UpStage Team

Investigation of Programming Languages

UpStage 2015 S2

Table of Contents

[Researching Criteria 2](#_Toc431313296)

[Introduction 3](#_Toc431313297)

[C# 4](#_Toc431313298)

[Java 6](#_Toc431313299)

[Node.js 9](#_Toc431313300)

[Python 12](#_Toc431313301)

[Proof of Concept 15](#_Toc431313302)

[Rationale for choosing C# 19](#_Toc431313307)

[References 22](#_Toc431313308)

|  |  |  |  |
| --- | --- | --- | --- |
| Version | Date | Author | Changes |
| 1 | September 21, 2015 | Alyssa Byun, Jing Han, Joshua Kartono, Siatua Uili, and William Stokes |  |
| 2 | September 30, 2015 | Alyssa Byun, Jing Han, Joshua Kartono, and Siatua Uili | * Introduction added * References added * Research evidence on Atmosphere and SingalR added |

# Researching Criteria

For the programming languages:

1. Readability / Expressiveness  
   - How easy to understand to code
2. Testability  
   - Are there any useful testing and debugging tools for this language?
3. Learnability  
   - How much knowledge does the team have of this language?  
   - Is it easy to learn?  
   - How much prior knowledge does it require to use this language?
4. Performance  
   - Can it run in real time (asynchronous communication)?  
   - Does it support the chosen architecture?  
   - How fast?

# Introduction

During the spike before the project team starts developing the new UpStage, we should decide which programming language to use for developing it. The existing UpStage has been developed with Python. The clients have said that the team does not need to stick with Python for the new UpStage. They also told us that they will be satisfied as long as the team completes this project and delivers a high quality product.

For the research of programming languages, we have identified several criteria so that we are able to measure the difference between the chosen languages for researching. The chosen languages to research are listed below.

* **Python**: The current UpStage is written in Python. Some code can be possibly reused. The team should learn this language to maintain the current UpStage.
* **Java**: All team members are familiar with Java as BCIS students in AUT are taught programming papers in Java, and so the new members joining each semester in the future are.
* **C#**: As familiar programming languages like Java can save some time for learning and getting used to it, we considered other familiar programming languages. One of the team members is used to using C# through her work experience and thought that C# and ASP.NET framework is a good to build a web-based system.
* **Node.js**: One of the past team members researched and suggested to use this language for the new UpStage as Node.js supports a lot of functions that UpStage requires.

All members might want to use the programming language they are most familiar with, but what we need to consider is whether it is the best programming language that is suitable to support the required functions for this product with an acceptable quality.

# C#

C# is defined as a simple, modern, general-purpose, type-safe, object-oriented and high-level programming language developed by Microsoft running on top of the .Net Framework (Benton, Cardelli, & Fournet, 2004).

**Learnability**

Pros

* Like Java, C# provides a number of features to make it easier for a developer to code in this language such as type checking, bounds checking, uninitialized variable checking and garbage collection.
* Language is simplified for some functions such as multithreading.
* Easy to learn if you are familiar with other programming language like Java or C++.
* As UpStage requires real time web functionality, SignalR can be used. SignalR is a library for ASP.NET developers that makes it incredibly simple to add real-time functionality to the application (“SignalR”, n.d.).

Cons

* There was another disadvantage of C#, which it was not supportable for the platforms other than Windows, but .NET Core became open source in 2014. Roslyn, ASP.NET 5, Entity Framework, etc. these are now all free to use (“.NET Blog”, 2014). However, if the platform is not Windows, the knowledge of using .NET framework on other platform such as Linux is required as C# is a language more native to Windows.

**Readability**

Cons

* C# uses the curly brackets like Java does. It is more readable than other languages such as Python that do not use the curly brackets.

Pros

* LINQ in C# makes easier to code but harder to read.

**Testability**

Pros

* .NET provides many testing tools for many purpose. i.e. Telerik Test Studio for a UI testing, TypeMock for unit testing, etc.
* ASP.NET has a useful tools like Elmah (Error Logging Modules and Handlers) or Glimpse (the open source diagnostics platform of the web to inspect web requests to reduce debugging time).

**Performance**

Pros

* Have all the benefits of .NET framework as C# is tied to .NET framework.

# Java

Java is a popular and widely supported programming language. It follows the WORA (Write Once, Run Anywhere) philosophy, which means that once compiled, java code will be able to run on any supported platforms without recompiling. For this project, in terms of options for programming the Back-End of UpStage, Java has the following pros and cons.

**Readability**:

Pros:

\* As Java, like C#, is a structured language with things such as the {} syntax, it is easier to read compared to scripting languages such as Python or Node.js as the code follows a more defined structure. For this project specifically, where AUT students will be working on it, the core language(s) taught at AUT are structured languages which means that they are suitable for current and future team members.

Cons:

\* Java is a relatively verbose language compared to some scripting languages, such as Python or Node.js, or languages such as C#, which means code that does the same thing is often longer in length in Java than it is in the other languages. However, team members’ familiarity with Java negates this point.

**Testability**:

Pros:

\* Forced/checked exceptions mean that they will not be left as an afterthought, reducing maintenance costs in the future and making it easier to debug the program. This is important as new team members come and some team members leave every semester, which means less people with knowledge of the current code and more people who need to understand the code. Checked exceptions will help reduce the amount of bugs, which new team members will not know about if these are not done.

\* Backwards compatibility is an integral part of Java, which will reduce maintenance costs in the future as functions will be less likely to break or stop working between versions or updates. The JVM, or Java Virtual Machine, is fully backwards compatible and can run any code compiled with previous versions of the JDK. This means that any recent browsers can be used to open UpStage without having to program specifically for them.

\* Java has access to excellent and powerful testing tools such as Junit. This will aid programmers in debugging, or otherwise testing, the code. Automated testing tools such as Selenium is also compatible with Java. This is useful for UpStage as being able to automate testing and test creation will save time, and the UpStage project is time intensive.

Cons:

\* Although programmers are forced to catch exceptions, they are not forced to handle the exceptions, and may instead mask them by an empty exception handling, which will make the code harder to debug or maintain. This can make future members confused because an exception is thrown but the source or cause is not stated. This can be avoided by forcing team members to handle all caught exceptions.

**Learnability**:

Pros:

\* An extensive library of modules and APIs lessens the chance of having to “Reinvent the Wheel”.

\* Familiarity, as Java is the main language taught to AUT students, and as such upskilling time for new members can be reduced.

Cons:

\* A vast library means it can be difficult to find the proper libraries and/or tools, which costs time. An inferior option may be used in place of a better option if not enough research is done in choosing which libraries to use, as some libraries are known to be of worse quality than their equivalents.

**Performance**:

Pros:

\* There are several libraries, such as Atmosphere and Spring Framework, which aids java in achieving asynchronous communication, which is a requirement for the project.

\* As Java depends on its JVM to run the code, its performance is relatively stable on all platforms, whereas C# might run slower on non-windows platforms, and Python and Node.js are dynamically typed, which will slow them down on run-time. This is especially true when the amount of data is not large.

Cons:

\* Java applications may have some performance spikes, such as when a class is loaded when it is first referenced, or when the garbage collection happens, which might result in non-asynchronous communication for a short period of time. As such, java’s real-time performance is not consistent. This will not come into play when the amount of data is not overly huge, as is the case with UpStage.

**Extra**:

Pros:

\* Widely supported on most platforms, including computers and mobile devices, like Python and Node.js, but unlike C# which is primarily made to work on windows platforms.

Cons:

\* Java is relatively heavy-weight compared to scripting languages like Python and Node.js.

# Node.js

Server side technology - (Replacing Twisted)

Pros:

+It is open source

+The best architecture candidate chosen for UpStage at this point is Event Driven Architecture and Node.js uses an event-driven model making it a great language for real-time applications (“Node.js”, n.d.).   
In terms of architecture patterns, node also has frameworks for MVC (“Node.js”, n.d.).

+ Node.js comes with npm (built-in package manager) which is where it holds all packages of open source libraries. So unlike other languages there is no need to find suitable APIs for UpStage and Node only needs a constant update of npm. Npm also allows the installation of other applications available to Node, such as debuggers, by using simple commands (“Npm”, n.d.).

+It is designed to work on many operating systems such as, Linux, which is used for the current UpStage server. It is also runnable in OS X, Microsoft Windows, Linux, FreeBSD, NonStop, IBM and a few more which give a lot of options to UpStage. But Linux will still be the best server OS since it is open source and the original project of Linux Foundation (“Node.js”, n.d.).

+Desktop IDEs for Node.js that are free to use and familiar to AUT students are: Eclipse IDE and NetBeans. These will be useful in AUT UpStage development.

+tutorials available for new users of node showing easy to follow steps in creating http server useful to UpStage in starting up a new version for students new to node (Cannaday, 2013).

+ Open source tools available for debugging and testing node (Selenium 2.0 –suitable for TDD and BDD; Vorlon.js – tool installable in npm).

Cons:

- The main advantage of node.js is the speed/performance aspect (asynchronous programming). This is not very useful due to the small user base in UpStage (making speed the least important factor in a language needed) and because the main issues the current version has (using Python’s Twisted) are testing and maintaining the code. There are some free tools identified in this research.

-UpStage development support, for upskilling or for other reasons, in node may not yet be available as it is still developing and is not as mature as languages such as C# and Java.

**Readability**

Pros:

+ should be easy for anyone with JavaScript knowledge

**Testability**

- JavaScript has a lack of features to assess modules or the code quality

+ Four types of tools needed for successful Node.js testing:  
A Testing Framework (Mocha, Vows, Intern)  
An Assertion Library (Chai, Assert)  
Stubs (Sinon)  
Module Control (Mockery, Rewire)   
(<https://github.com/mochajs/mocha>)

**Learnability**

+ NodeJS apps are written in JavaScript so it is easy to learn if you are familiar with JavaScript  
-+ It still has a long way to mature as it has a little online support in developing. There are social networks for Q and A also video sites for tutorials.  
(<http://nodetuts.com/> - for nodeJS video tutorial)

**Extendibility**

Pros:

+ For server development and web development

- Node.js structure is not mature enough to identify any packages that can be trusted

**Performance**

Pros:

+ Uses V8 developed by Google for use in Chrome and it compiles and executes at a high speed due to its use of single-threading for asynchronous programming (Sychay, 2014)

+ Non-blocking I/O

+ asynchronous programming : Ability to handle thousands of concurrent connections with minimal overhead on a single process compared to creating multiple threads that may show different results for different users of UpStage. This would be useful for the UpStage server.

Cons:

- doesn't support multi thread.

# Python

Python is an interpreted, object-oriented, high-level programming language with dynamic semantics. Python's simple, easy to learn syntax emphasizes readability and therefore reduces the cost of program maintenance. Python supports modules and packages, which encourages program modularity and code reuse. The Python interpreter and the extensive standard library are available in source or binary form without charge for all major platforms, and can be freely distributed.

**Pros**

* Python is a high level dynamically typed language. It is easy to pick up for new programmer and easy to use. Because it is a high level language, Python is typically shorter than other programming language like Java.
* If people are used to Python, it actually more efficient in developing stage. For developing same functionality, by using Python or other languages, the code written in Python is shorter.
* Python is easy integration with and extensibility using C and Java.
* For IDE, NetBeans 8.0.2 started supporting Python which is a bit easier for AUT students (We all familiar with NetBeans)
* For current UpStage, the code is written in Python and we need to maintain the current UpStage. We must learn Python. If we use Python for our new UpStage, we don't need waste extra time for learn other new programming languages.
* Debugging Python programs with PyDev is easy: a bug or bad input will never cause a segmentation fault. Instead, when the interpreter discovers an error, it raises an exception. When the program doesn't catch the exception, the interpreter prints a stack trace. A source level debugger allows inspection of local and global variables, evaluation of arbitrary expressions, setting breakpoints, stepping through the code a line at a time, and so on.
* Our clients know a bit more about scripting languages (like Python) than other programming languages. The code will be more readable for our clients if we use Python.

**Cons**

* Python do not has brackets for functions, if the code is hundreds or thousands lines, python will a bit a hard to read without proper comments.
* Python is an interpreted language. It is often many times slower than compiled languages.
* Code is hard to read, easy to lose.
* Python is a high level language, it runs slower.
* Python 3 is not backwards compatible. (The current UpStage is using Python 2.7.3) If we are going to use Python, we need to make sure the consistency of the version of Python.
* Although the current Upstage is written in Python, AUT students are taught by Java. As this is an ongoing project, if we decide to use Python, which means each semester for the new team members, they have to spend extra time for learning how to use Python.
* Absence from mobile computing and browsers. "Python is present on many server and desktop platforms, but it is weak in mobile computing; very few smart phone applications are developed with Python," says Carbonnelle. "It is also rarely seen on the client side of a Web application." For this reason, Python is not a good candidate for the Upstage, because we need to support mobile device for the Upstage.

**Readability**

Pros:

* Python is using Tab to separate the functions. The code in Python is clean and short.

Cons:

* Python is a dynamically typed language. We do not need to declare the type of variables that make the code become hard to read without proper comments. This is also a potential risk. The programmer might get some unexpected result (like input Boolean type but output is an integer)

**Testability**

Pros:

* Python has its own tools for unit testing called: unittest. It is the test module in the Python standard library. Its API will be familiar to anyone who has used any of the JUnit/nUnit/CppUnit series of tools.
* The Doctest is another testing tools for Python. The doctest module searches for pieces of text that look like interactive Python sessions in docstrings, and then executes those sessions to verify that they work exactly as shown.
* Doctests have a different use case than proper unit tests: they are usually less detailed and don’t catch special cases or obscure regression bugs. They are useful as an expressive documentation of the main use cases of a module and its components. However, doctests should run automatically each time the full test suite runs.

**Learnability**

Pros:

* Python is a high level language and dynamically typed language, it is easy to learn and easy to use.

**Performance**

Pros:

* Python can used in Linux/Windows/IOS Mac X. Python can be used in most popular platforms.

Cons:

* Python is a scripting language, it runs slower than other structure languages. (e.g. Java, C#)

# Proof of Concept

The project team has been researching on four programming languages whether those candidates are suitable to be chosen as a programming language to build UpStage. Java and C# has been selected as appropriate languages for UpStage development based on the researching criteria to measure and judge the programming languages. However, it was not easy to choose the best only one from those two languages, because both seem incomparably suited for UpStage. Therefore the team decided to make a very simple version of UpStage. As a prototype is a demonstration of what is actually feasible with existing technology and where the technical weak spots exist, the developers are able to understand better about the development environment (Bernstein, n.d.).

As UpStage is a platform for cyberformance, it is important to consider how to implement the real time web technology. Some of team members prefer to choose C# and some prefer Java. The individual’s preference of programming language is objective but still important. The team thought that is important for everyone to experience both languages to try to be fair. Therefore, we work in pairs and make two parts for the prototyping so that after completing the first part, the pairs can exchange the project to complete the second part. Then we see the differences between those two programming languages again.

## Process

* Part 1: Simulation of aspects of chat function

Two pair programmers work with the task to create a simple chat app in  
different programming language.

* + C# – Alyssa and Sia
  + Java – Jing and Joshua
* Part 2: Simulation of aspects of showing avatar

The pairs exchange the programming language to extend by adding an image  
button that adds or removes an image in real-time.

* + C# – Jing and Joshua
  + Java – Alyssa and Sia

## Criteria

1. Ease of Use  
   - How easy to code?
2. Readability  
   - How easy to understand the code written by other members?
3. Extendibility  
   - How easy to extend the code?
4. Responsiveness  
   - Always responsive?
5. Real-time  
   - How well it works when there are more than two clients at the same time?

## Libraries

**Atmosphere (Java)**

Atmosphere is a real-time framework for Java which allows asynchronous communication between two or more instances of a web application that uses the client-server architecture. It uses Webㄴockets to achieve asynchronous communication but also has fallbacks in cross-browser operations.

It supports most popular browsers, such as Firefox, Google Chrome, Internet Explorer and Opera, and also supports both iOS and Android, which is a requirement for UpStage as it has to be able to run on all popular platforms. If Java is used to develop the backend for UpStage, Atmosphere can be integrated with Java code by simply adding the necessary Atmosphere jar files as the project’s dependencies in the IDE.

The main advantage of Atmosphere for the UpStage project is that it works with all popular browsers, and the real-time functionality is ensured on all platforms without having to explicitly program for some platforms. This means the project team only has to come up with one solution, saving time and resources.

**SignalR (C#)**

SignalR is an open source library for ASP.NET developers that simplifies the process of adding real-time web functionality to applications. SignalR provides a simple API for creating server-to-client remote procedure calls (RPC) that call JavaScript functions in client browsers from server-side .NET code. SignalR also includes API for connection management (for instance, connect and disconnect events), and grouping connections.



It handles connection management automatically, and lets users broadcast messages to all connected clients simultaneously. It also supports server-push functionality, in which server code can call out to client code in the browser using RPC, rather than the request-response model.

## Findings

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Prototype Criteria: | | | | |
| Language | **Ease of Use** | **Readability** | **Extendibility** | **Responsiveness** | **Real-time** |
| C# | There wasn't much use of C# in the prototype but SignalR (the API for chat) was fairly easy to use. The pair programmers involved with chat were familiar with SignalR so it was easily implemented. The second pair completed the extension in a short period mainly because JavaScript was the main language involved. | The steps to complete the prototypes were documented and made it easier to understand what had been done. However when extending the code it was difficult to understand coding. In C# the unfamiliar layout made it hard to read. | Adding the avatar button was easily done | Was always responsive | yes |
| Java | Most of part 1 consisted of researching for a suitable API for the chat function (Atmosphere was used). Chat was easily developed but after atmosphere was used it became difficult to use the code. | In Java it was the Atmosphere's code that was difficult to read. | The code was difficult to extend due to the complex coding in the Atmosphere API (used for chat). | Was always responsive | The chat prototype was real-time but extension (i.e. adding avatar button) was not. In the end it was not solved. |

# Rationale for choosing C#

Each team member has chosen one programming language and researched on it to find out its advantages and disadvantages based on several criteria we have decided, so that we can decide which language is suitable for the new UpStage. The researched languages are Java, C#, Python and Node.js. We have also made simple prototypes to see how easy they are to implement and if they support real-time with multiple users. Each of these has its pros and cons, but we want to discuss why we should use C# and how it will work for UpStage.

The criteria that we have decided on are Readability, Testability, Learnability and Performance. To ensure that the code will be easily maintainable by future members of the UpStage team, the language chosen must be readable. It should also be compatible with popular or good testing tools to aid in testing. Ideally, the language is easily learnable as well, to minimize potential upskilling time of current and future members. Finally, it has to be able to support asynchronous communication and not take up too many resources, as being real-time is a criterion for the final product. However, as UpStage only has roughly 10 to 30 audience at a time for a given stage, this is not a high priority criterion. Instead, more emphasis is placed on testability, as being easily testable will help ensure that the project stays manageable and successful in the long run.

The criteria for the prototypes are Ease of Use, Readability, Extendibility, Performance Consistency and Real-time. Based on the research criteria, as we value the Testability, we decided that Java or C# should be used. The team developed prototypes for both languages in groups. The prototype is a simple chat room application, and after it is done, the other group tries to extend the prototypes with the functionality of displaying or hiding an image. These prototypes must be able to perform in real-time, and we tested this by opening multiple windows of the chatroom and seeing if anything done in one window is reflected in the other window.

C# is a modern, type-safe, high level object oriented language that is developed by Microsoft, running on top of the .Net framework. As UpStage is a web-based application used for online performances, the performers’ performance must be seen by the audience immediately without a page refresh. A technology that can make real-time web applications is required for UpStage and the SignalR library for C# allows this. Of course, other programming languages also have tools that provide real-time function such as Atmosphere for Java or twisted for Python, and Node.js supports real-time by itself.

Java and C#, being structured languages which use certain syntaxes such as curly brackets, make the code more structured and easier to follow. Scripting languages like Python and Node.js do not have this syntax, and thus code written in those languages can be seen as messy or overwhelming by people not familiar with them. This will become more and more relevant as the code base grows. This is an important thing to consider as making the code easily readable will improve the maintainability of the code, and makes it easier to locate the source of bugs. However, Python and Node.js code are shorter due to the lack of syntax, which makes them easier to read in their own rights. As the project team is more experienced in Java and structured languages in general, we feel that the advantages of C# are more applicable than those of Node.js and Python.

When we tried to extend the prototypes, we had difficulty extending the Java prototype as the real-time library we used, Atmosphere, was confusing and hard to read. On the other hand, the code for the C# equivalent which is SignalR, is very short and easy to read and understand. This ties to the ease of use of the two languages; Java and C# is relatively similar in terms of their ease of use, but the real-time libraries we found is vastly different from each other. As SignalR code is more organized and easier to read and implement, we found that C# is overall more easy to use and readable in this project.

Ease of testing is also a big part of the reason we chose C#. C# has access to an excellent IDE with powerful debugging tools in Visual Studio, and the debugger is easy to use. Python has plug-ins for existing IDEs such as PyDev for the eclipse IDE that allow testing of Python code, while successfully testing Node.js code requires getting and learning how to use 4 components, namely the testing framework, an assertion library, stubs and a module control tool. These are unnecessarily complicated, which is something we want to avoid.

Learnability is another factor to consider. Node.js is written in JavaScript, which is a relatively simple language to learn. Some of the papers taught at AUT also use JavaScript, which means it is likely that current and future team members have some knowledge on JavaScript already, and thus reduce the amount of upskilling needed. The front-end of the current UpStage is also written in JavaScript, which further increases its advantages. Team members should already be familiar with Java as it is the main language taught in AUT, so it also requires minimal upskilling. Python is a required learning regardless of the language we decide to use, as the current UpStage uses Python, and it needs to be maintained. However, for future team members, when the current UpStage no longer needs to be maintained, Python will no longer have this advantage. C# is similar object oriented, which is the same as Java, and thus it is relatively easy to learn.

In terms of performance, Node.js is ideal as it is relatively lightweight, unlike Java and C#. It is also faster than those languages, because it does not utilize blocking threads and thus do not have too much downtime. Also, unlike Python which is interpreted during runtime, it does all processing beforehand, thus ensuring that performance is consistent during runtime. While Java can be faster in certain cases, it is prone to inconsistent performance as classes are only loaded when first referenced, or garbage collection can happen at random times, which can cause performance spikes. Node.js also has non-blocking I/O which ensures smooth performance during runtime. C#’s SignalR library handles these and ensures asynchronous communication at all times, even with multiple users, and this is a good thing to have as the clients have expressed that they want a stable performance with varying numbers of visitors. This means that while both Java and C# can perform real-time operations, C# is more stable in terms of performance.

There are other advantages we did not mention here but it is worth to choose with these reasons. As developers ideally want to create small and fast micro-services that are easy to maintain, C# is a good choice for its ease of implementation and testability. It will help the team make a great application that is a good quality and easy to maintain. Node.js can still potentially be used for certain parts of UpStage if needed, particularly parts which require consistently high performance or do not require rigorous testing but C# is the main language that will be used.

# References

Atmosphere (n.d.). In *Wikipedia.* Retrieved from <https://github.com/Atmosphere/atmosphere/wiki>

Benton, N., Cardelli, L., & Fournet, C. (2004). Modern concurrency abstractions for C#. *ACM Transactions on Programming Languages and Systems, 26*, 769-804.

Bernstein, L. (n.d.). *Importance of software prototyping.* Retrieved September 29, 2015, from <http://condor.depaul.edu/sjost/hci430/documents/prototypes-general/BernsteinPrototyping.htm>

Cannaday, B. (2013). *An absolute beginner’s guide to Node.js*. Retrieved from [http://blog.modulus.io/absolute-beginners-guide-to-nodejs](http://l.facebook.com/l.php?u=http%3A%2F%2Fblog.modulus.io%2Fabsolute-beginners-guide-to-nodejs&h=GAQHzDwihAQGCXWLyhiJbzi5MvY42SYGOrX-AyGXSiRB7Fg&enc=AZNHloibMH99FrLnHKKioURtOqUsnsUTvOysn8br-Q3xWpQ_jICcfNTxswwlyKk1MW39lj7byAB9fG49OaCNMBj1YcnCi3Tljop5wTfX3YtjAM6zySGooi2jLEZXIcZM6CzK8XXOBu-KRRpyYNK2JAK8tgxPXhI9a01_yeyVT1NxdQ&s=1)

Java (programming language). (n.d.). In *Wikipedia.* Retrieved from  
<https://en.wkipedia.org/wiki/Java_(programming_language)>

Kopp, M. (2011). *The impact of Garbage Collection on Java performance*. Retrieved from  
[http://apmblog.dynatrace.com/.../the-impact-of-garbage.../](http://apmblog.dynatrace.com/2011/03/24/the-impact-of-garbage-collection-on-java-performance/)

Node.js. (n.d.). Retrieved from <https://nodejs.org/en/>

Node.js. (n.d.). In *Wikipedia.* Retrieved from <https://en.wikipedia.org/wiki/Node.js>

Npm (software)). (n.d.). Retrieved from <https://en.m.wikipedia.org/wiki/Npm_(software))>

Reitz, K. (2014). *Testing Your Code.* Retrieved from <http://docs.python-guide.org/en/latest/writing/tests/>

SignalR. (n.d.). *What is ASP.NET SignalR*. Retrieved August 19, 2015, from <http://signalr.net/>

Python (programming language) (n.d.) in *Wikipedia*. Retrieved From <https://en.wikipedia.org/wiki/Python_%28programming_language%29>

Krill, P. (2015). *A developer's guide to the pros and cons of Python*. Retrieved from <http://www.infoworld.com/article/2887974/application-development/a-developer-s-guide-to-the-pro-s-and-con-s-of-python.html>

*What is the difference between C, C++ and C#?* (n.d.). Retrieved from <https://www.quora.com/What-is-the-difference-between-C-C++-and-C>

.NET Blog. (2014). *.NET Core is open source.* Retrieved August 19, 2015, from <http://blogs.msdn.com/b/dotnet/archive/2014/11/12/net-core-is-open-source.aspx>