**GWT – Pros and Cons**

I love JavaScript. With the advent of [jQuery](http://jquery.com/" \o "" \t "_blank) and [Mootools](http://mootools.net/" \o "" \t "_blank), my love for JavaScript has only increased plenty-fold. Given a choice I would use either of the aforementioned frameworks for any web application I develop. But being in the service industry, time and again I have to succumb to the client’s pressure and work in their choice of technology – whether or not it is the right one (*The one who pays the piper calls the tune. Isn’t it?*). One such client exposed me to the world of [GWT](http://code.google.com/webtoolkit/).

I have given GWT a shot couple of years back on the day it was released. I didn’t like it that much then, so I dismissed it and never returned back. But, over the past six months working on this project I have a slightly different impression on this framework. I still cannot say that GWT is the next big thing since sliced bread, but at least it is not as bad as I thought it was. I have just documented my observations, both good and bad during the course of this project and thought some fellow developer might find it useful while evaluating GWT.

**Pros:**

1. If you are a Java veteran with experience in Swing or AWT, then choosing GWT should be a no-brainer. The learning curve is the least with this background.
2. Even if you are not experienced in Java GUI development, the experience in working on server-side Java for years will come in handy while developing GWT apps
3. You can create highly responsive web applications with heavy lifting on the client-side and reduced chattiness with the server-side
4. Although there are numerous JavaScript libraries out in the wild and most of them are worth their salt, many conventional developers don’t understand its true power. Remember, a powerful language like JavaScript is a double-edged sword. If you don’t know how to use it, even you won’t be able to clean the mess you create
5. You can migrate from a typical web application to a GWT application iteratively. It is not an all or nothing proposition. You can use a clever trick called JSNI to interact with loads of JavaScript functions you already possess. But it is always better to move them to GWT sooner rather than later
6. The IDE support for GWT cannot be better. Java IDEs have matured over the past decade to be one of the best in the world and GWT can take direct advantage of it
7. The integrated debugging beauty is something you can kill for. The excellent debugging support offered by the mature Java IDEs is one feature that could sway anybody’s decision in favor of GWT
8. The built-in IDE support to refactor Java code can directly be put to good use to maintain a simple design at all times. Doing this in JavaScript is not for the faint at heart
9. The IDE syntax highlighting, error checking, code completion shortcuts etc are overwhelming – to say the least
10. GWT is being actively developed by Google. We know that the project is not going to die off anytime soon. Until now their commitment towards the project says a lot about its future in the industry.
11. The community behind the project is also a big PLUS. Discussions take place daily in Stack overflow, discussion forums, wikis and personal blogs. A simple search with the right keyword could point you in the right direction
12. GWT is a well thought-out API; not something that was put together in a hurry. This helps you as a developer to quickly comprehend the abstractions and makes it really intuitive to use
13. You can use GWT’s built-in protocol to transfer data between the client and the server without any additional knowledge of how the data is packaged and sent. If you prefer more control, you can always use XML, JSON or another proprietary format of your choice. Even in that case, while using JSON, you don’t have to use an non-intuitive java JSON library. You can use JSNI to ‘eval’ the JSON using straight javascript. Cool huh!
14. You have the advantage of being able to use standard Java static code analyzers like FindBugs, CheckStyle, Detangler, PMD etc to monitor code and design quality. This is very important when you are working in a big team with varying experience.
15. You can use JUnit or Test NG for unit testing and JMock or another mock library for mocking dependencies. Following TDD is straight-forward if you already practice it. Although there are JavaScript based unit testing frameworks like jsunit and qunit, come on tell me how many people already know that or are itching to use that.
16. The GWT compiler generates cross-browser JavaScript code. Today, any marketing person who says this will probably be beaten. It has now become a basic necessity, not a luxury
17. The GWT compiler optimizes the generated code, removes dead code and even obfuscates the JavaScript for you all in one shot
18. Although the compilation process takes hell a lot of time, you don’t have to go through that during development. There is a special hosted mode that uses a browser plug-in and direct java byte-code to produce output. That is one of the main reasons you are able to use a Java debugger to debug client side code.
19. Rich third-party controls are available through quite a few projects like Smart GWT, Ext GWT etc. They are well designed, easy to use and theme-able. So, if you have a requirement where existing controls don’t just cut it, you should be looking into one of these projects. There is a really fat chance that one of those components will work out. Even if that doesn’t work out, you can always roll out your own.
20. GWT emphasizes the concept of a stateful client and a stateless server. This results in extremely less load on the server where many users have to co-exist and high load on the client where only one user is working
21. I18N and L10N are pretty straight-forward with GWT. In fact locale based compilation is taken care by the GWT compiler itself. The same cannot be said about regular client-only frameworks
22. GWT comes built-in with browser back button support even while using AJAX. If you are an AJAX developer, I can almost feel your relief. This is priceless.

**Cons:**

1. GWT is a fast developing project. So, there are a lot of versions floating around. Many functions, interfaces and events get deprecated and keeping up with their pace is not too much fun when you have other work to do
2. There were quite a few GWT books during the beginning. Not so much these days. For example, I haven’t found many books on the 2.0 version of GWT. This leaves us only with Google’s documentation. I agree that the documentation is good and all, but nothing can beat a well written book
3. GWT is not fun to work with. After all it is Java and Java is not a fun language to work with. If you add the fact that entire layouts and custom controls should be created in java, you can easily make a grown programmer cry. With the introduction of UI binder starting version 2.0, that problem is kind of solved, but now you have a new syntax to learn.
4. The Java to JavaScript compilation is fairly slow, which is a significant con if you choose GWT.
5. I personally prefer defining structure in HTML and styling it using CSS. The concepts used in HTML are clean and straight-forward and I have years of experience doing just that. But in GWT, I am kind of forced to use proprietary methods to do the same. That combined with the fact that GWT doesn’t solve the styling and alignment incompatibilies for me compounds the problem. So, writing layout code in GWT is something I despice. But with UI Binder and HTMLLayout from version 2.0 onwards, I feel I am back in my own territory
6. It requires some serious commitment levels to get into GWT, coz, after that, a change in client side technology could require a complete rewrite of your app, as it is a radically different approach than other client side frameworks
7. There is not a defined way to approach an application development using GWT. Should we use only module per app or one module per page or somewhere in between. These design patterns are slowly evolving only now. So, typically people tend to develop all in one module until the module size goes beyond being acceptable and then they refactor it into multiple modules. But if it is too late, then refactoring could not be that easy either
8. Mixing presentation and code doesn’t sound right although typical desktop GUI applications does just that. But these days, even desktop application frameworks like Flex and Silverlight have taken an XML based declarative approach to separate presentation from logic. I think GWT 1.x version had this disadvantage. With the introduction of UI Binder starting from version 2.0, I think this disadvantage can be written off although it is yet another painful XML language to learn
9. You would often be punching in 3X to 5X more code than you would with other client libraries – like jQuery – to get simple things done
10. You should also remember that with GWT, the abstraction away from HTML isn’t complete. You’ll still need to understand the DOM structure your app is generating in order to style it, and GWT can make it harder to see this structure in the code.
11. GWT is an advantage only for Java developers. Developers with .NET or PHP background won’t gain anything here
12. If you have tasted the power of JavaScript and know how to properly utilize it to your advantage, then you will feel crippled with an unexpressive language like Java