, leaving us with 7 papers to be the start set.

3.2.6 Limitations of literature study

Our literature study is exposed to the limitation that we have only used one search string in all databases, meaning that our study lacks articles that do not fit into this search query.

Our literature study is also based on performing queries in four different databases and therefore faces the risk of not including articles published in other sources.

3.3 Iterations

3.3.1 Performing forward and backward snowballing in iterations

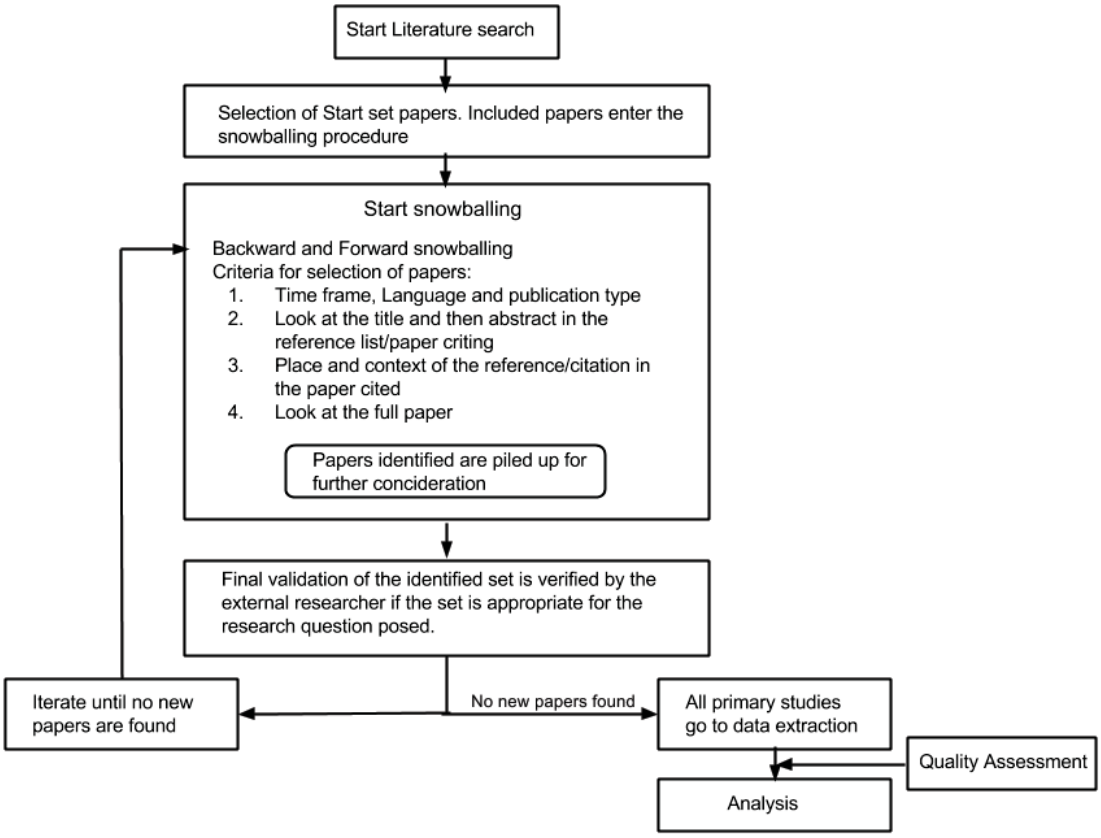

Figure 2: Literature Review using snowballing approach

Backward and forward snowballing was applied on the start set of papers. This resulted in two iterations. First, backward snowballing was performed by looking at the references of the papers from the start set and applying the previously defined Inclusion and Exclusion criteria and adding the Detailed Inclusion Criteria defined in A, 2nd step. Next, forward snowballing was performed. Forward snowballing is looking at the citations of the papers from the start set. Google Scholar was used as a source to identify the citations. The inclusion criteria remained stable between the forward and backward snowballing.
It is unlikely that any research articles on Node.js have been published before 2009 since Node.js was released in 2009 and therefore we have chosen to only include articles published after 2009.

A. Inclusion Criteria:
1st step: Basic Inclusion Criteria

- Time frame: After 2009 to 2015
- Language: English
- Type of publication - peer reviewed

2nd step: Detailed Inclusion Criteria

- Title – is it tentatively a paper to include?
- Publication venue - Is it published in a place where relevant papers may be published?
- Authors - Do we know that the authors have published relevant paper in the area studied before?
- How they were used when referring to them, place and context of the reference
- Abstract - is the paper in scope?

B. Exclusion Criteria:
The articles that don't satisfy the above mentioned requirements would not be considered as primary studies in this work.

3.4 Results & Analysis of the literature review

Here we present the results and our analysis of the literature review.

3.4.1 Start set

Phase 1:
In total, 21 candidates from 70 candidates were identified for inclusion for tentative start set (next phase), after applying inclusion and exclusion criteria.

Phase 2:
7 candidates for inclusion were identified from 21 tentative start set based on a number of relevant citations and references. First going through the title, looking at the relevant study and then at the abstract. Finally, full-text of all the 7 candidates is briefly read before pileup to snowballing.
The 7 candidates are now denoted as C1, C2, C3, C4, C5, C6 and C7 to indicate that they are candidates for inclusion. The 7 candidates are:

ID	Authors	Title	Year published
C1	Kai Lei(1); Yining Ma(2); Zhi Tan(3)	Performance Comparison and Evaluation of Web Development Technologies in PHP, Python, and Node.js	2014
C2	Ojamaa, A.(1); Duuna, K.(2)	Assessing the security of Node.js platform	2012
C3	Shuman Zhao(1); Xiaoling Xia(1); Jiajin Le(1)	A real-time web application solution based on Node.js and WebSocket	2013
C4	Rice, J.L.(1); Phoha, V.V.(1); Cappelaere, P.(2); Mandl, D.(3)	Web farm-inspired computational cluster in the cloud	2011
C5	Griffin, L.(1); Ryan, K.(1); de Leastar, E.(1); Botvich, D.(1)	Scaling Instant Messaging communication services: A comparison of blocking and non-blocking techniques	2011
C6	Karagoez, Mehmet Fatih; Turgut, Cevahir	Design and Implementation of RESTful Wireless Sensor Network Gateways Using Node.js Framework	2014

C7	Chaniotis, I.K., Kyriakou, K.-I.D., Tselikas, N.D.	Is Node.js a viable option for building modern web applications? A performance evaluation study	2014

Table 2: Candidates for inclusion in literature review

3.4.2 Results

In total, 7 papers (marked C1, C2, C3, C4, C5, C6 and C7: see in phase 2) were selected for the start set. 5 were published in conferences and 2 in journals. These papers were written by the total of 23 authors between 2011 and 2014.

3.4.3 Iteration 1

3.4.3.1 Backward Snowballing

During backward snowballing, 181 references and 10 citations were evaluated. 19 references were removed based on the publication year, 148 were removed based on the publication type and 14 removed after abstract or full paper screening.
No papers were therefore left to be included from the first iteration of backward snowballing.

3.4.3.2 Forward snowballing

During forward snowballing, 1 was removed based on the publication language, 1 was removed based on the publication type, 4 removed based on the abstract or full paper screening and 1 turned out to be a duplicate.
Finally 3 papers denoted (C8, C9, C10) were included.

ID	Authors	Title	Year published
C8	W De Groef, F Massacci, F Piessens	A security architecture for server-side JavaScript	2014
C9	J Frizelle	A scalability study into Server Push technologies with regard to server performance	2011
C10	K Fysarakis, D Mylonakis, C Manifavas	Node. DPWS: High performance and scalable Web Services for the IoT	2015

Table 3: Iteration 1 forward snowballing

In total, the first iteration resulted in including 3 papers, denoted C8 and C9 and C10.
1 was a conference paper, 2 were journals. There were a total of 7 authors and the time period that the papers were written in was 2011 and 2015.
The results from the first iteration of forward and backward snowballing are depicted in Figure 3.

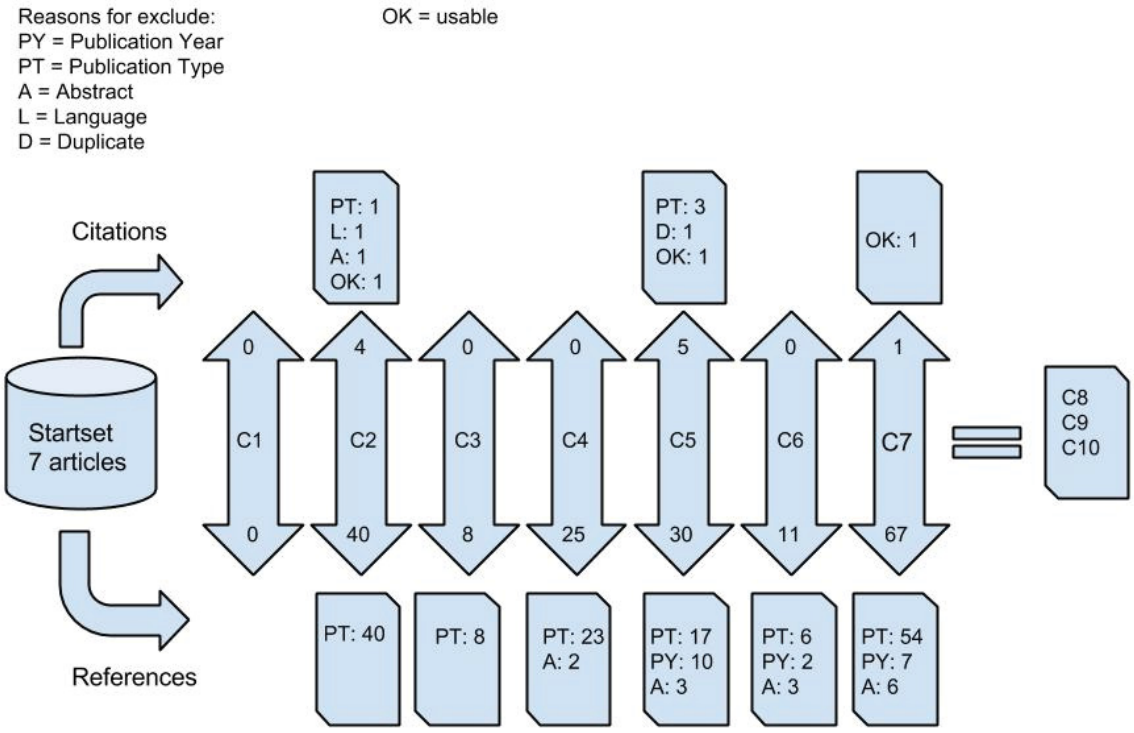

Figure 3, Snowball iteration 1

3.4.4 Iteration 2

3.4.4.1 Backward snowballing

During backward snowballing, 43 references and zero citations were evaluated. 17 references were removed based on the publication year, 13 were removed based on the publication type, 6 were removed based on duplicate article/author and 7 removed after abstract or full paper screening. Finally no papers are included.

3.4.4.2 Forward snowballing

During forward snowballing, no citations were found.
No papers were therefore left to be included from the first iteration of backward snowballing as shown in figure 4.

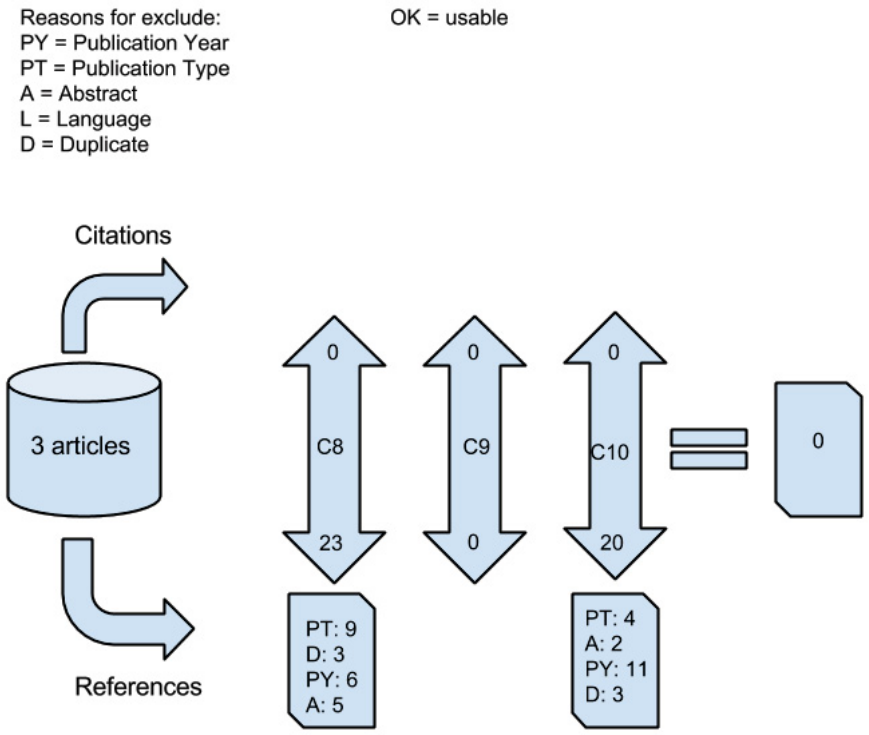

Figure 4, Snowball iteration 2

3.4.5 Data Extraction and Analysis

3.4.5.1 Categorization based on study types

The resulted studies are categorized in two dimensions, namely research methodology (case study, survey, framework and tool proposal) defined by Runeson [19] and type of study (evaluative, proposal, solution, validation) suggested by Wierlinga et al. [20]

Out of the 10 studies, 5 papers (C1, C2, C5, C7 and C9) were evaluative case studies;
C1 [76] compared the performance of Node.js, Python-Web and PHP, with benchmark tests and scenario tests. The experimental results yield some valuable performance data, showing that PHP and Python-Web handle much less requests than that of Node.js in a certain time. For example, in scenario tests the authors choose peak users at 500 to do tests in a “Login” scenario and observed mainly “hits per second”, throughput and average transaction response time as the number of users goes up. The throughput of Node.js was about 5,000,000 byte/s, however, the throughput of PHP and Python-Web are both below 1,000,000 byte/s.
These results demonstrate that Node.js is lightweight and efficient, which is an ideal fit for I/O intensive websites among the three, while PHP is only suitable for small and middle scale applications, and Python-Web is developer friendly and good for large web architectures.

C2 [77] outlines several possible security pitfalls to be aware of when using the Node.js platform and server side JavaScript. For example, the fact that Node.js uses a single threaded event loop based architecture is a major weakness, mainly because any programming mistake throwing an unexpected exception that breaks the loop will terminate the whole application. C2 also describes two discovered vulnerabilities (regular expressions and resource management) and give recommendations for developing and configuring secure and resilient web applications on the Node.js platform.

C5 [80] explores design choices for large scale message delivery to existing Instant Messaging users. In particular the authors explore message throughput, accuracy and server load for several alternative implementation strategies. These strategies focus on approaches to concurrency, with best practice in current and emerging techniques thoroughly benchmarked. Specifically, a conventional Java Executor approach is compared with a functional approach realised through Scala and its Actors framework. A third approach has also been measured - a "non-blocking I/O" based on an alternative to Java Virtual Machine approaches - employing Node.js and Javascript.

C7 [82] examines the implications of end-to-end web application development. The paper describes a distributed architecture, suitable for modern web application development, as well as the interactivity components associated with it. Furthermore, the authors conducted a series of stress tests, on popular server side technologies. The PHP/Apache stack was found inefficient to address the increasing demand in network traffic; while both Nginx and Node.js had no failed requests, Apache started failing requests significantly right after the 19k concurrent connections threshold.
Nginx was found more than 2.5 times faster in input/output (I/O) operations than Apache, whereas Node.js outperformed both. Node.js, although robust in I/O operations and resource utilization, was found lacking in serving static files using its built in HTTP server, while Nginx performed great at this task. The authors found that building cross platform applications based on web technologies, is both feasible and highly productive, especially when addressing stationary and mobile devices, as well as the fragmentation among them. The study concludes that Node.js offers client-server development integration, aiding code reusability in web applications, and is the perfect tool for developing fast, scalable network applications.

C9 [84] summarises the results of a performance analysis of the use of Node.js for the execution of high volumes of long-running requests on a single fixed size server. The Node.js performance is compared to a traditional Java threaded model using the industry standard Tomcat 6 and Tomcat 7 hosting environment. The results indicate that Node.js can provide significant performance improvements in this type of task.

2 papers were proposing solutions through tools (C3 and C8).
C3 [78] proposes a real-time web application solution based on Node.js and WebSocket. C3 also proposes that the new solution is applied in the game “You draw, I guess” to theoretically and practically study the viability of the solution and the advantages compared to traditional solutions. According to the results the authors found that the new solution combining Node.js and WebSocket has significant advantages in easily creating high-performance real-time applications and is faster and more simply compared to the traditional way. Also, the performance is very high.

C8 [83] proposes that “In order to support the least-privilege integration of libraries we develop NodeSentry, the first security architecture for server-side JavaScript.”

3 papers were proposals for frameworks or tools (C4 [79], C6 [81] and C10 [85]).
C4 [79] introduces a web farm-inspired framework for dynamic and concurrent computational processing in the cloud. The authors compare and contrast this with the Hadoop-cloud framework, discuss the main problems associated with their approach, and give suggestions on ways to overcome said challenges. The authors use Node.js to implement the web-inspired framework. They perform experiments to reveal two preliminary results that showcase the framework's functionality and scalability. One, for non-blocking operations, worker nodes which use Node.js servers are significantly faster than those which use traditional servers. In particular, a single Node.js server is (on average) 2.11 times faster than one Ruby Webrick server, and is (on average) 1.88 times faster than two Ruby Webrick servers. Two, the authors find that increasing the number of worker nodes improves overall performance for blocking computational operations. As the number of worker nodes increase, the total execution time decreases exponentially and the number of requests per second increases linearly.

C6 [81] proposes a design and implementation method for RESTful WSN Gateways using Node.js. The gateway is designed as an Embedded Linux device, which can handle multiple accesses with both sensors and cloud servers. All communication APIs residing inside the gateway is designed to be RESTful, HTTP API for communication with cloud servers and Constrained Application Protocol (CoAP) for sensor communication. The proposed approach in this paper provides seamless integration between cloud applications and sensors with its well-defined standard API and minimizes the development time of WSN gateways.

C10 [85] presents Node.DPWS, a novel implementation of the Devices Profile for Web Services (DPWS) based on the Node.js platform. Node.DPWS can be used to deploy lightweight, efficient and scalable Web Services over heterogeneous nodes, including devices with limited resources. The performance of the presented work is evaluated on typical embedded devices, including comparisons with implementations created using alternative DPWS toolkits. The performance assessment revealed that Node.DPWS outperforms the most attractive alternative currently available, the WS4D-JMEDS toolkit; the Node.DPWS device was able to reply to twice the number of requests per second than the second best performing device, which was the WS4D-JMEDS CDC one.

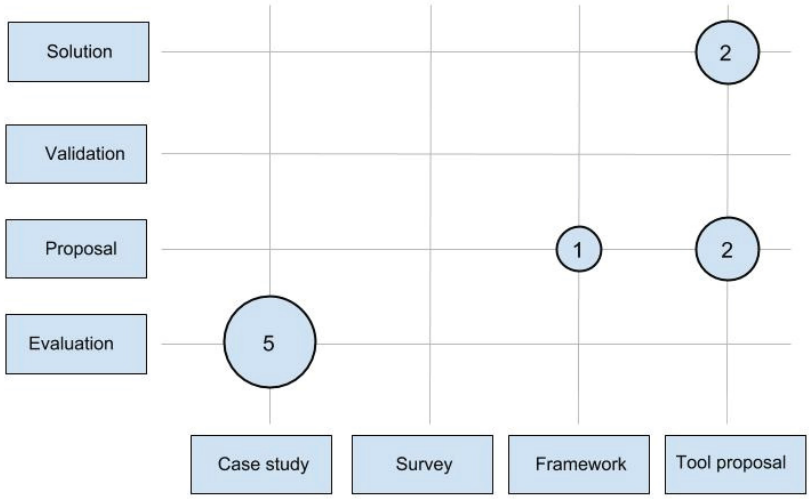

Figure 5, Classification of primary studies

3.4.5.2 Categorization based on study topic

Out of the 10 studies we have identified 6 papers (C1 [76], C3 [78], C4 [79], C7 [82], C9 [84] and C10 [85]) where the focus of the paper was on performance; aspects such as message throughput, accuracy and server load are investigated. Stress-tests and long running performance analysis have been performed as well as benchmark tests and scenario tests.

Performance

C1 [79] compares the performance of Node.js, Python-Web and PHP, using benchmark tests and scenario tests. The experimental results yield some valuable performance data, showing that PHP and Python-Web handle much less requests than that of Node.js in a certain time. In conclusion, the results clearly demonstrate that Node.js is quite lightweight and efficient, which is an ideal fit for I/O intensive websites among the three, while PHP is only suitable for small and middle scale applications, and Python-Web is developer friendly and good for large web architectures. For example, their tests with Node.js's "mean requests per second" is the highest which increases to 3703.5 times per second when the users of Node.js are 100. Meanwhile, the "mean time per request" is the shortest which is 0.27ms. Then the "mean requests per second" slows down and maintains a steady state around 2700. As to PythonWeb, the "mean requests per second" keeps stable around 500 and the highest is 559.42. At this moment, the "mean time per request" is the shortest which is 1.788ms. To PHP, it is also at a peak when users are 100. The "mean requests per second" is 2977.54 at that time. The "mean time per request" is the shortest 0.336ms. With users increasing, the "mean requests per second" decreases to 200 and remains stable. In short, the performance of Node.js is

better than two others at the same number of users. The performance of Node.js is two times larger than PHP on the basic performance comparison and six to seven times larger than Python-Web.

C3 [78] proposes a new real-time web application solution based on Node.js and WebSocket, aiming to significantly enhance real-time performance, and more efficiently use the processing power of the server. According to the authors, their solution combining Node.js and WebSocket has significant advantages in easily creating high-performance real-time applications and is faster and simple compared to the traditional way. Finally, the new solution is applied in the game "You draw, I guess" to theoretically and practically study the viability of the solution and the advantages compared to traditional solutions.

C4 [79] introduces a web farm-inspired framework for dynamic and concurrent computational processing in the cloud. The authors compare and contrast this with the Hadoop-cloud framework, discuss the main problems associated with their approach, and give suggestions on ways to overcome said challenges. The authors performed experiments to reveal two preliminary results that showcase the functionality and scalability of Node.js. One, for non-blocking operations, worker nodes which use Node.js servers are significantly faster than those which use traditional servers. In particular, a single Node.js server is (on average) 2.11 times faster than one Ruby Webrick server, and is (on average) 1.88 times faster than two Ruby Webrick servers. Two, the authors find that increasing the number of worker nodes improves overall performance for blocking computational operations. As the number of worker nodes increase, the total execution time decreases exponentially and the number of requests per second increases linearly.

C7 [82] examines the implications of end-to-end web application development, in the social web era. The paper describes a distributed architecture, suitable for modern web application development, as well as the interactivity components associated with it. The authors conducted a series of stress tests, on popular server side technologies. The PHP/Apache stack was found inefficient to address the increasing demand in network traffic. Nginx was found more than 2.5 times faster in input/output (I/O) operations than Apache, whereas Node.js outperformed both. Node.js, although excellent in I/O operations and resource utilization, was found lacking in serving static files using its built in HTTP server, while Nginx performed great at this task. The study concludes that Node.js offers client-server development integration, aiding code reusability in web applications, and is the perfect tool for developing fast, scalable network applications.

C9 [84] summarises the results of a performance analysis of the use of Node.js for the execution of high volumes of long-running requests on a single fixed size server. The Node.js performance is compared to a traditional Java threaded model using the industry standard Tomcat 6 and Tomcat 7 hosting environment. The results indicate that Node.js can provide significant performance improvements in this type of task. For example, when serving 20,000 concurrent requests in Long Poll requests at various numbers of concurrent requests, the throughput for each server was a total number of 454,000 responses sent and 650 responses per second for Node.js, 380,000 responses sent and 540 responses per second for Tomcat 6 and 270,000 responses and 385 responses per second for Tomcat 7.

C10 [85] presents Node.DPWS, a novel implementation of the Devices Profile for Web Services (DPWS) based on the Node.js platform. Node.DPWS can be used to deploy lightweight, efficient and

scalable Web Services over heterogeneous nodes, including devices with limited resources. The performance of the presented work is evaluated on typical embedded devices, including comparisons with implementations created using alternative DPWS toolkits.

Security assessment
2 papers (C2 [77] and C8 [83]) focused on security assessment.
C2 outlines several possible security pitfalls to be aware of when using Node.js platform and server side JavaScript. The authors also describe two discovered vulnerabilities and give recommendations for developing and configuring secure and resilient web applications on the Node.js platform.
C8 proposes that "In order to support the least-privilege integration of libraries we develop NodeSentry, the first security architecture for server-side JavaScript".

Communication and network
The 2 remaining papers (C5 [80] and C6 [81]) focused on communication and networks.

C5 mentions that their "work explores design choices for such a service: large scale message delivery to existing Instant Messaging users". In particular the authors explore message throughput, accuracy and server load for several alternative implementation strategies. The authors summary show that a simple change of programmatic style can result in a more stable and controlled approach, guaranteeing a baseline QoS (Quality of Service) for operators. The power of the non-blocking approach, championed by Node.js, showed that mass message delivery can not only be accurate, but timely as well.

In C6, a design and implementation method for RESTful WSN Gateways is proposed using Node.js. The gateway is designed as an Embedded Linux device, which can handle multiple accesses with both sensors and cloud servers. All communication APIs residing inside the gateway is designed to be RESTful, HTTP API for communication with cloud servers and Constrained Application Protocol (CoAP) for sensor communication. The proposed approach in this paper provides seamless integration between cloud applications and sensors with its well-defined standard API and minimizes the development time of WSN gateways. The authors used a total number of 10000 packets for each test with different concurrency levels, and they measured the average response time per request and number of lost packets for each request. For all their test scenarios, the number of lost packets is zero.

ID	Study topic	Mentioned in studies
T1	Performance	C1, C3, C4, C7, C9, C10
T2	Security	C2, C8
T3	Communication/Networks	C5, C6

Table 3: Categorization based on study topic

3.4.5.3 Analysis
From the analysis, we can conclude that a significant focus among the research papers on Node.js is about performance evaluation. As Node.js has aimed to improve performance in web applications and claims to be "efficient and perfect for data-intensive real-time applications"[6], it is natural that

performance evaluations on the technology is performed by the research community. The performances covered in the articles are evaluations on:

message throughput which is the rate of successful message delivery over a communication channel. The data these messages belong to may be delivered over a physical or logical link, or it can pass through a certain network node.[21]

Accuracy and server load which is the process of putting demand on a system or device and measuring its response. Load testing is performed to determine a system's behavior under both normal and anticipated peak load conditions.[22]
C5 [80] explored *message throughput, accuracy and server load* for several alternative implementation strategies. The authors of C5 examined the characteristics of blocking (both conventional and functional) and non-blocking I/O, taking a popular service as a domain for comparison. The result is that "The power of the non-blocking approach, championed by the emerging Node.js, showed that mass message delivery can not only be accurate, but timely as well."

Stress-tests which is a software testing activity that determines the robustness of software by testing beyond the limits of normal operation.[23]
C7 [82] conducted a series of stress tests, on popular server side technologies. Nginx was found more than 2.5 times faster in input/output (I/O) operations than Apache, whereas Node.js outperformed both. Node.js, although excellent in I/O operations and resource utilization, was found lacking in serving static files using its built in HTTP server, while Nginx performed great at this task. The study concludes that Node.js offers client-server development integration, aiding code reusability in web applications, and is the perfect tool for developing fast, scalable network applications.

Long running performance analysis (Soak testing) which involves testing a system with a significant load extended over a significant period of time, to discover how the system behaves under sustained use.[24]
C9 [84] summarises the results of a performance analysis of the use of Node.js for the execution of high volumes of long-running requests on a single fixed size server. The Node.js performance was compared to a traditional Java threaded model using the industry standard Tomcat 6 and Tomcat 7 hosting environment. The results indicate that Node.js can provide significant performance improvements in this type of task.

Benchmark tests which is the act of running a computer program, a set of programs, or other operations, in order to assess the relative performance of an object, normally by running a number of standard tests and trials against it. [25]
Scenario tests which is a software testing activity that uses scenarios: hypothetical stories to help the tester work through a complex problem or test system. The ideal scenario test is a credible, complex, compelling or motivating story the outcome of which is easy to evaluate. [26]
C1 [76] used *benchmark tests* and *scenario tests* and the experimental results yield some valuable performance data, showing that PHP and Python-Web handle much less requests than that of Node.js in a certain time. The results clearly demonstrate that Node.js is quite lightweight and efficient, which is an ideal fit for I/O intensive websites among the three.

Missing from these performance evaluation studies are studies on spike testing, observing the behaviour of a system when suddenly increasing the load generated by a very large number of users, and studies on configuration testing to determine the effects of configuration changes to the system's components on the system's performance and behaviour.

As the technology grows and is used in a larger scale, security assessments will become more important to ensure that security pitfalls are avoided. It is interesting to note that Ojamaa and Duuna concludes that “it (Node.js) should be avoided in security critical applications”. The authors do, however, also note that “As the project matures, the security of the platform is also expected to improve over time”[27].

We note that we have been unable to find papers focused on the Node.js open source community.

4.0 Exploratory case study

4.1 Research question

What are the most common usages of Node.js in open source projects on GitHub?
By usage, the authors focuses on how Node.js is used and in what way.
Usages are classified into the following categories:

- Web Applications
- Web Servers
- Network applications (Websocket)
- Security

The categories are based on the literature review results where the identified papers were classified into four distinct categories based on study focus.

1. **CAT1: Web applications** are covered primarily in papers C3 [78], C7 [82] and C10 [85]. Paper C3 proposes a solution for a new real-time web application solution based on Node.js and WebSocket. Paper C7 examines the implications of end-to-end web application development. The paper describes a distributed architecture, suitable for modern web application development, as well as the interactivity components associated with it. Paper C10 presents Node.DPWS, a novel implementation of the Devices Profile for Web Services (DPWS) based on the Node.js platform.
2. **CAT2: Web servers** are covered primarily in papers C1 [76], C4 [79], C6 [81] and C9 [84]. Paper C1 proposes that there is often a dual impact on web server performance, from the calculation, and from the number of users. Their experiment has taken each of these effects in account. Paper C4 introduces a web farm-inspired framework for dynamic and concurrent computational processing in the cloud. The authors compare and contrast this with the Hadoop-cloud framework, discuss the main problems associated with their approach, and give suggestions on ways to overcome said challenges. Paper C6 describes a design and implementation method for RESTful WSN Gateways is proposed using Node.js. C9 summarises the results of a performance analysis of the use of Node.js for the execution of high volumes of long-running requests on a single fixed size server.
3. **CAT4: Websocket** is covered in papers C5 [80] and partially in paper C3 [78] as well (mentioned under CAT1). We will call this category Network applications since they are closely related technologies. Paper C5 explores design choices for large scale message delivery to existing Instant Messaging users. In particular the authors explore message throughput, accuracy and server load for several alternative implementation strategies.
4. **CAT5: Security** is covered in papers C2 [77] and C8 [83]. Paper C2 describes two discovered vulnerabilities and give recommendations for developing and configuring secure and resilient web applications on the Node.js platform. Paper C8 proposes the first security architecture for server-side JavaScript and that a useful policy would be to block specific clients from accessing specific files via the web server.

ID	Study focus	Mentioned in studies
F1	Web Applications	C3, C7, C10
F2	Web Servers	C1, C4, C6, C9
F3	Network applications (Websocket)	C5
F4	Security	C2, C8

Table 4: Categorization based on study focus and literature review results.

This has served as the start set for our study. This list of categories has expanded throughout our research as we have found more usages of Node.js.

4.1.1 Definition of active repositories

A repository that has no commits newer than 6 months (since 2015-04-17) is deemed inactive. This does not affect the inclusion of the repository in this study but it is a valuable metric for our analysis. This definition of inactive repositories is our own.

4.1.2 Definitions of categories

We will use these definitions as a basis for categorizing the repositories gathered in our dataset from Github;

Initial list of categories

CAT1: Web applications: Any software that runs in a web browser. Created in a browser-supported programming language (such as the combination of JavaScript, HTML and CSS) and relies on a web browser to render the application. [88] In this context, all candidates that are using Node.js in some integral way and are web applications falls under this category.

CAT2: Web servers: Information technology that processes requests via HTTP, the basic network protocol used to distribute information on the World Wide Web. Can refer either to the entire computer system, an appliance, or specifically to the software that accepts and supervises the HTTP requests. [89]

CAT3: Network applications: In computer networks, networked computing devices pass data to each other along data connections.[90] The projects that will fall under this category are network applications or are using websocket (a protocol providing full-duplex communications channels over a single TCP connection) in some way.

CAT4: Security: Authorization (specifying access rights to resources related to information security and computer security in general and to access control in particular)[91], authentication (the process of determining whether someone or something is, in fact, who or what it is declared to be)[92] and encryption (software that can encrypt and decrypt data)[93].

During the course of our study we have discovered a further set of categories which are defined below;

Extended list of categories

CAT 5: Test: Software tools and frameworks (separate from the software being tested) to control the execution of tests and the comparison of actual outcomes with predicted outcomes.

CAT 6: Command line interface: A user interface to a computer's operating system or an application in which the user responds to a visual prompt by typing in a command on a specified line, receives a response back from the system, and then enters another command.

CAT 7: Object-relational mapping (ORM): A programming technique for converting data between incompatible type systems in object-oriented programming languages. This creates, in effect, a "virtual object database" that can be used from within the programming language.

CAT 8: Operating systems (OS): Software that manages computer hardware and software resources and provides common services for computer programs. The operating system is an essential component of the system software in a computer system.

4.2 Design for the empirical part

Our case study has been performed as an exploratory case study. An exploratory case study is initial research that tries to look for patterns in the data and come up with a model within which to view this data[94].
We created a script for collecting the dataset by performing advanced searches using the Github API. The performed search was aimed towards revealing the 1000 most popular, i.e. has the most stars, since we have discovered that "stars" is the best measurement for popularity on Github[28][29], containing the keyword “node.js”. After some initial analysis and discussion with the supervisor, it was decided to analyze the first 500 repositories.

The advanced search, **"node.js" created:>20100101**, i.e. search all repositories including Node.js, created after 20100101, was performed on Github[61]. The resulting list has been sorted by most "stars", in descending order. We have collected the title and date created of each repository and a URL[30] to the repository. We have stored the results as a static html page to ensure that the result set is constant and can not be manipulated and to ensure reliability.
The reason for excluding results from before 2010 is because Node.js was not released until 2009.

The full script of the design can be found in Appendix A. The script constructed uses the technologies HTML, JQuery and Javascript.
The search string[1] is based on Github's own query documentation API [31]. The value at “page=1” increases with 1, in 10 iterations, because the result could only contain at most 100 results. With the solution it queries 10 times and gets 100 results each time. The next iteration is started when the current query has been parsed in the success callback to ensure that the order of the query-results will remain trustworthy and accurate.

[1] https://api.github.com/search/repositories?q=node.js+created:>2010-01-01&sort=stars&order=desc&per_page=100&page=1

4.2.1 Inclusion and Exclusion criteria

A. Inclusion Criteria:
1st step (excluded automatically by the script)

- Title and/or README[32] file includes “node.js”
- Time frame: Created after 20100101

2nd step

- Language: English
- Title: is it tentatively a repository to include?
- README: is Node.js used as an integral part in the repository?

B. Exclusion Criteria and Rationale:

- Node Package Manager (NPM)[33]: The repository only uses the NPM for installation and does not use Node.js in any other way. The installation phase is not important to us, it shows one way of using Node.js but it does not provide any real usage for the application as such.
- The repository can not just be a fork[34] of Node.js, since a fork is simply a copy of the repository.
- Duplicates.

4.3 Analysis procedure

On the 500 repositories revealed by the advanced search on GitHub we have identified what functionality each repository provides by reading the documentation in the README file of each repository and, if it was deemed necessary, downloaded the repository and tested the functionality. We have followed the inclusion and exclusion criteria stated in 4.2.1 for each repository. The list of repositories can be found at[2].

The data collected is a collection of different usages for Node.js. These usages have been sorted in an initial number of categories, consisting of Web Applications, Web Servers and Network applications. This list of categories has been expanded as more usages were found. We have kept the categories on a high level to minimize the risk of having too many specific categories. We set the maximum limit of number of categories to ten.

4.4 Synthesis procedure

The synthesis was conducted by gathering the data from the analysis. By analysing the gathered data we have been able to provide an answer to our research question and engage in further discussion on what the results actually mean.

4.5 Limitations

The limited amount of time available to this research only allows us to investigate a limited number of repositories and therefore the risk remains that they can be incomplete.

Regarding duplicates the study is exposed to the risk that we have been unable to identify potential duplicates as it is possible that a repository has been forked and then renamed.

[2] The full list of repositories can be found here: http://www.student.bth.se/~kele12/nodejs/

4.6 Validity discussion

We have identified validity threats based on the categorization introduced by Wohlin et al. [64];
Conclusion validity. This validity is concerned with the relationship between the treatment and the outcome. This study is exposed to the validity threat of reliability of measures, as our categorization of usages involves human judgement.
Internal validity. If a relationship is observed between the treatment and the outcome, we must make sure that it is a causal relationship, and that it is not a result of a factor of which we have no control or have not measured.
The result set from our search could have changed if we had been forced to perform the search on separate dates. To avoid this we have written a script to collect the necessary number of repositories for the entire study in one sitting. This allowed us to gather all results at one specific date and time and thereby eliminating the risk for alterations to the result set entirely.
The direction of a project might change over time and the description of said project in its repository will subsequently be changed as well. This is a minor threat due to the limited time frame of our research because the risk is low that a project could have changed its content and direction to the degree that we would have had to categorize it differently.
Construct validity. This validity is concerned with the relation between theory and observation. If the relationship between cause and effect is causal, we must ensure two things: (1) that the treatment reflects the construct of the cause well and (2) that the outcome reflects the construct of the effect well.
This study is exposed to the design threat of inadequate preoperational explication of constructs. We have therefore maintained a clear and obvious research question. This study is also exposed to the mono-method bias threat since we have only used research data gathered from one source, the source being GitHub.
To avoid the threat of hypothesis guessing we have not made an hypothesis for this study. To avoid the threat of experimenter expectancies we have kept close collaboration with our supervisor to keep our own views unbiased.

5.0 Results

Out of the 500 repositories included in the study, 101 were excluded as per our exclusion criteria. The number of repositories that remain after applying the exclusion criteria is therefore 399. A summary of how these 399 repositories actually use Node.js is displayed in the diagram in Figure 6.
Note that a repository can belong to more than one category.

In the sections 5.3 to 5.7 there are diagrams of a certain year followed by an analysis of that year.

5.1 Result

Category	No. of repositories
Web Applications/frameworks	247
Web Servers	62
Network applications	70
Test	31
Security	26
Command line interface	25
ORM	7
OS (Operating System)	1

Table 5: Total number of repositories in all categories

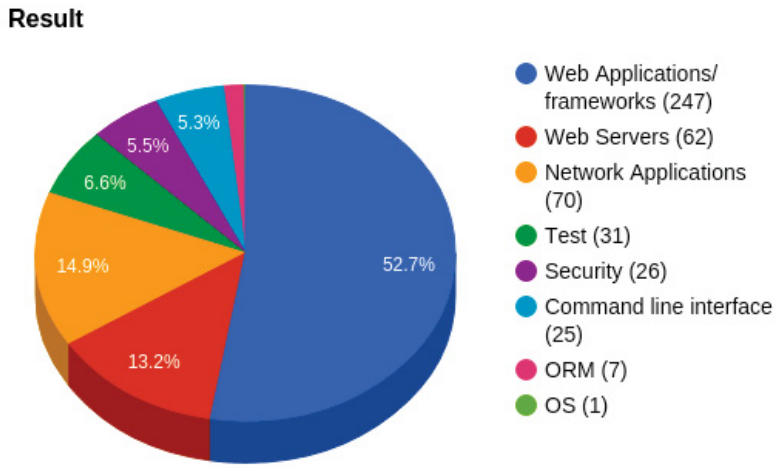

Figure 6: Result

Table 5 and figure 6 shows the total number of repositories included in the study and how the different usages are divided into the different categories.

5.2 Analysis of the category summary

The total category summary shows that a clear majority of the projects investigated belongs in the categories derived from the literature review. The study provided new categories which represents almost fifteen percent (15%) of the total projects investigated.

The results support that Node.js is indeed being used to a great extent in ways for which it was designed, such as in web applications, network applications and web servers. However the way that the technology is being used also goes beyond the usages for which the technology was created; test frameworks and tools, security focused projects and command line interface tools make up over 17% of the total usages.

Web applications come in at over 50% of projects and as this is a broad category and Node.js is a technology aimed to solve problems in this scope [6] this result would have to be seen as logical. Network applications and web servers take almost an equal part of the pie chart, coming in at 14% and 13% respectively. This exemplifies the notion that Node.js is perceived as a very useful technology in both of these areas of usage [71].

6% of the investigated projects have been built with the intention of being used as test frameworks and tools, 5% focused on security related issues, such as authentication and authorization, and a further 5% focused on command line interface tools. By making this observation we can come to the conclusion that the usages of Node.js show a great diversity and a great many different projects use the technology in a vast variety of ways.

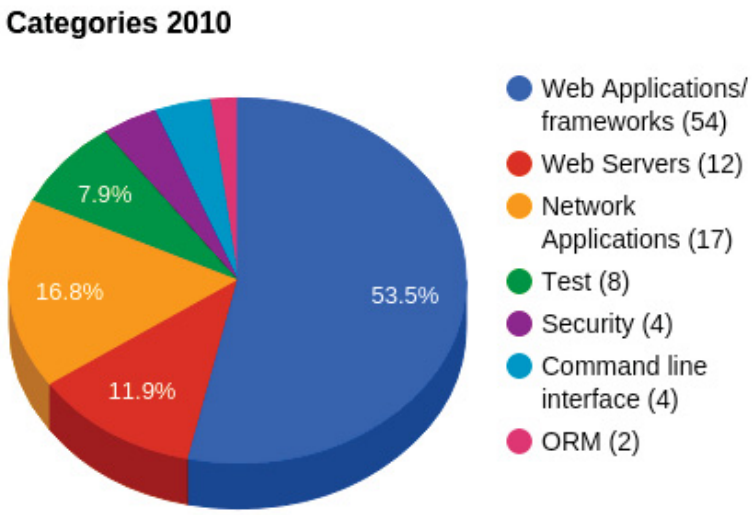

Figure 7: Category summary 2010

Figure 7 shows the total number of repositories included in the study, created in 2010, and how the different usages are divided into the different categories.

5.3 Analysis of 2010

In 2010, Node.js was fairly new on the open source market. As seen in Figure 7, the focus was not so much on security as on testing. Compared to the initial start set of categories there were some new usage areas introduced as projects on Github: Test, Command line interface and ORM.

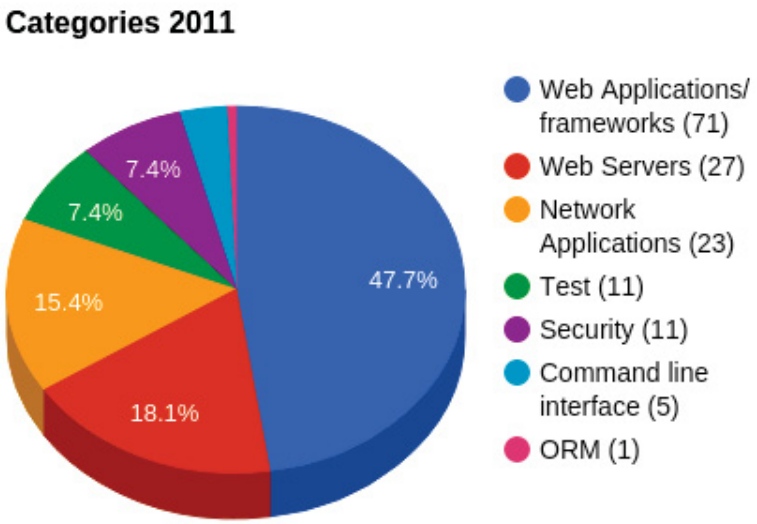

Figure 8: Category summary 2011

Figure 8 shows the total number of repositories included in the study, created in 2011, and how the different usages are divided into the different categories.

5.4 Analysis 2011

In 2011, the npm package manager [35] was introduced to Node.js and made it easier for developers to contribute and to install/uninstall packages and libraries. Joyent also worked with Microsoft to develop a native version of Node.js for Windows [36]. This could contribute to the way in which the distribution of usage areas are more leveled than previous years, with more breadth and greater diversity of developers and an easier management of libraries and packages. This could also contribute to the drop in the Web applications and frameworks category.

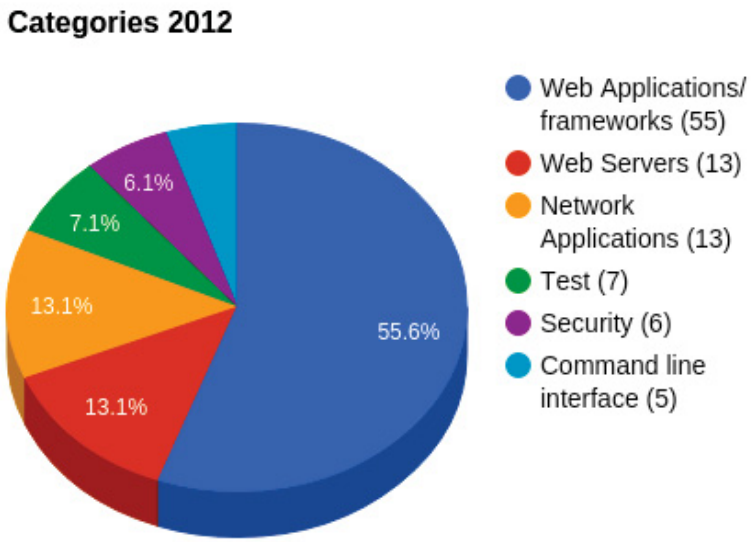

Figure 9: Category summary 2012

Figure 9 shows the total number of repositories included in the study, created in 2012, and how the different usages are divided into the different categories.

5.5 Analysis 2012

Figure 9 shows a major increase in the number of ongoing projects with a focus on Web applications and frameworks. The usage areas regarding command line interface has also gained importance in 2012. Projects using Node.js as a Web server has decreased, while projects that focuses on Command line interface has increased. One reason for the increased usage of Node.js in Web applications could be that in 2011, HTML5 became finalized [37] and the result of that could be an increased interest for Web applications, due to the ease and comparability of JavaScript and HTML5.

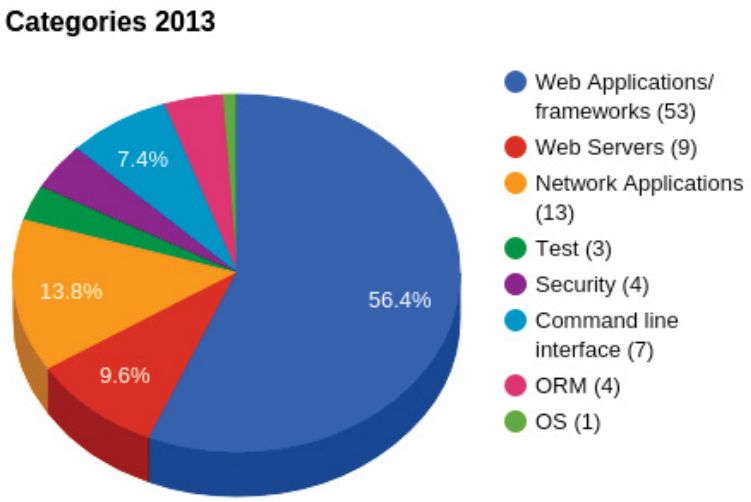

Figure 10: Category summary 2013

Figure 10 shows the total number of repositories included in the study, created in 2013, and how the different usages are divided into the different categories.

5.6 Analysis 2013

In 2013, not much has changed since the previous year. Projects with a focus on Web applications has increased as well as Command line interface. This year, the last category was discovered, Operating Systems (OS), which contains a project where the focus is on building an entire operating system with Node.js.[38]

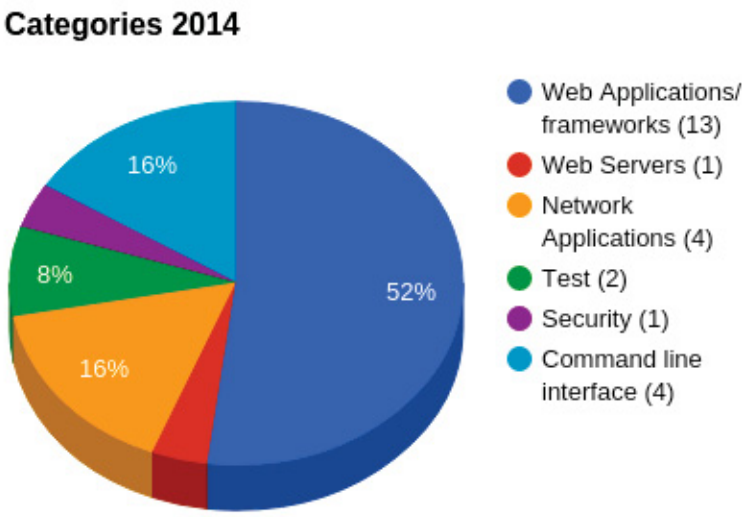

Figure 11: Category summary 2014

Figure 11 shows the total number of repositories included in the study, created in 2014, and how the different usages are divided into the different categories.

5.7 Analysis 2014

In 2014, Figure 11 shows a stable majority of Web applications. Instead there was a major increase in projects with a focus on the Command line interface. The Web server category falls short compared to previous years and the reason for that could be that perhaps the limits to how much Node.js could provide as a Web server has been reached and that the developers focused more on the actual applications using web servers, regarding the significant increment of the category Network applications.

5.8 Analysis of 2015 and discussion on future trends

As seen in Figure 12, there have not been nearly as many projects created on Github, that matches this study's criteria, as in previous years. One reason for this is that the study was done in May 2015 but aside from that aspect, the number does not match the amount of projects created in previous years. Based on the diagrams from previous years, a certain trend can be hypothesized.
The authors will not take the categories OS and ORM in consideration in the analysis, because the study resulted in only 1 project in OS and 7 in the ORM category.

Web applications and frameworks are roughly 50% off analyzed project and have held that status through all years, since 2010. The authors believe that the percentage will stand, this year and in the future, as long as there is a need for web applications and web based solutions.

Web servers has gone down the last few years from 18.1% in 2011 to around 3-4% in 2014. The future leaves room for speculation and one possibility to the long downturn of Web servers could be that Node.js's potential regarding web servers has reached its peak and is as good as it can get, or that the developers and end-users are satisfied with the evolvement. Another possibility is that Node.js and Joyent has lost a lot of contributors from the open source community to io.js. It is, however, beyond the scope of this paper to investigate whether that is actually the case.

Command line interface projects has had a steady upgoing percentage, starting with approximately 4% in 2010 and ends up with 16% in 2014. The command line interface (CLI) is a useful tool, not only regarding Node.js, but also overall for developers. A conclusion of the constant upgoing percentage could be that, aside from the declining number of created projects, a lot of developers uses and favors a CLI[39] and are dedicated to modifying and evolving existing ones, now and in the future.

Network applications has held a steady 13-16% throughout the years. Following that streak, it is possible, and likely, that it will keep having that room. To easily build network applications was one of the reasons for creating Node.js in the first place.[40]

Test frameworks and tools have also held a steady percentage, around 7%, apart from a dip in 2013. A reason for the continuous usage of Node.js in projects with a focus on test could be that almost all applications and software needs to be tested. As new software is created and evolves, the tests need to follow. Some test frameworks and test tools can certainly be reused, but with everything leaning more on automated tests[41], new test tools and frameworks needs to be created.

Security has swayed between approximately 3-4% and peaked at 7.4% in 2011. A reason for this can be, as for the test-category, that new projects needs a different type of security. Since the topic is rather important and has been important for a long time, there will most certain be a continuous development of projects focused on security.

5.9 Trends

5.9.1 Number of created repositories

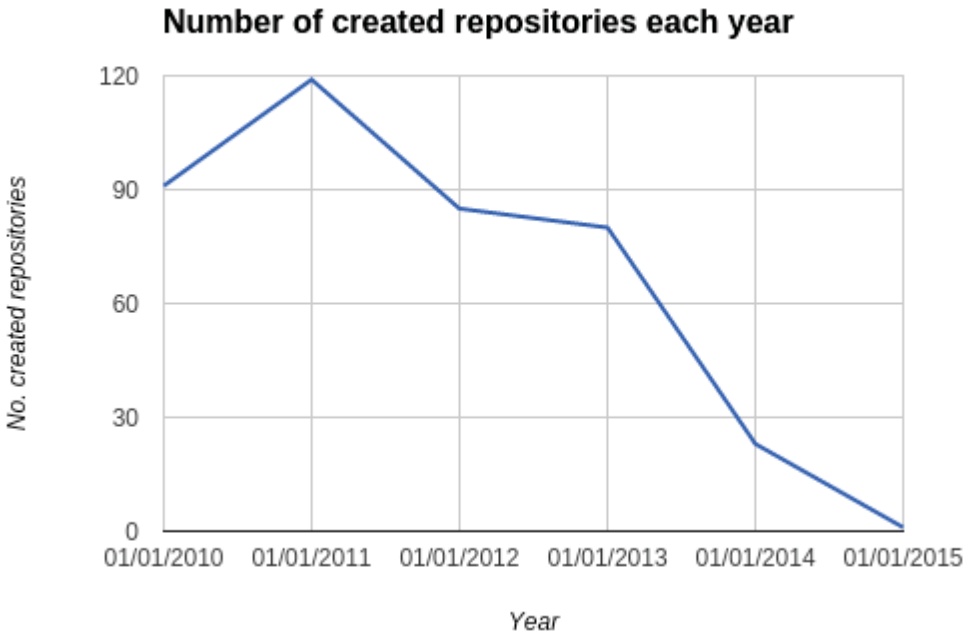

Figure 12. Number of created repositories each year

Figure 12 depicts the number of created repositories each year. "Created" means that a repository has been initialized and uploaded to Github.
The y-axis shows the number of created repositories and the x-axis shows the year. The start date for inclusion into this study is January 1st 2010.
It is important to note here that it would be difficult for a repository to accumulate a number of "stars"[29] big enough to be included in this study, if it has been created later in time. This means that repositories created in 2015 are unlikely to have accumulated enough "stars" to actually be included in this study. A repository created in 2010 on the other hand will have had more time to accumulate a larger number of stars.
Another reason for the low amount of created repositories in 2015 could be that, when following the trend in Figure 12, the popularity and innovative ideas regarding Node.js has reached a point where the developers are satisfied with what Node.js can offer.

The total number of created repositories were as follows:

Year	No. repositories created
2010	91
2011	119
2012	85
2013	80
2014	23
2015	1

Table 6: No. of repositories created per year

5.9.2 Number of contributors

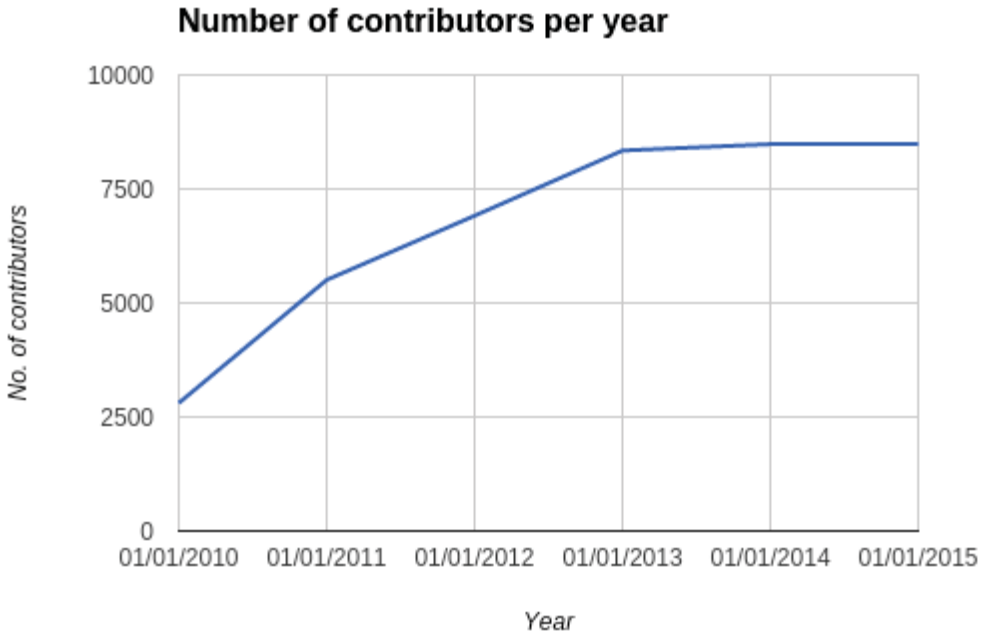

Figure 13: Number of contributors per year

Figure 12 depicts the total number of contributors to all repositories in the Node.js community on Github. The y-axis shows the number of contributors, the x-axis shows the year.
Important to note here is that the number of contributors is accumulated from year to year and this graph needs to be looked upon with that in mind. With that said, we can observe that there is a clear

stagnation in the number of contributors. From having a steady increase each year up till the year 2013, the number of contributors have thereafter virtually stopped increasing.

5.10 Analysis

Looking at the two graphs in Figure 12 and Figure 13 related to trends revolving Node.js it becomes quite apparent that there has been a significant stagnation in how many projects that are created each year and also in the contribution to these projects. While it might be a natural evolution of a project to stagnate as it matures and the new ways in which a project can actually be used becomes more limited, it is interesting to look at Node.js in a broader sense and speculate whether the stagnation of the Node.js community is connected with the dissatisfaction of said community on how Joyent [42], the main sponsor and steward of Node.js, has handled the maintenance of the technology. In this broader sense, it is also interesting to discuss whether the technology suffered at all from its creator, Ryan Dahl, stepping aside in January 2012[43] and if this had any implications for the Node.js community.

Dahl's departure coincides quite well with the stagnation of contribution but we have found no articles or posts to back any claim that the loss of Dahl as head of the project would have any drastic effects on the project. It is, however reasonable to think that his departure has led to events that would cause concern and unhappiness within the community.

The dissatisfaction shown from the Node.js community has been well documented and is nicely summarized in a blog post entitled "*Node.js vs io.js: Why the fork?!?*" by Anand Mani Sankar[44]. In this post, Sankar gives an explanation for the recent fork[45] of Node.js, creating io.js[46], and gives the reader a glimpse into the way of thinking of both the community and of the Node.js core team members.
Talk of a possible forking started mid 2014, after some developers became discontented with the role of Joyent as the steward of the Node.js project. Some of the core contributors tried working with Joyent to move the project into a structure where the contributors and community could step in to solve the problems facing Node.js. Some independent efforts were started by contributors to improve Node and its ecosystem, in the name of 'Node Forward'[86].
But Fedor Indutny, one of the core team members, got tired of getting nowhere with the discussion and started the fork that is currently called io.js. Other contributors close to Fedor volunteered and jumped in[44].

An active and supportive community is vital to an open source project. Without a community, there is no project and a community needs careful management and does not thrive automatically[62].
Another factor for a successful open source project is that it takes more effort from the developers to maintain a growing project. When the amount of contributors and developers increases, the project needs more attendants to bug fixing than actual development[63].
It becomes quite clear that the discontent of the Node.js community has been growing over a long period of time and it could be argued that the actions of the community are well justified, given the development of Node.js since its initial uprising. As mentioned in the blog post *"Node.js vs io.js: Why the fork?!?"*[44], one of the reasons for that discontent already mentioned was that Fedor Indutny, one of the core team members, got tired of getting nowhere in the discussion with Joyent. Another reason mentioned was that the community wanted contributors to have more control and to seek consensus.

Looking at the trends of commits to the official Node.js repository it becomes visibly clear that the trends of the core project coincides with the overall contribution of the community;

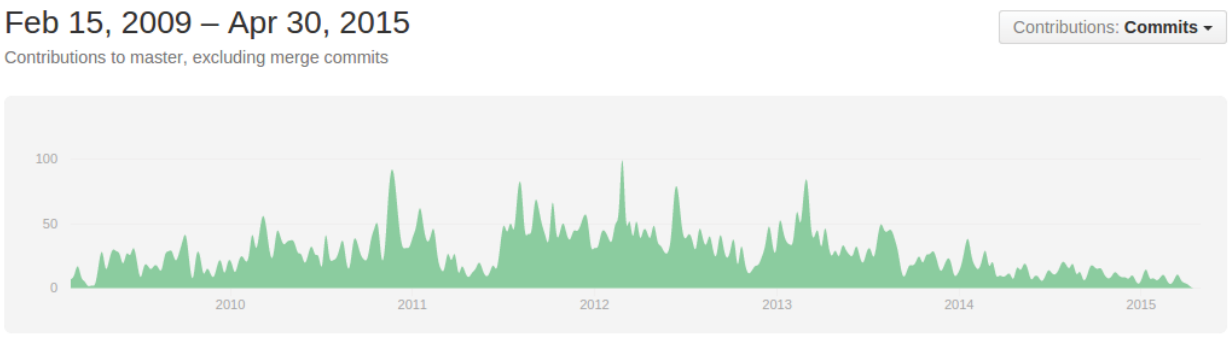

Figure 14: Commit history of the core Node.js repository on GitHub[47]

Through our research we can observe that the decline in the creation of new projects using Node.js started even earlier than 2014; the creation of new projects actually started declining quite dramatically between 2013 and 2014 and as of the time of writing this trend is unchanged, though it should be noted that our research focus is on popular repositories and not overall contribution.
Is this a general trend among popular repositories [65] or can this be attributed to the dissatisfaction specifically tied to the Node.js community? We can quite easily compare the number of commits to a few other popular repositories on Github, such as Bootstrap[48], Angularjs[49] and Ruby on Rails[50] and see if we can draw any conclusions;

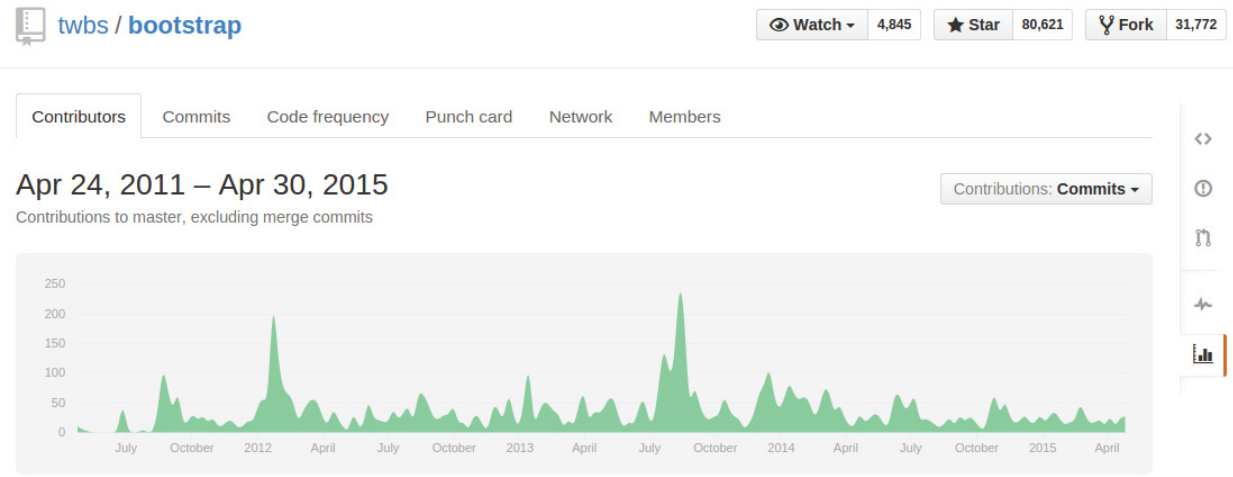

Figure 15: Commit history of the core Bootstrap repository on GitHub[51]

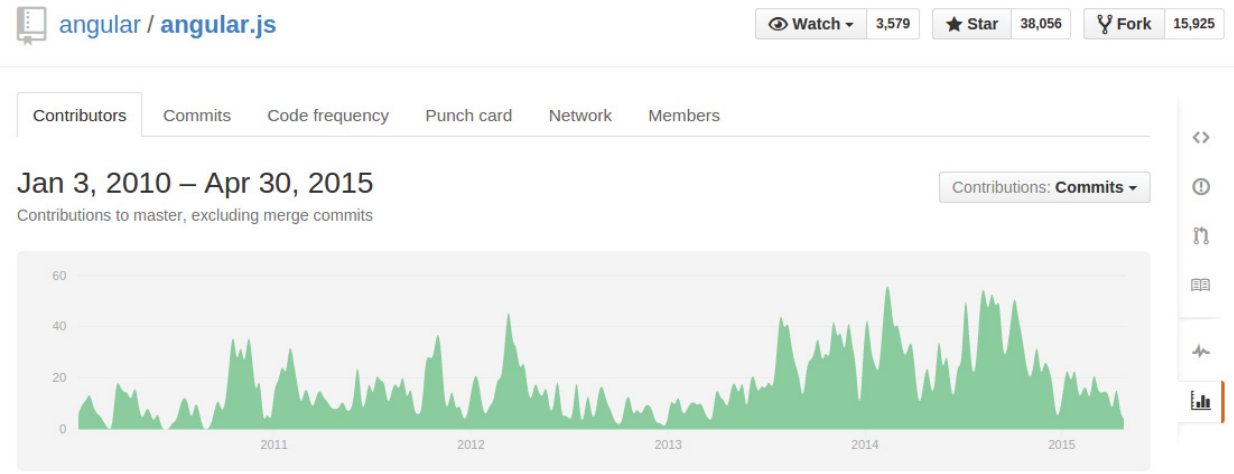

Figure 16: Commit history of the core Angular.js repository on GitHub[52]

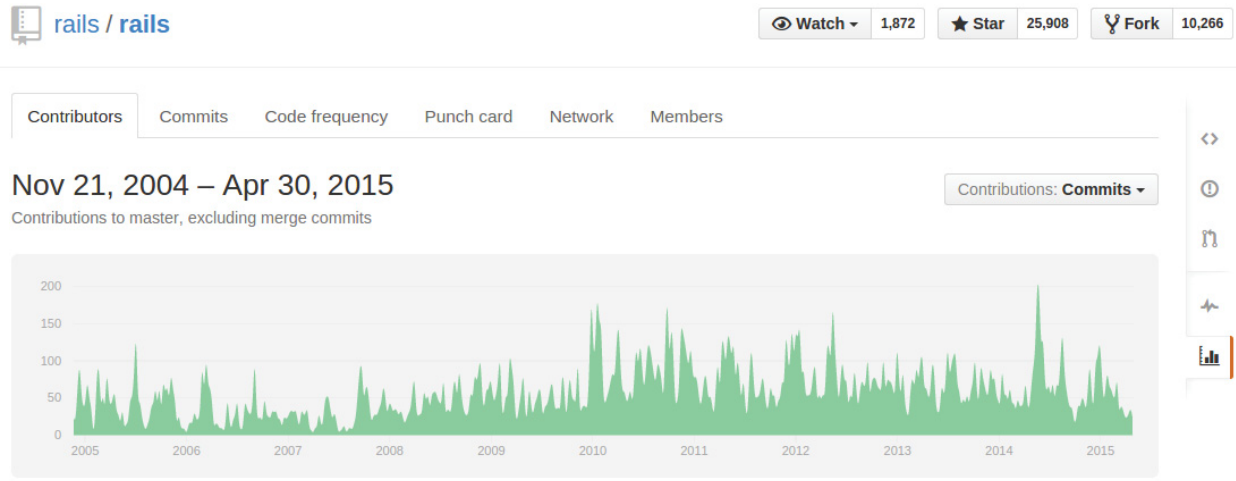

Figure 17: Commit history of the core Rails repository on Github[53]

The repositories displayed in Figures 15, Figure 16 and Figure 17 have had quite steady commit histories. There have been ups and downs but here we don't see the same level of decline as we see with Node.js.

This does not provide the full picture, however. We can only speculate in what the way the communities and ecosystems of these projects have evolved over time but what we can say with certainty is that we do not see the same decline in commits to these projects. One would be inclined to think that this correlates with the overall contribution and collaboration of the communities. It would probably not be rash to draw the conclusion that a thriving community ecosystem provides a stronger foundation for the evolution of the project. Node.js has, however, continuously been successful in gaining traction in the tech community and has been adopted by companies and organisations in a large spectrum of sectors, which becomes apparent in the survey "*What is Node.js used for: The 2015 Node.js Overview Report*"[54] by Gabor Nagy, which leads one to wonder whether a thriving community is really necessary for an open source project to evolve or whether the main reason for the success of open source projects is the backing by corporations with strong financial power, which is the case with both Bootstrap (backed by Twitter[55]), Angularjs (backed by Google[56]) and Node.js. Software vendors face numerous challenges when developing software

within a software ecosystem [66] and many software producing organizations do not know how to measure, compare, and analyse their governance policy in software ecosystems [67] and in the case of Node.js and Joyent it would seem that these challenges have developed into real problems.
It could definitely be argued that, when comparing Node.js with these other major open source projects, there has not been nearly as much public controversy surrounding these projects as there has been with Node.js. In spite of this, Node.js as a technology seems to be thriving still. Is this because it provides a very specific use case? Our research says otherwise; we can clearly see the large number of ways that Node.js can be and is being used by the open source community. It would have made more sense for this to drive innovation even further and create even more projects over time. As becomes visible through our research, this is not the case.

Is the stewardship of Node.js by Joyent holding the community back? The community seems to think so, which becomes clear with the major fork leading to io.js. Has Node.js continued to be successful in spite of this community backlash? The number of major organisations and companies using Node.js as an important part of their tech stack would serve as an argument to back such a claim. It will be interesting to see what the future holds for Node.js; can it regain the contribution from its community or will the community go with io.js? Joyent has gone to some lengths to try and heal its relationship with the community and come to a reconciliation [57] with previous core team developers now working on io.js. It is still to early to say if these measures will have any effect and ultimately unify the two repositories under one name. You'd be bound to think that this would be the goal of Joyent; to have Node.js evolve entirely under their governance once more.
In the latter case where the community sticks with io.js; would Node.js be able to maintain its strong position in the tech world without the backing of a strong community or will it be overtaken by other, similar technologies that actually have true community backing? Hopefully this thesis serves as motivation to further research into these questions.

Looking at the results from the OSS (Open Source Software) perspective, it becomes interesting to understand Node.js in the light of OSS research. We have seen how large companies such as Facebook, Google and Twitter have moved into OSS and/or created open source projects of their own[58] but to a great extent these companies are also using and contributing to already existing open source projects. While you would understand that a company would like to maintain control over a project that they have initiated themselves, it is interesting to speculate how strong this control can be before the community objects.

The level of control is going to be related to if (1) the technology becomes a commodity and (2) what the core business model of these companies actually is and in what way they make the most of their money. All of the above mentioned companies (Facebook, Google and Twitter) have all released open source projects, however in most cases these companies do maintain a certain level of control; it's one thing to simply release something freely on Github and watch it evolve but a bit different to give very specific instructions on how to contribute to the project, which is the case with the Cassandra database[59], for instance. This could be said to be a form of control but it could also be argued that this a way to maintain a high level of quality; the argument can be turned both ways. Companies do face the risk of losing that control when they release their products into the open source community, however. This is, for instance, demonstrated by Microsoft's investment and thereby involvement in Cyanogen, whose business model is based on building their own version of the Android operating system [69], which of course is a Google product that has been released as open source. As a sidenote,

Microsoft is now also a stakeholder in the Node.js community with its inclusion as one party of the newly founded Node.js Foundation [70].

What is important to note, however, is that companies such as Facebook and Google use software as one way of gaining a commercialized, competitive edge [68]. For these companies, operating the software on a large scale is what makes them money and if there's a community that can improve the software, this will only benefit these companies. In the case of Node.js and Joyent, there is a greater need for the backing company to maintain a stronger control of the project since the technology itself is what gives Joyent the competitive edge. If someone were to build upon Node.js and create something that works better and release it for themselves, it would be a great loss for Joyent. Whether this is an explanation for why Joyent has handled the maintenance and thereby the evolution of Node.js the way that they have, or if there are other circumstances that come into play is hard to say. It is impossible to look at every major event since the initiation of Node.js, leading to the overall resentment from the Node.js community, which in turn led to the fork of the repository and the parallel project that is io.js.

The tension between Joyent and the Node.js community has been clear for everyone to see though, and it will be interesting to see if any of the measures that Joyent has taken, such as forming the Node.js Advisory Board [60] and the Node.js Foundation [70], will have any effect and help bring the community back in line with the ambitions of Joyent.

6.0 Conclusion

This study has looked into the different ways of how Node.js is used in open source projects on Github. We have analysed the way in which Node.js is actually being used and observed current trends of its community ecosystem.
Research into Javascript in general is important because it has become one of the most used programming languages on the Web and therefore research into Javascript and its libraries provides an insight into how the web evolves and the technologies that are being built. Being a Javascript technology, researching the usages of Node.js is important since it allows us to analyse an important way in which Javascript is being moved forward.
The results show that web applications is by far the most well represented category with over 50% of all usages falling into this category. Network applications and Web servers come in at second and third position with 14% and 13% respectively. There are however a multitude of other ways in which Node.js is being used, which the results demonstrate.

6.1 Answer to the research question

Our research question was: "What are the most common usages of Node.js in open source projects on GitHub?"
The most common usages of Node.js in open source projects on GitHub are Web applications, Network applications, Web servers, Test frameworks and tools, security focused applications and command line tools, in that order.
Web applications is by far the most well represented category with over 50% of all usages falling into this category. Network applications and Web servers come in at second and third position with 14% and 13% respectively.

6.2 Discussion

This case study has been aimed at displaying the different ways of how Node.js is used and provide an insight into the diversity of the technology. The study does indeed show that there are a multitude of ways in which Node.js can be and is being used by the open source community.
To answer the stated research question associated with this study, the most common usages for Node.js in open source projects on Github are web applications, web servers and network applications. These categories are followed by test frameworks and tools, security applications and command line interface applications and thereby expands the list of potential usage categories discovered during the literature review.
An unintended discovery made during the course of this study was the stagnation in the creation of new projects involving Node.js and the decline in community participation from the developer ecosystem. The reasons for this are discussed in the analysis of this study and hopefully this paper can stimulate further studies into the community of Node.js.

6.3 Contribution

In this study we have focused on Node.js and looked at the technology from a community standpoint. We have investigated in what ways the surrounding ecosystem of developers actually use Node.js. In this sense, this study will hopefully provide new insights and a different perspective than previously performed research. Developers of all levels can find related projects and scientists as well as the Node.js community can observe the trends and diagrams for further investigation.

6.4 Future work

We hope that this study is going to inspire future studies into the community of Node.js and in what ways Node.js is currently used and in what ways the technology might be used in the future. We also hope that future research studies will focus on the evolution of Node.js considering recent events and what effects these events have had on the community.

7.0 References

1. (2014) The W3Tech homepage. [Online]. Accessed: 2015-04-07 Available: http://w3techs.com/
2. (2014) W3Techs Usage of web servers. [Online]. Accessed: 2015-04-07 Available: http://w3techs.com/technologies/overview/web_server/all
3. (2014) W3Techs Usage of JavaScript libraries. [Online]. Accessed: 2015-04-07 Available: http://w3techs.com/technologies/overview/javascript_library/all
4. T. Capan. (2012) Why The Hell Would I Use Node.js? [Online]. Accessed: 2015-04-07 Available: http://www.toptal.com/nodejs/why-the-hell-would-i-use-node-js
5. (2014) Node.js on Wikipedia.org. [Online]. Accessed: 2015-04-08 Available: https://en.wikipedia.org/wiki/Node.js
6. (2014) Node.js homepage. [Online]. Accessed: 2015-04-08 Available: http://nodejs.org/
7. (2014) Node.js on W3Tech. [Online]. Accessed: 2015-04-09 Available: http://w3techs.com/technologies/details/ws-nodejs/all/all
8. (2014) Node.js results on StackOverflow. [Online]. Accessed: 2015-04-09 Available: http://stackoverflow.com/search?q=node+js
9. (2014) How to get started with node.js on StackOverflow. [Online]. Accessed: 2015-04-09 Available: http://stackoverflow.com/questions/2353818/how-do-i-get-started-with-node-js/5511507#5511507
10. (2014) Node.js industry on Node.js homepage. [Online]. Accessed: 2015-04-09 Available: http://nodejs.org/industry/
11. J. O'Dell (2012) Why Walmart is using Node.js on venturebeat.com. [Online]. Accessed: 2015-04-09 Available: http://venturebeat.com/2012/01/24/why-walmart-is-using-node-js/
12. N. C. Hong. (2013) Choosing a repository for your software project on software.ac.uk. [online]. Accessed: 2015-04-010 Available: http://software.ac.uk/resources/guides/choosing-repository-your-software-project
13. (2014) Software hosting popularity on wikipedia.org. [Online]. Accessed: 2015-04-09 Available: http://en.wikipedia.org/wiki/Comparison_of_open-source_software_hosting_facilities#Popularity
14. R.A. Ghosh. (2007) Economic impact of open source software on innovation and the competitiveness of the Information and Communication Technologies (ICT) sector in the EU. [online]. Accessed: 2015-04-04 Available: http://ictlogy.net/bibliography/reports/projects.php?idp=895
15. E. Knorr. (2014) The open source way of being. [Online]. Accessed: 2015-04-04 Available: http://www.infoworld.com/article/2607921/application-development/the-open-source-way-of-being.html
16. (2015) Node.js homepage. [Online]. Accessed: 2015-04-05 Available: https://nodejs.org/community/
17. (2015) Node.js GitHub repository. [Online]. Accessed: 2015-04-05 Available: https://github.com/joyent/node

18. (2014) C. Wohlin, Guidelines for Snowballing in Systematic Literature Studies and a Replication in Software Engineering, in: Proceedings of the 18th International Conference on Evaluation and Assessment in Software Engineering (EASE'14), ACM, London, UK.
19. (2014) P. Runeson, M. Host, A. Rainer, B. Regnell, Case Study Research in Software Engineering: Guidelines and Examples (John Wiley & Sons, 2012b). ISBN 9781118181003
20. (2006) R. Wieringa, N. Maiden, N. Mead, C. Rolland, Requirements engineering paper classification and evaluation criteria: a proposal and a discussion. Requirements Engineering 11(1), 102-107
21. Wikipedia homepage [Online]. Accessed: 2015-04-10 Available: http://en.wikipedia.org/wiki/Throughput
22. Wikipedia homepage [Online]. Accessed: 2015-04-10 Available: http://en.wikipedia.org/wiki/Load_testing
23. Wikipedia homepage [Online]. Accessed: 2015-04-11 Available: http://en.wikipedia.org/wiki/Stress_testing_(software)
24. Wikipedia homepage [Online]. Accessed: 2015-04-11 Available: http://en.wikipedia.org/wiki/Soak_testing
25. Wikipedia homepage [Online]. Accessed: 2015-04-12 Available: http://en.wikipedia.org/wiki/Benchmark_%28computing%29
26. Wikipedia homepage [Online]. Accessed: 2015-04-10 Available: http://en.wikipedia.org/wiki/Scenario_testing
27. (2012) A. Ojamaa and K. Duuna, "Security assessment of Node.js platform," Proc. of the International Conference on Information Systems Security.
28. Stackexchange [Online] Accessed on 2015-04-10 Available: http://webapps.stackexchange.com/questions/29756/what-does-starring-a-repository-on-github-do
29. (2012) kneath, "Notifications & Stars" Github homepage[Online]. Accessed on 2015-04-10 Available: https://github.com/blog/1204-notifications-stars
30. Wikipedia homepage [Online]. Accessed: 2015-04-10 Available: http://en.wikipedia.org/wiki/Uniform_resource_locator
31. Github API docs.[Online]. Accessed on 2015-04-10 Available: https://developer.github.com/v3/search/#search-repositories
32. Wikipedia homepage [Online]. Accessed: 2015-04-13 Available: http://en.wikipedia.org/wiki/README
33. Wikipedia homepage [Online]. Accessed: 2015-04-14 Available: http://en.wikipedia.org/wiki/Npm_%28software%29
34. "Fork A Repo", Github homepage [Online]. Accessed: 2015-04-10 Available: https://help.github.com/articles/fork-a-repo/
35. "About npm", npmjs homepage [Online]. Accessed 2015-04-30 Available: https://www.npmjs.com/about
36. Wikipedia homepage [Online]. Accessed: 2015-04-15 Available: http://en.wikipedia.org/wiki/Node.js
37. Wikipedia homepage [Online]. Accessed 2015-04-30 Available: http://en.wikipedia.org/wiki/Web_application
38. "NodeOS", Github homepage [Online]. Accessed 2015-04-30 Available: https://github.com/NodeOS/NodeOS

39. "Command line vs. GUI", Computerhope homepage [Online]. Accessed 2015-04-30 Available: http://www.computerhope.com/issues/ch000619.htm
40. Node.js homepage [Online]. Accessed 2015-04-30 Available: https://nodejs.org/
41. (2007) Markus Clermont, Google testing blog homepage [Online]. Accessed 2015-05-08 Available: http://googletesting.blogspot.se/2007/10/automating-tests-vs-test-automation.html
42. Joyent homepage [Online]. Accessed: 2015-04-30 Available: https://www.joyent.com/
43. Wikipedia Homepage, [Online], Accessed 2015-04-30 Available: http://en.wikipedia.org/wiki/Node.js
44. (2015) Anand Mani Sankar, "Node.js vs io.js: Why the fork?!?", [Online], available: http://anandmanisankar.com/posts/nodejs-iojs-why-the-fork/, Accessed 2015-04-30
45. (2014) James Chesters, "Io.js, Node.js Fork, Plans First Release for January 2015", [Online], availble: http://www.infoq.com/news/2014/12/iojs, Accessed 2015-04-30
46. Github, [Online], available: https://github.com/iojs/io.js, Accessed 2015-04-30
47. Github, [Online], available: https://github.com/joyent/node/graphs/contributors, Accessed 2015-04-30
48. Github, [Online], available: https://github.com/twbs/bootstrap, Accessed 2015-04-30
49. Github, [Online], available: https://github.com/angular/angular.js, Accessed 2015-04-30
50. Github, [Online], available: https://github.com/rails/rails, Accessed 2015-04-30
51. Github, [Online], available: https://github.com/twbs/bootstrap/graphs/contributors, Accessed 2015-04-30
52. Github, [Online], available: https://github.com/angular/angular.js/graphs/contributors, Accessed 2015-04-30
53. Github, [Online]. available: https://github.com/rails/rails/graphs/contributors, Accessed 2015-04-30
54. (2015) Gabor Nagy, "What is Node.js used for: The 2015 Node.js Overview Report", [Online]. Accessed 2015-04-30 Available: http://blog.risingstack.com/what-is-nodejs-used-for-the-2015-nodejs-overview-report/,
55. Bootstrap Homepage, [Online]. Accessed 2015-04-30 Available: http://getbootstrap.com/
56. AngularJs Homepage, [Online]. Accessed 2015-04-30 Available: https://angularjs.org/
57. (2015) Paul Krill, "Reconciliation between Node.js and io.js could be near", [Online]. Accessed 2015-04-30 Available: http://www.infoworld.com/article/2889835/application-development/reconciliation-between-node-js-and-io-js-could-be-on-the-horizon.html
58. (2013) Matt Asay, "Technology Innovation Management Review", [Online]. Accessed 2015-05-01 Available: http://timreview.ca/article/650
59. (2015) Cassandra wiki, "HowToContribute", [Online]. Accessed 2015-05-01 Available: http://wiki.apache.org/cassandra/HowToContribute
60. (2014) Scott Hammond, "Node.js Advisory Board", [Online]. Accessed 2015-05-01 Available: https://www.joyent.com/blog/node-js-advisory-board
61. Github Homepage, [Online]. Accessed 2015-04-10 Available: https://github.com/
62. (2013) Matthew Mascord, "How To Build An Open Source Community", [Online]. Accessed 2015-05-11 Available: http://oss-watch.ac.uk/resources/howtobuildcommunity
63. Midha, V. and Palvia, P. "Factors Affecting the Success of Open Source Software". Journal of Systems and Software. Volume 85, Issue 4, April 2012, pp. 895–905. [Online]. Accessed 2015-05-11 Available: http://libres.uncg.edu/ir/uncg/f/P_Palvia_Factors_.pdf

64. (2012) Wohlin, C., Runeson, P., Höst, M., Ohlsson, M. C., Regnell, B., & Wesslén, A. *Experimentation in software engineering*. Springer Science & Business Media.
65. (2013) Krill, Paul. "GitHub's top 10 most popular projects", Computerworld, [Online]. Accessed 2015-05-14 Available: http://www.computerworld.com/article/2473298/application-development/120001-GitHub-s-top-10-rock-star-projects.html#slide1
66. (2009, May). Jansen, S., Finkelstein, A., & Brinkkemper, S. A sense of community: A research agenda for software ecosystems. In Software Engineering-Companion Volume, 2009. ICSE-Companion 2009. 31st International Conference on (pp. 187-190). IEEE.
67. (2012) Baars, A., & Jansen, S. A framework for software ecosystem governance. In Software Business (pp. 168-180). Springer Berlin Heidelberg.
68. (2007, Dec. 4) Om Malik, "Google's Infrastructure is its Strategic Advantage", Gigaom, [Online] Accessed 2015-05-14 Available: https://gigaom.com/2007/12/04/google-infrastructure/
69. (2015) Alba Davey, "Microsoft to Invest in Android Startup Cyanogen, Says Report", Wired, [Online] Accessed 2015-05-14 Available: http://www.wired.com/2015/01/microsoft-invest-android-startup-cyanogen-says-report/
70. (2015, Feb.) "PRESS RELEASE Joyent Moves to Establish Node.js Foundation", Joyent, Marketwatch, [Online] Accessed 2015-05-14 Available: http://www.marketwatch.com/story/joyent-moves-to-establish-nodejs-foundation-2015-02-10
71. (2013, April) Greg Ferro, "Thoughts on Choosing node.js for Network Automation" Ethereal Mind, [Online] Accessed 2015-05-14 Available: http://etherealmind.com/thoughts-on-choosing-node-js-for-automation/#wrap
72. Engineering Village, [Online], Available: https://www.engineeringvillage.com/
73. IEEE xplore, [Online], Available: http://ieeexplore.ieee.org/
74. Scopus, [Online], Available: http://www.scopus.com/
75. Web of science, [Online], Available: http://apps.webofknowledge.com.miman.bib.bth.se/
76. Lei, K., Ma, Y., & Tan, Z. (2014, December). Performance Comparison and Evaluation of Web Development Technologies in PHP, Python, and Node. js. In Computational Science and Engineering (CSE), 2014 IEEE 17th International Conference on (pp. 661-668). IEEE.
77. Ojamaa, A., & Duuna, K. (2012, December). Assessing the security of Node. js platform. In Internet Technology And Secured Transactions, 2012 International Conference for (pp. 348-355). IEEE.
78. Zhao, S. M., Xia, X. L., & Le, J. J. (2013). A Real-Time Web Application Solution Based on Node. js and WebSocket. Advanced Materials Research, 816, 1111-1115.
79. Rice, J. L., Phoha, V. V., Cappelaere, P., & Mandl, D. (2011, November). Web Farm-inspired Computational Cluster in the Cloud. In Cloud Computing Technology and Science (CloudCom), 2011 IEEE Third International Conference on (pp. 730-737). IEEE.
80. Griffin, L., Ryan, K., de Leastar, E., & Botvich, D. (2011, June). Scaling Instant Messaging communication services: A comparison of blocking and non-blocking techniques. In Computers and Communications (ISCC), 2011 IEEE Symposium on (pp. 550-557). IEEE.
81. Karagoez, M. F., & Turgut, C. (2014, May). Design and Implementation of RESTful Wireless Sensor Network Gateways Using Node. js Framework. In European Wireless 2014; 20th European Wireless Conference; Proceedings of (pp. 1-6). VDE.
82. Chaniotis, I. K., Kyriakou, K. I. D., & Tselikas, N. D. (2014). Is Node. js a viable option for building modern web applications? A performance evaluation study. Computing, 1-22.

83. De Groef, W., Massacci, F., & Piessens, F. (2014). A security architecture for server-side JavaScript. status: published.
84. Frizelle, J. (2011). A scalability study into Server Push technologies with regard to server performance (Doctoral dissertation, Waterford Institute of Technology).
85. Fysarakis, K., Mylonakis, D., Manifavas, C., & Papaefstathiou, I. (2015). Node. DPWS: High performance and scalable Web Services for the IoT. arXiv preprint arXiv:1503.01398.
86. NodeForward homepage [Online]. Accessed: 2015-05-22 Available: http://nodeforward.org/
87. Wikipedia homepage [Online]. Accessed 2015-04-30 Available: https://en.wikipedia.org/wiki/GitHub
88. Wikipedia homepage [Online]. Accessed 2015-03-06 Available: http://en.wikipedia.org/wiki/Web_application
89. Wikipedia homepage [Online]. Accessed 2015-03-06 Available: https://en.wikipedia.org/wiki/Web_server
90. Wikipedia homepage [Online]. Accessed 2015-03-06 Available: http://scitechnol.com/computer-engineering/articles-on-computer-network.php
91. Wikipedia homepage [Online]. Accessed 2015-03-06 Available: https://en.wikipedia.org/wiki/Authorization_(computer_access_control)
92. Wikipedia homepage [Online]. Accessed 2015-03-06 Available: https://en.wikipedia.org/wiki/Authentication
93. Wikipedia homepage [Online]. Accessed 2015-03-06 Available: https://en.wikipedia.org/wiki/Encryption
94. "Genres in academic writing: Case studies" UEAFAP homepage [Online]. Accessed 2015-03-08 Available: http://www.uefap.com/writing/genre/casestud.htm

Appendix A

index.html

```
<!DOCTYPE HTML>
<html>
	<head>
		<title></title>
		<meta http-equiv="content-type" content="text/html; charset=utf-8" />
		<meta name="description" content="" />
		<meta name="keywords" content="" />
		<link rel="stylesheet" href="css/style.css" />
	</head>
	<body>
		<table id="result">
			<tr><td>Nr</td><td>Name</td><td>Created</td></tr>
		</div>
		<script src="js/jquery-2.1.3.min.js"></script>
		<script src="js/main.js"></script>
	</body>
</html>
```

js/main.js

```
(function(){
        'use strict';
        var counter = 0;
        var pageCount = 2;

        $.ajax({
                url:
"https://api.github.com/search/repositories?q=node.js+created:>2010-01-01&sort=stars&order=
desc&per_page=100&page=1",
                dataType: "json",
                success: function (returndata)
                {
                    console.log(returndata.items[0]);
                    for(var i = 0; i < returndata.items.length; i++) {
                        counter++;
                        $("#result").append("<tr><td>" + counter + "</td><td> <a href=" +
returndata.items[i].svn_url + " target='blank'>" + returndata.items[i].name +
"</a></td><td>" + returndata.items[i].created_at + "</td></tr>");
                    }
                    next(pageCount);
                }
        });

        var next = function(page) {
            $.ajax({
                    url:
"https://api.github.com/search/repositories?q=node.js+created:>2010-01-01&sort=stars&order=
desc&per_page=100&page="+page+"",
                    dataType: "json",
                    success: function (returndata)
                    {
                        for(var i = 0; i < returndata.items.length; i++) {
                            counter++;
                            $("#result").append("<tr><td>" + counter + "</td><td> <a href="
+ returndata.items[i].svn_url + " target='blank'>" + returndata.items[i].name +
"</a></td><td>" + returndata.items[i].created_at + "</td></tr>");
                        }
                        pageCount++;
                        if(pageCount < 11) {
                            next(pageCount);
                        }
                    }
            });
    }
 })();
```