The Future of GWT Report

Statistics of today and desires of tomorrow

2012

Let the community guide the way

“2012 has brought many exciting features to GWT and it has also changed how GWT will be developed in the future. While the product and the community are stronger than ever, people have been asking what happens next with GWT?

Google announced in June that the development model for GWT will be made more open. To facilitate the move and future development, the GWT Steering Committee was founded. Vaadin is proud to be a part of it together with Google, Sencha, RedHat and others. To collect input on what the users of GWT are doing with it and what are their expectations, we stepped forward and asked.

The Future of GWT Survey is the result of the work of Vaadin, Ray Cromwell (Google representative and acting GWT Steering Committee Chair), Artur Signell (VP of R&D, Vaadin Representative), Mike Brock (Project Lead, JBoss Errai, RedHat Representative), David Chandler (Developer Advocate, Google), Daniel Kurka (developer of mGWT, GWT-phonegap) and Bhaskar Janakiraman (Google) - and you can see their comments and reactions throughout this report.

With over 1300 respondents (1349 to be exact), who answered over 30 questions, we believe that this is the most complete survey of the GWT community ever completed. We’d like to thank all the survey respondents for their time and honesty, and look forward to your comments on the data!”

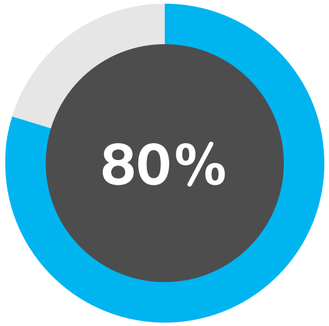
Joonas Lehtinen CEO Vaadin

Contents

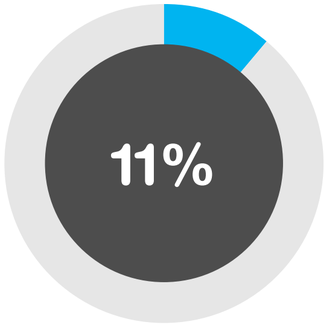
Section 1:

Learn who, where and how GWT is used today

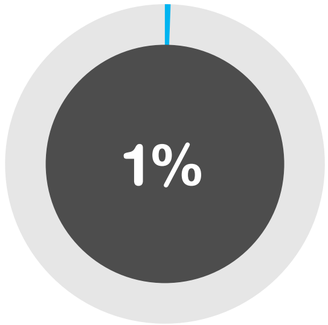
How is GWT used today?



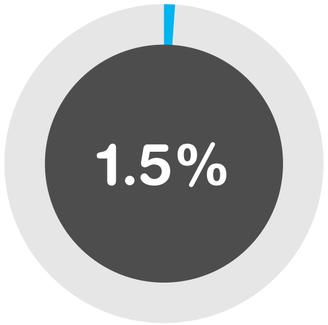
Business application



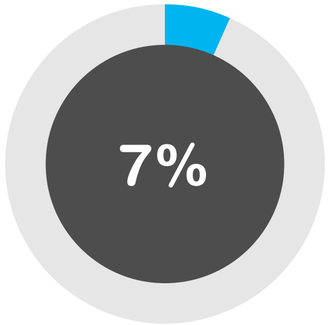
Content-rich website



Game



Portlet

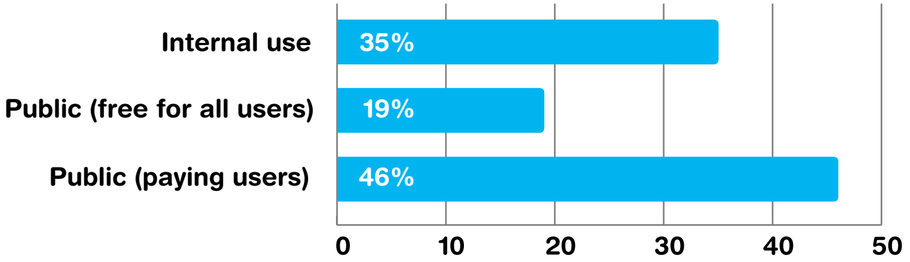


Other

1.1 All work and no play makes Duke a happy puppy

As might be expected GWT users are much more likely to be building business applications (79%) over portlets or games, but content-rich websites also make up for a significant portion of usage (11%).

What you might not have expected, is that more applications are being built for public use (65%), rather than for internal usage (35%). Even the publicly used applications are very business oriented, 71% of them are for paying customers.



"I feel that the number of portlets is surprisingly low, unless they are included in "business applications".

Artur

"The large number of apps developed for public use show that we need to pay a lot of attention to security."

Bhaskar

"This surprised me - The number of people working on paid applications is higher than I expected."

Daniel

"Historically, if you look at many of the consumer facing apps done in recent years, they've been done with Ruby/Python/PHP in the backend and jQuery, Backbone, or some other JS library, so it is quite a surprise to see a healthy market of consumer-facing, public, for-fee GWT applications. This might be a signal that GWT is better for modern HTML5 and Mobile applications and would be a significant benefit for a large part of the developer base."

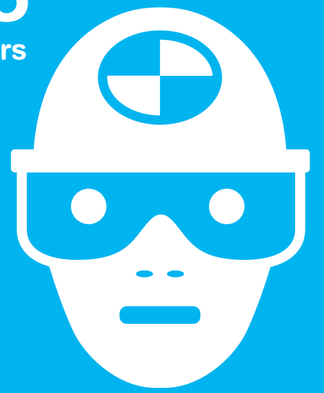
Ray

1.2 Your average GWT team

The average team on a GWT project is made up of just over 12 people. This is a combination of Back End Developers, Front End Developers, Designers, Testers, Project/Product Managers, and a few others. We did not ask respondents who those “others” might be... perhaps this could be a customer representative, someone focused on UX, or perhaps GWT teams have found a way to bring the elusive Bigfoot into a corporate environment.

3.7
Back End Developers

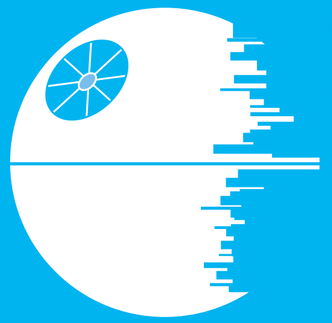
2.5
Testers



0.9
Designers

2.9
Front End Developers

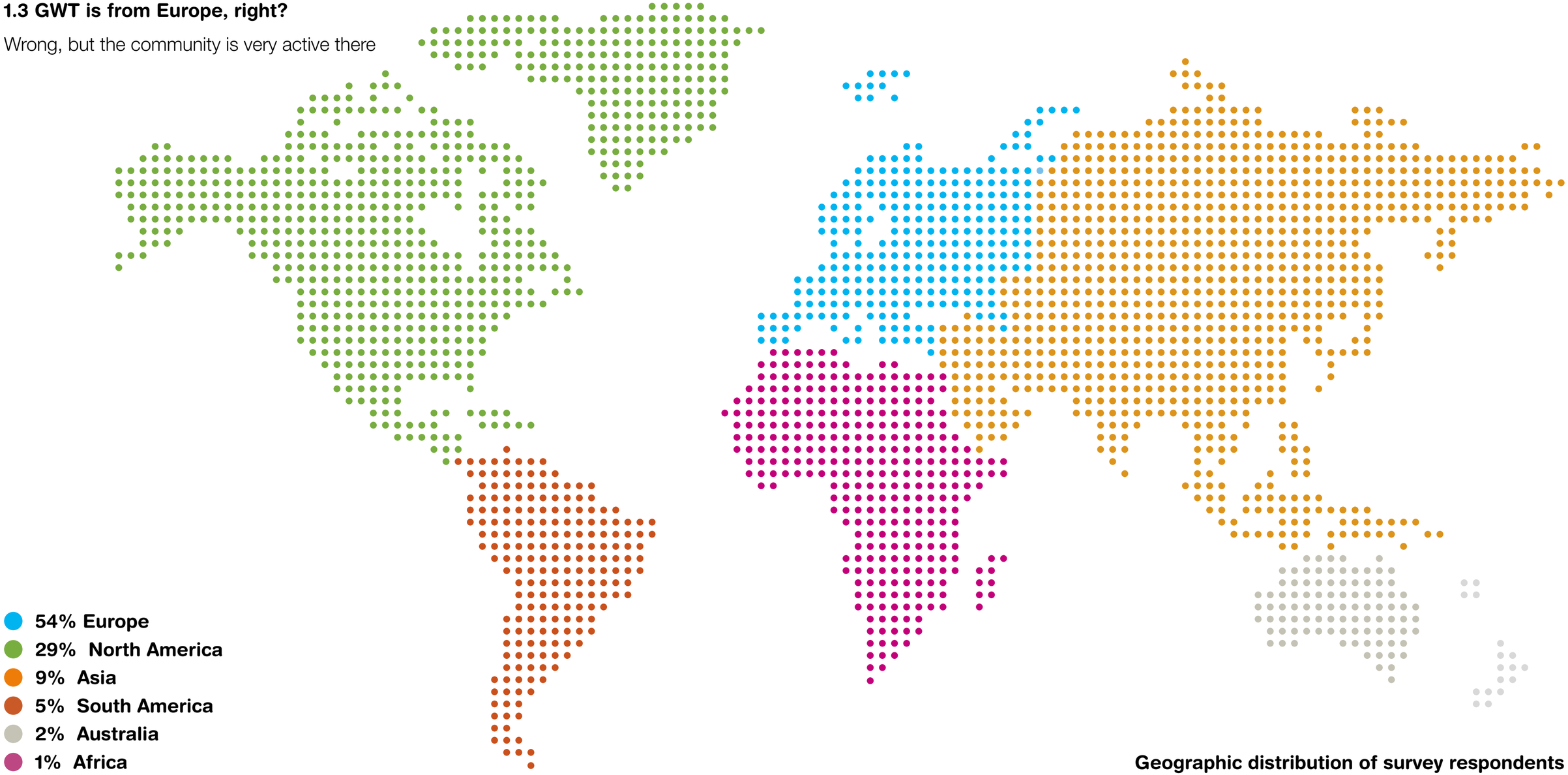
1.3
Project / Product Managers



1
Other

1.3 GWT is from Europe, right?

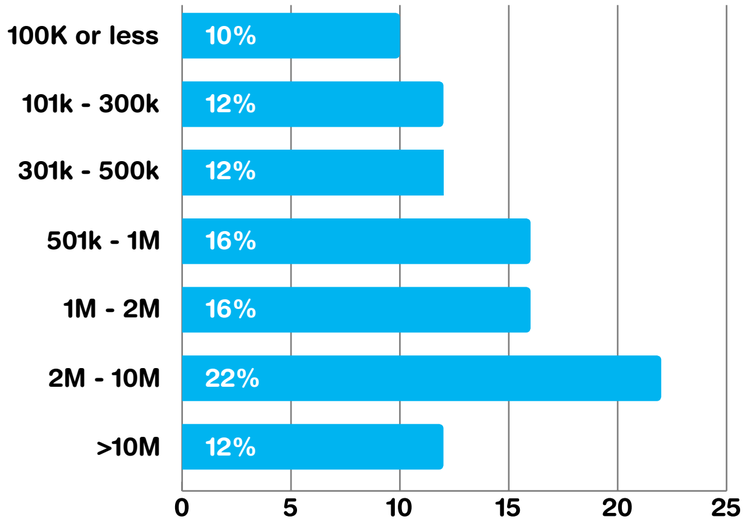
Wrong, but the community is very active there



Geographic distribution of survey respondents

1.4 Look at the size of those apps!

We asked our respondents to tell us about the largest application they were building with GWT. The size of their (uncompressed obfuscated) JavaScript range from 100k to >10M, and the distribution is spread rather evenly across the spectrum. GWT developers are clearly building some of the largest client-side web applications in the world.



Size of the uncompressed obfuscated JavaScript

One of the things that people often ask me when I'm giving talks is "when should I use GWT? Or when not to use GWT?". My answer is kind of borne out by what I see here: GWT is the right tool for building large, maintainable applications. JavaScript is quick and nimble. But when the app gets large, maintainability quickly falls off a cliff.

Mike

"When different people are working on the same JavaScript code, you have to keep up conventions that are not enforced by a compiler > this makes it difficult to scale teams or maintain applications over time, whereas with GWT, these conventions are enforced by the compiler, making the technology much easier to use, grow, and maintain"

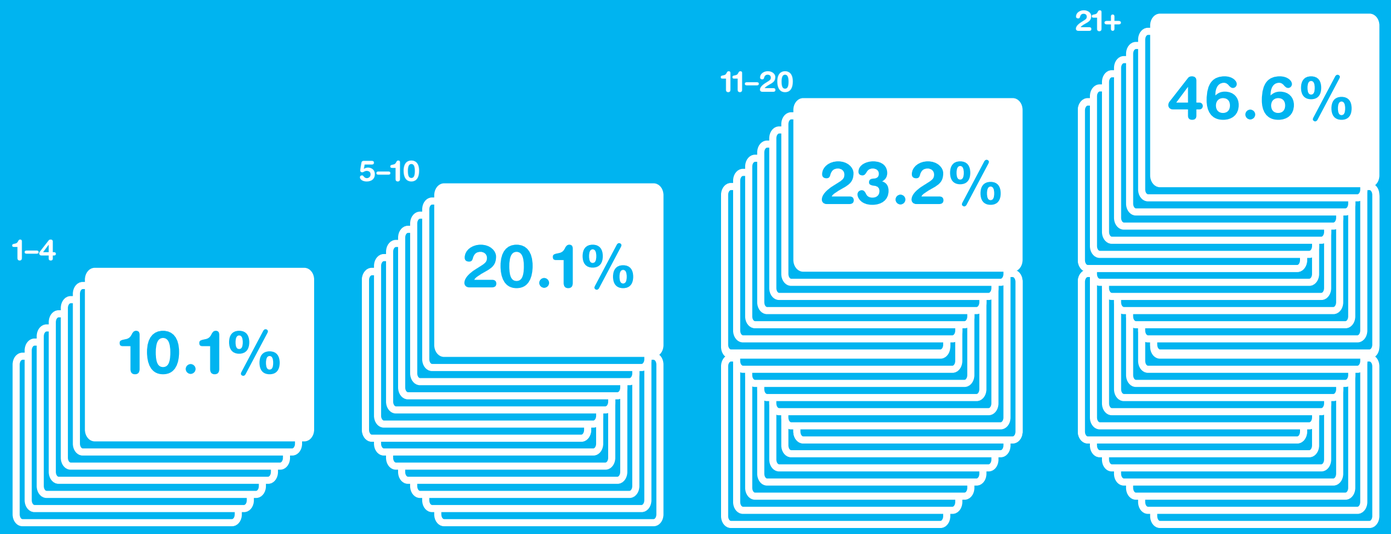
Daniel

There's always been a great debate about static typing vs dynamic typing and tooling in the industry. I think this lends evidence to the idea that larger scale code bases and teams derive more benefit from static typing (which tends to serve as a natural documentation of the code base and a way to enhance collaboration), and this is where GWT shines.

Ray

"GWT is the right tool for building large, maintainable applications."

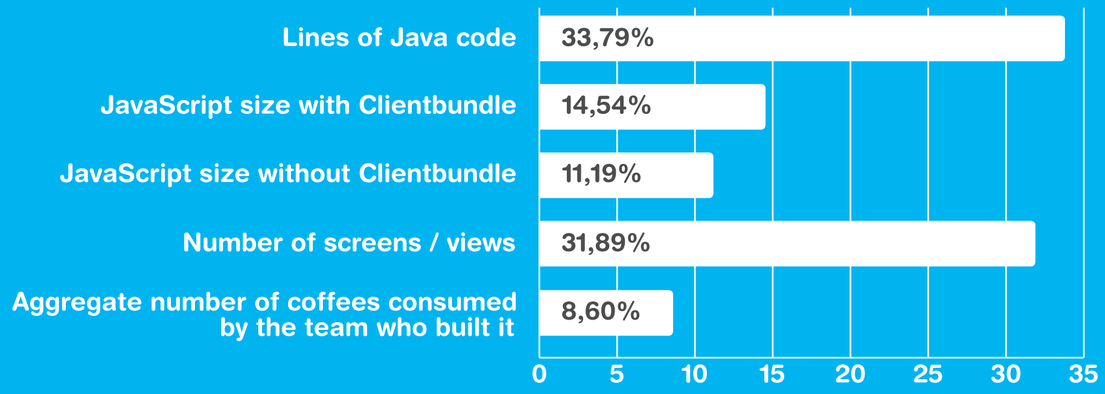
Mike Brock



Number of screens

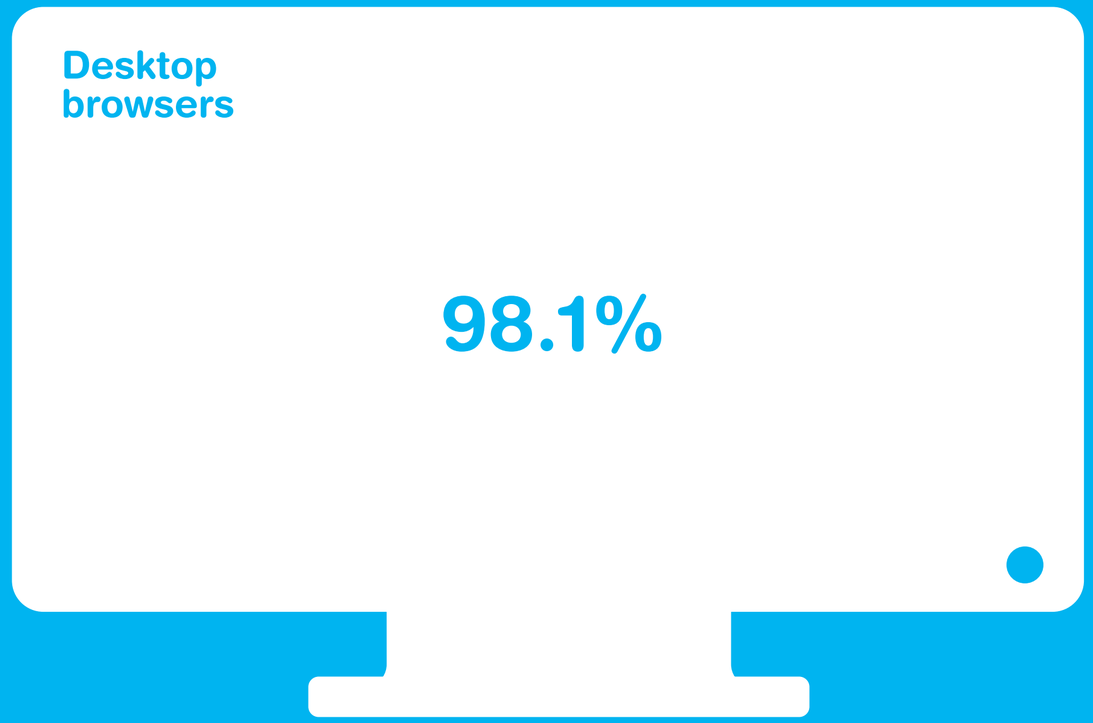
1.5 How do people measure app size?

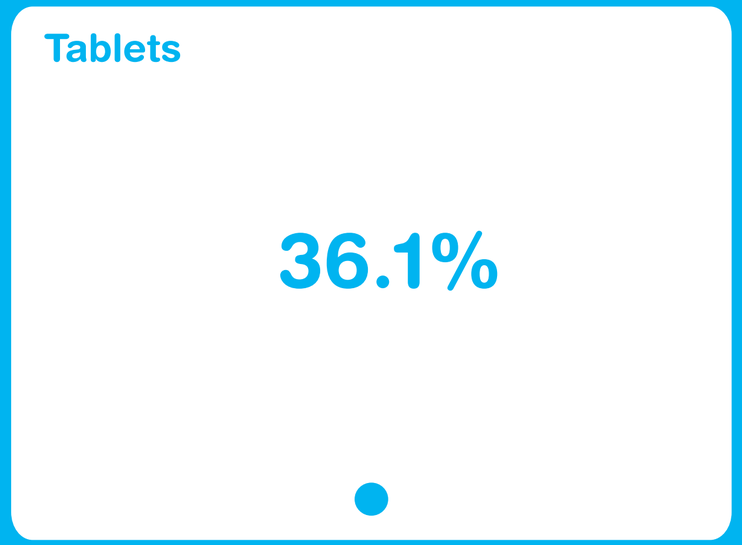
We also asked what respondents thought would be the best way to measure the size of their app to be able to ask correct questions in the future. It seems that “Aggregate number of coffees consumed by the team who built it” is a valid measure of application size.

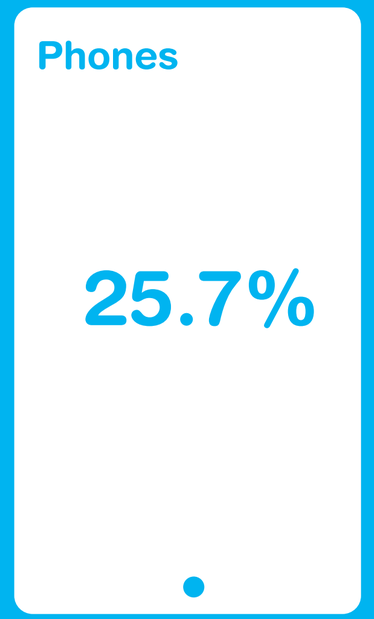


1.6 Going mobile

GWT is a versatile technology that allows developers to create application UI for desktop, tablet, and mobile consumption. As can be expected, over 98% of apps support desktop browsers, but we found it interesting that tablets had overtaken phones (at least when it came to support from GWT-based apps). In US, the number of apps supporting tablets was as high as 46%, while it was just 34% in Europe.







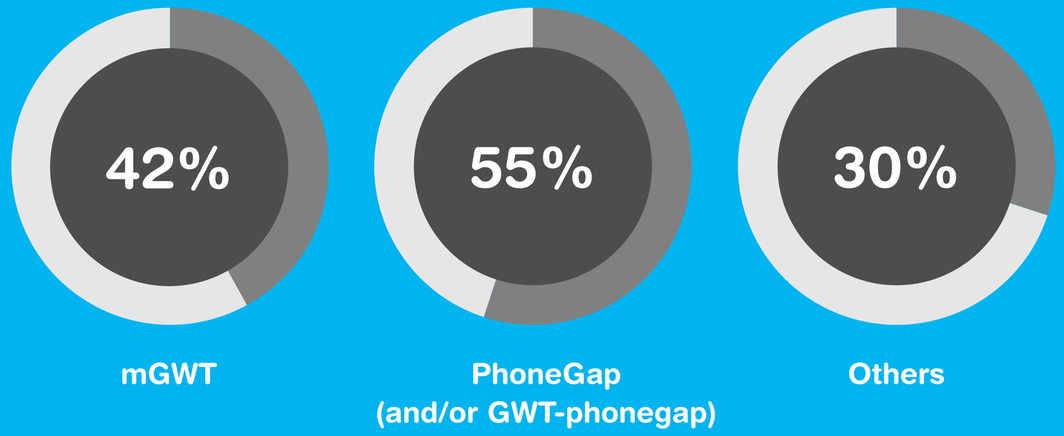
"Since gwt is used extensively in the enterprise, this may explain why tablets are more popular than support for phones"

Daniel

What kind of devices does your app support?

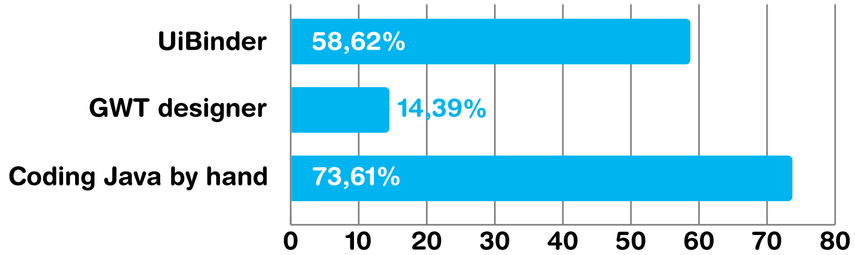
1.7 Most popular technologies for mobile development with GWT

While you can develop mobile applications without any additional technologies in GWT, there are many frameworks around to help you. mGWT and PhoneGap are much more widely used than any others. Out of the people who use some technologies for tool for using some tool, these two are used by 42% and 55% respectively. Most of the ones using PhoneGap also use GWT-phonegap. While 30% of people chose “Other”, we failed to find a trend there. People are building apps and custom frameworks on top core GWT as well as using technologies not related to GWT for building their mobile UIs.



1.8 Java > XML >> Designer

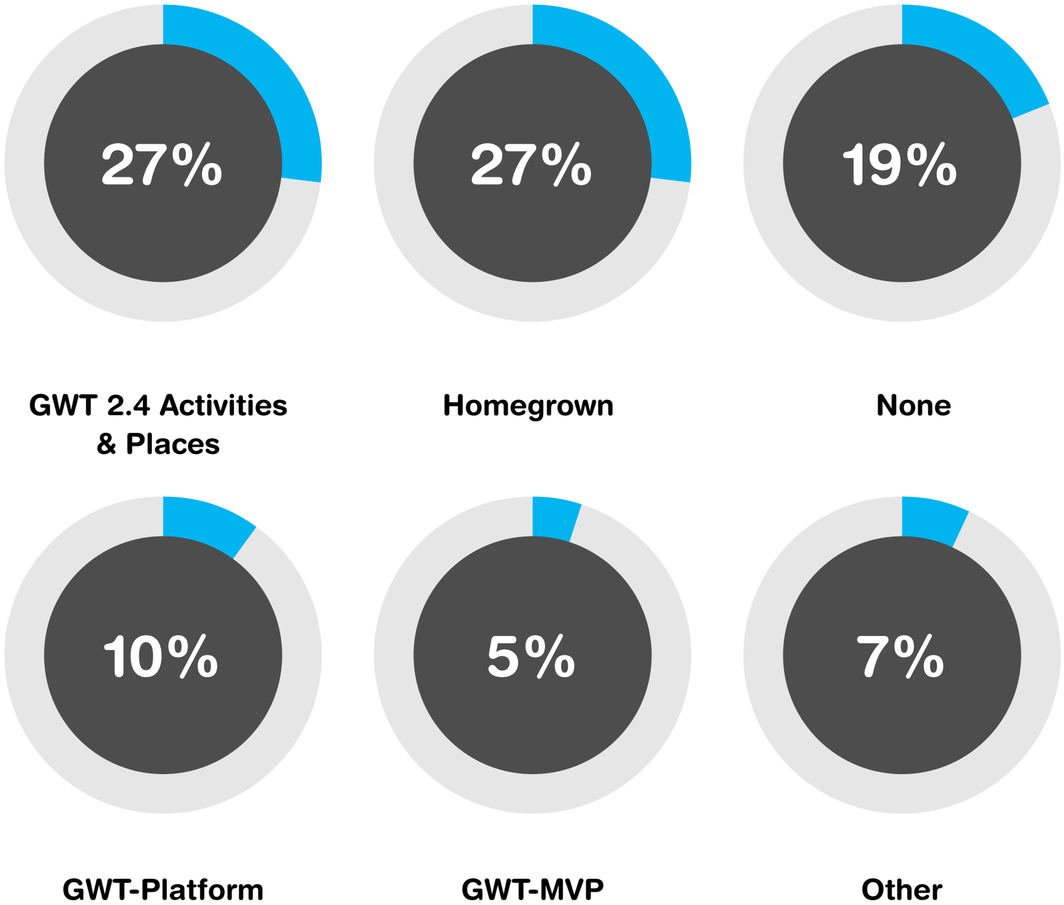
Coding Java by hand is still the most popular method of building the UI on GWT projects. What surprised us was how popular UiBinder has become. Now 59% of GWT developers are using XML to build their UIs declaratively. On the other hand, GWT designer has not been able to get much traction.



How UIs are built

1.9 Model-View-Presenter

Opinions are split on whether and what kind of pattern one should use to organize component oriented UI. One of the most popular patterns used is MVP and there are quite a few frameworks for GWT to help use of the pattern.



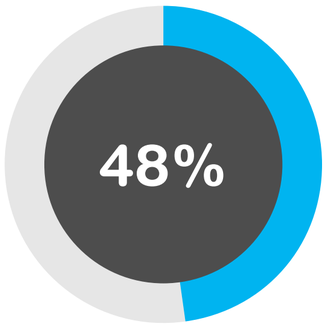
1.10 Crash test dummies

Two of every three respondents say that their apps are either tested manually or not tested at all. It is striking that only one third of developers use automated tests for their GWT apps. The most popular tools used today for testing GWT apps are Selenium and GWTTestCase.

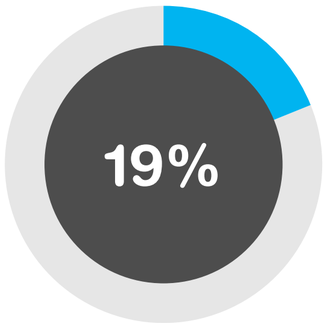
> Its surprising to see so much manual testing. This should be another area of focus for future GWT releases – support for automated testing of GWT apps.
>
> **Bhaskar**

> 50% of the people doing manual testing is not that great since there are very good tools out there for doing automated tests, such as Jucikito.
>
> **Daniel**

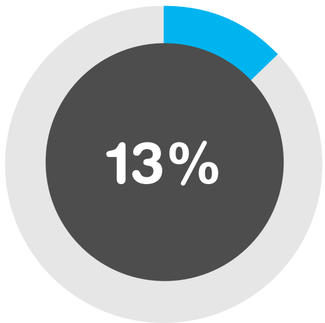
> The volume of manual testing indicates a demand for a tool for automation. There have been tools in the past like Selenium IDE/Recorder that let developers record manual tests and play them back, but it could be made a lot better with deeper integration into IDEs like IntelliJ and Eclipse to make it far less painless to record and generate tests.
>
> **Ray**



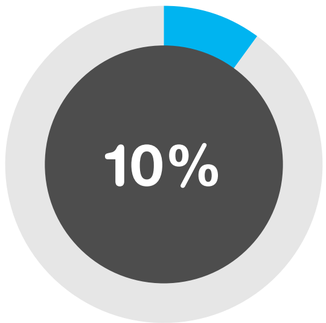
Manual testing



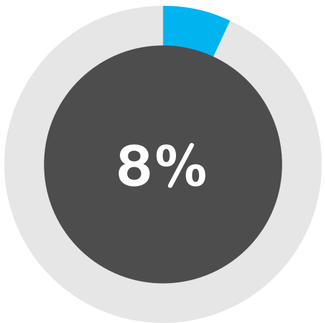
Selenium



GWTTestCase



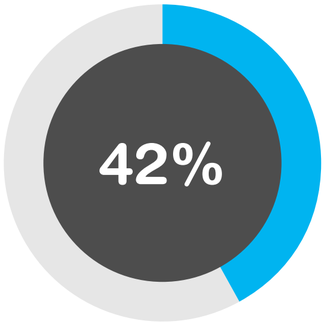
I’m not testing my UI



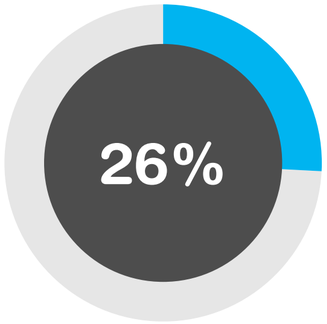
Other

1.11 Extensions everywhere

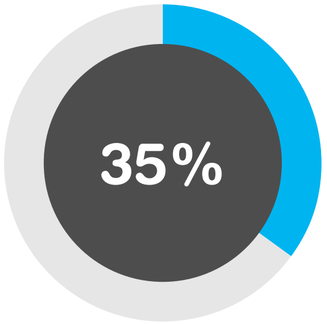
People are using a lot of extensions with GWT. Here are the key areas where extensions are used most. Obviously, respondents were able to choose multiple areas where they use extensions, so results do not add up to 100%.



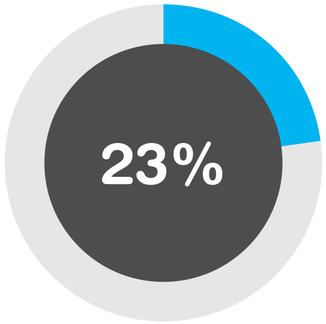
UI functionality (like GWT-dnd, GWT-fx)



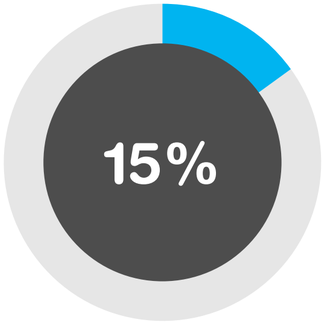
Data access (like GWT-rpc, GWT-dispatch)



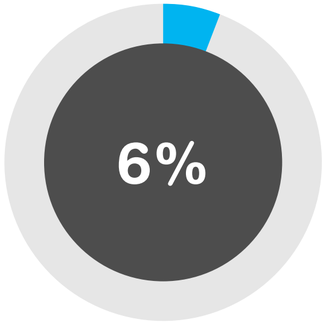
Application structure (like gin, GWT-presenter)



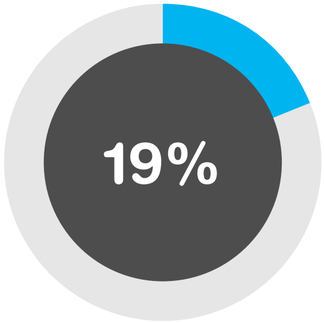
Access to services (like Google APIs, GWT-bootstrap)



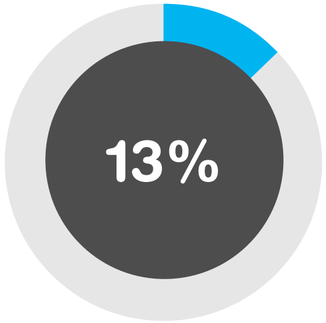
Application functionality (like GWT-platform)



Data handling (like gilead)



Don’t use any extensions

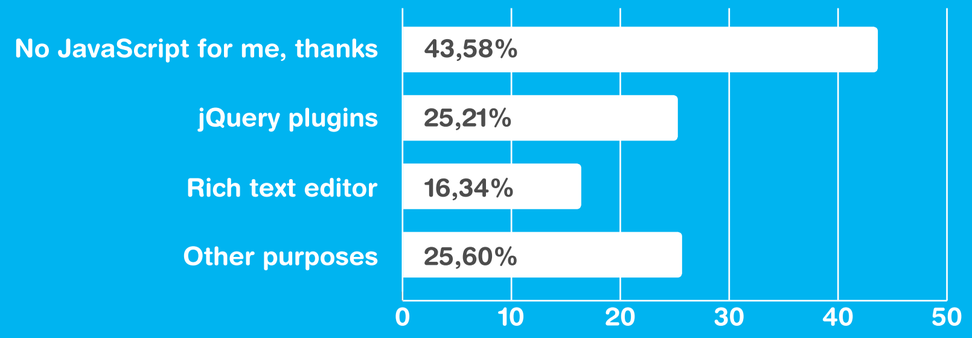


Other (please specify)

1.12 Use of JavaScript

Using big chunks of JavaScript in a GWT application is technically possible, but it also has a downside: the GWT Compiler can not optimize those parts of a project. So you should always ask yourself if their isn't an equal implementation in Java (GWTQuery instead of JQuery) that could be used instead or if rewriting something from scratch might be an option.

Daniel



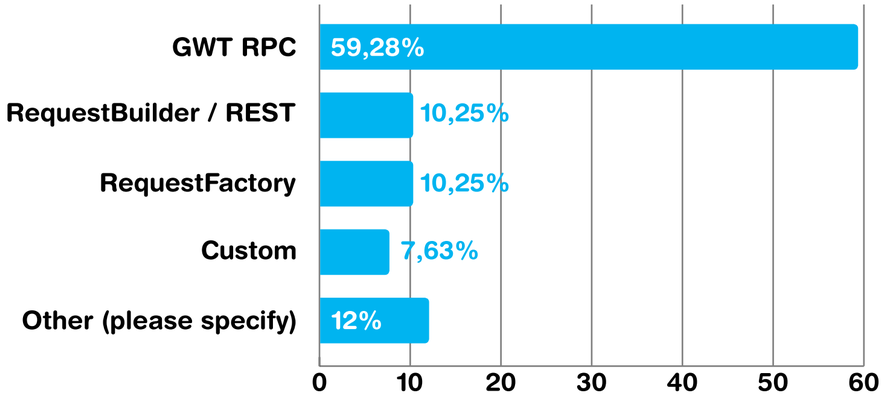
1.13 Backend communication

Not surprising to see GWT RPC winning given how easy it is to use with existing Java code.

Ray

However, Using GWT RPC for mobile phonegap applications is not such a good idea since client and server always need to have the same versions. With apps in store you are not in control of the update cycle, so requestfactory or autobeans are way better in this kind of situation.

Daniel



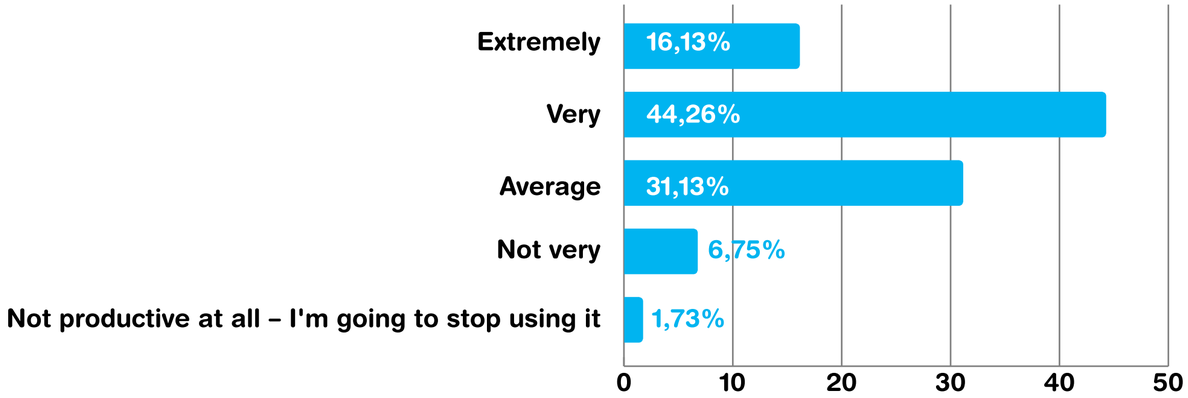
Section 2:

Take a crash course on the strengths and weaknesses of GWT

The Strengths and Weaknesses of GWT

2.1 Feeling productive?

Normally, when I'm asked to talk about my feelings, I suffer flashbacks to 7th grade dances, with all the boys on one side of the room, and all the girls on the other... my eyes begin to dart from side to side, and my palms get clammy and sweaty... It's not pretty. When we asked our respondents for their feelings on productivity with GWT, I'm pleased to report that everyone was much more positive! 60.39% of respondents reported feeling "Very" or "Extremely" productive when using GWT - as opposed to the 8.48% who were "Not Very" or "Not productive at all".

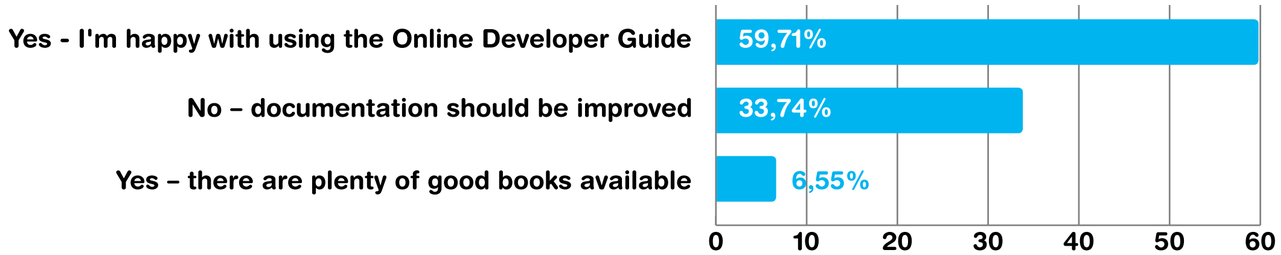


How productive respondents feel using GWT

60.39% of respondents reported feeling "Very" or "Extremely" productive when using GWT

2.2 We do RTFM

One of the key elements of the productivity for a framework is how well it is documented.
Nothing kills productivity like endless guessing and using sources as the documentation.

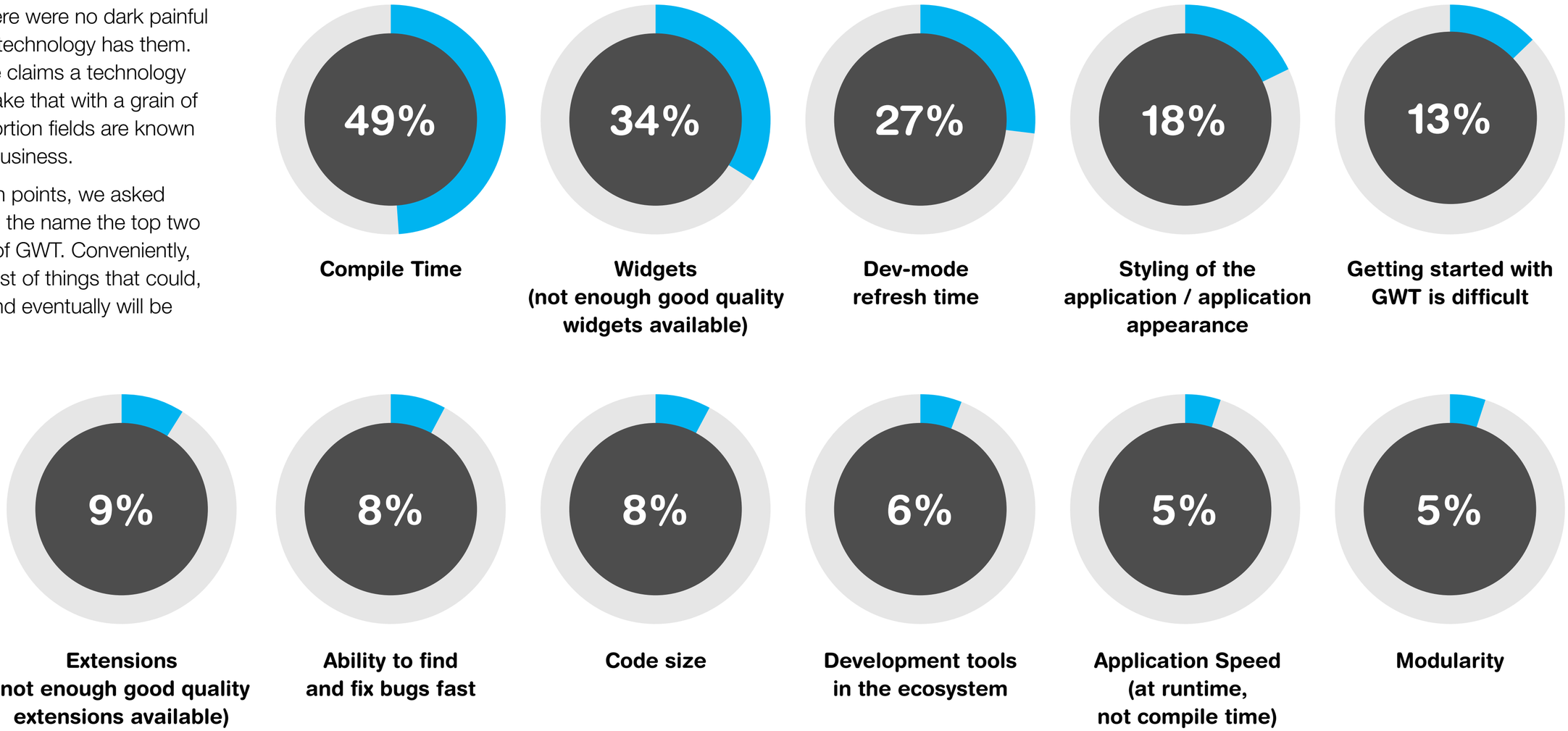


How good is the documentation

2.3 Welcome to the dark side of GWT

You thought there were no dark painful secrets? Every technology has them. When someone claims a technology to be perfect, take that with a grain of salt: reality distortion fields are known to exist in this business.

To discover pain points, we asked people to name the name the top two worst features of GWT. Conveniently, this gives us a list of things that could, should, must and eventually will be improved.



Worst features of GWT

2.4 Read while Compiling

The compile time is the #1 pain point for GWT developers. Fortunately we asked a couple of questions about it and learned the reason. Because GWT really shines on building large apps, guess what - people build really large apps with it. And they take a while to compile.

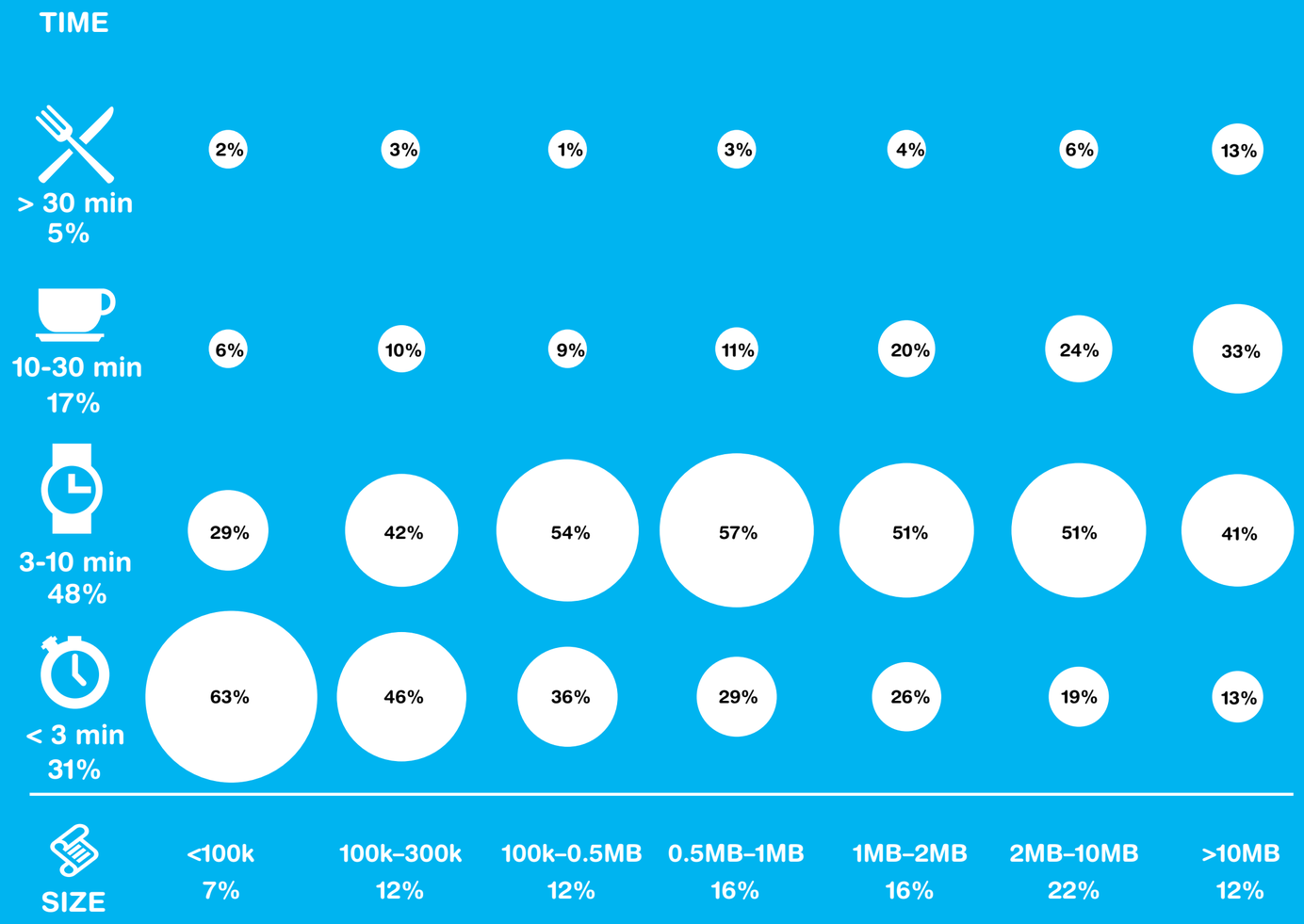
There are three phases of compile time: Precompilation, Optimization of Permutations, and Linking. Obviously, the second phase can grow proportional to the number of permutations. At Google, we shard out GWT permutations to a cluster of machines, sometimes hundreds, to speed up compile.

While the general tools to do this are available in the GWT SDK, no one has made an easy to use frontend for them. Some of the Cloud vendors or IDE vendors who ship continuous build products could be helpful in this regard by offering easy to use management solutions for build clustering.

In general, giving more memory to the compiler (-Xmx4g), using SSD drives, faster CPUs, will speed up the compile.

We are working on several ideas right now to speed up the compiler for the next version of GWT.

Ray

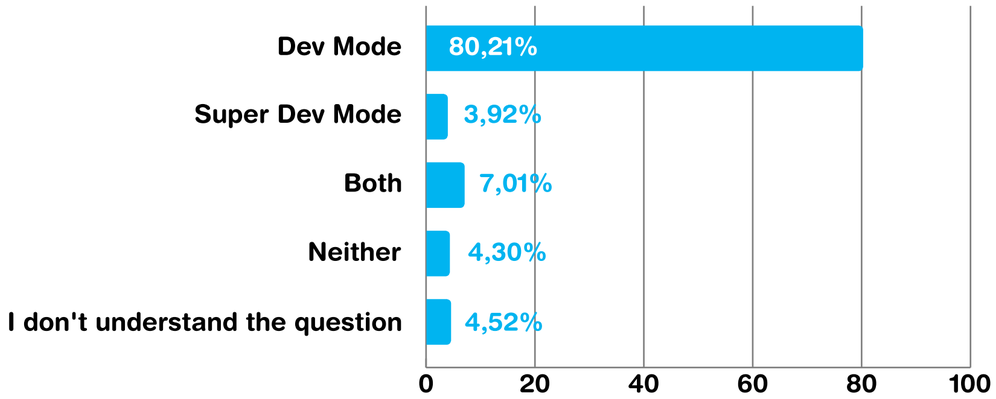


Compilation time correlation to size of the (uncompressed obfuscated) JavaScript

2.5 Are you a dev or a super dev?

The key to minimizing compilation time in the project is skipping compilation in the first place. GWT includes an amazing dev mode that allows running a GWT app on the JVM instead of compiling it to JavaScript. This should mostly eliminate compilations from the “make a change -see what it looks” -cycle.

With the newly released GWT 2.5, the next generation of dev mode is included. Naturally it is called super dev mode. Though it is still marked as experimental, over 10% of our respondents have already started using it and enjoy the benefits of dev mode without cumbersome browser plugins or browser-JVM bridge slowing the application down.



In which mode are you?

Updating the dev mode plugins has been a problem in the last two years since browser vendors started to iterate very fast with their browsers, breaking our development plugins almost every 6 weeks. With super devmode we are no longer dependent on browser plugins. There is a new emerging standard for cross compiled languages called “source maps” that will eventually be supported by every browser. It allows any language that is compiled to JavaScript to maintain a bidirectional mapping between the source language and javascript. For GWT this means that you can directly debug “Java Code” in Chrome. Going forward this should be the way to support every browser for development (including mobile) without any plugins.

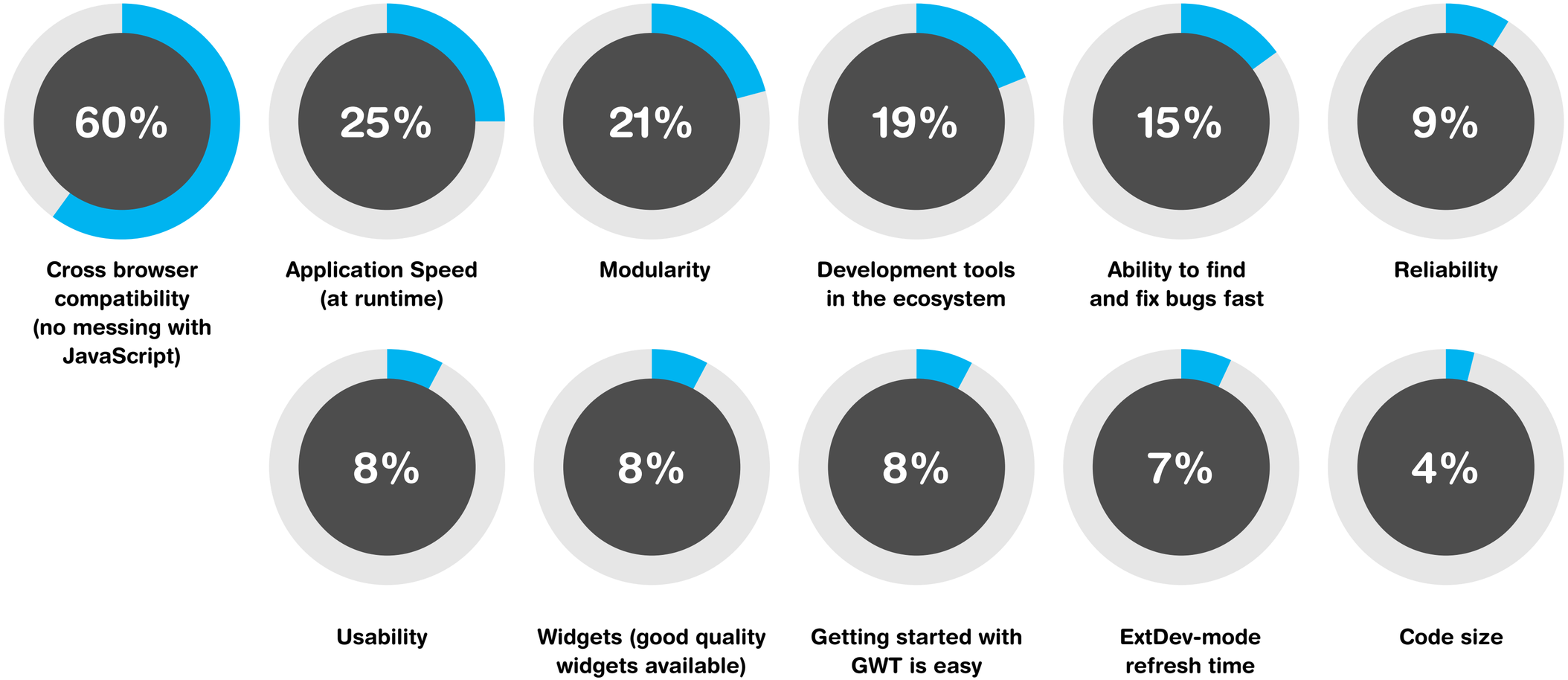
Daniel

When we started the SuperDevMode project, we knew we had to speed up compilation dramatically, but we had to be realistic about it. How much delay could a developer tolerate for a refresh without becoming annoyed? For smaller apps, 1s is achievable, but for medium sized apps it turns out 10 seconds was a good guess. This varies depending on the amount of GWT generators that run, the more UIBinder, I18N, GWT RPC, and others used, the slower. We are working on speeding up SuperDevMode even more to make it useful for all applications.

Ray

2.6 For the love of GWT

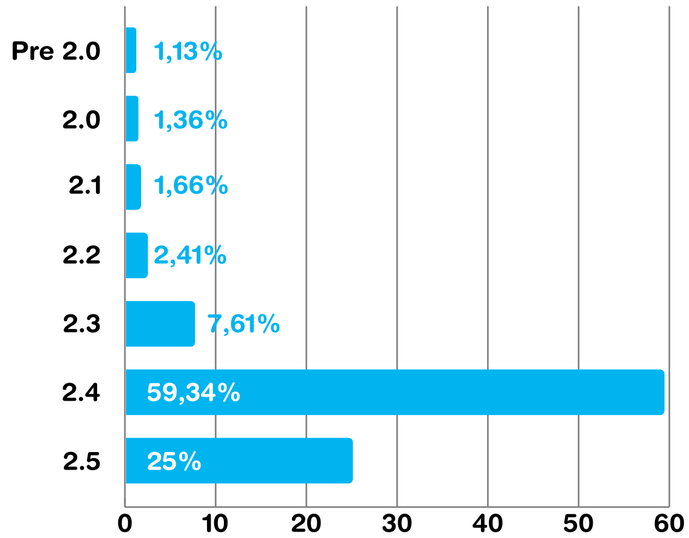
What are the strong points where GWT really shines? We asked everyone to name two of their favorite features. Although compilation with GWT takes time, the framework saves time by magically making your application cross browser compatible and well optimized without the need to fine-tune JavaScript by hand.



Best features of GWT

2.7 Latest is the greatest

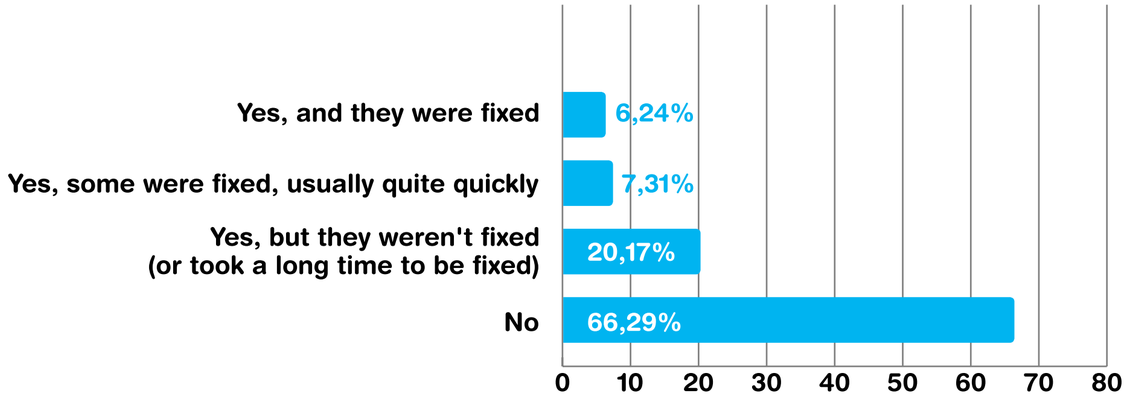
How up to date are most people when it comes to GWT? Well, as it turns out, most teams are using a version of GWT that is barely a year old and over 84% of respondents are using GWT 2.4 or newer. One fourth of the users have already upgraded to 2.5 that was released just a month ago on October 26th. That's a pretty good sign of a healthy, up-to-date community.



Version of GWT used today

2.8 Does it bug you?

When we asked if people have reported bugs in the past, it's not surprising to see that most folks haven't. Whether that means that they've never encountered a bug, or just haven't reported it is left open to your own interpretation, but just in case it's the latter, this link will take you to the best place to report bugs in GWT.



Have you reported bugs in GWT?

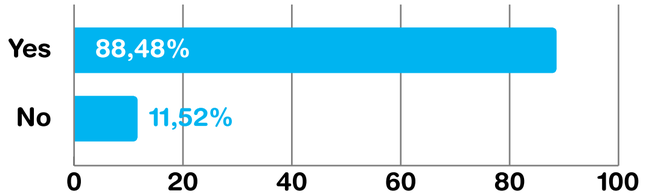
Section 3:

Where should GWT be heading?

3.1 GWT is doomed - or is it?

This is what we have been hearing from the doomsday prophets of the blogosphere since the dawn of Dart. While predicting the future is hard, dynamics of an open source project are somewhat more predictable. In most cases, if there is a strong developer community that wants to use some technology, it will - and the technology will thrive. Especially if the technology happens to be already established and widely in production use in huge business apps by top companies.

So we asked a pivotal question - “If you were going to start a new project that had a UI component, would you include GWT in it?” and heard blazing 89% of the people say hell yeah. Well, it was actually just a polite “Yes”, but with that percentage one is allowed to add superlatives, right?



Would you use GWT on your next project?

Google has large GWT apps, several of them exceeding 1 million lines of source code, many of them under active development. We have a keen interest in making sure GWT stays healthy. As part of the steering committee, we plan to continue to improve GWT and make it even better.

Bhaskar

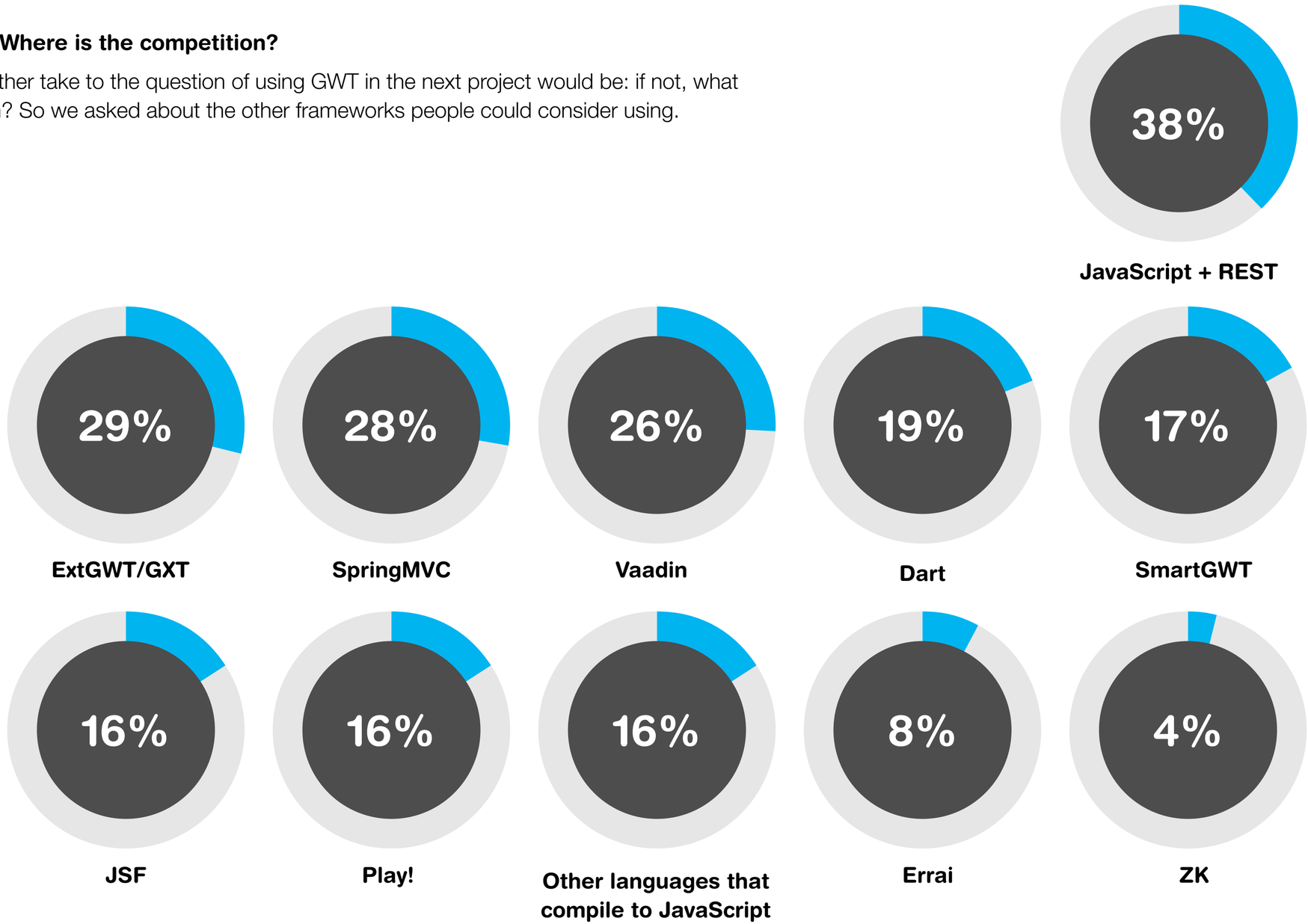
Vaadin Framework was born 12 years ago and has been using GWT for the past 5 years. We are so committed in GWT that we just merged it directly into the core of Vaadin 7.

Joonas

89% of respondents would use GWT on their next project

3.2 Where is the competition?

Another take to the question of using GWT in the next project would be: if not, what then? So we asked about the other frameworks people could consider using.



Errai is about bringing a unified programming model across the client and server. We think if you're going to use the same language, why not the same APIs?

Mike

Vaadin, GXT, SmartGWT and Errai can hardly be described as competition because they all build on top of GWT. While the others extend GWT, Vaadin also complements it with server-side development model.

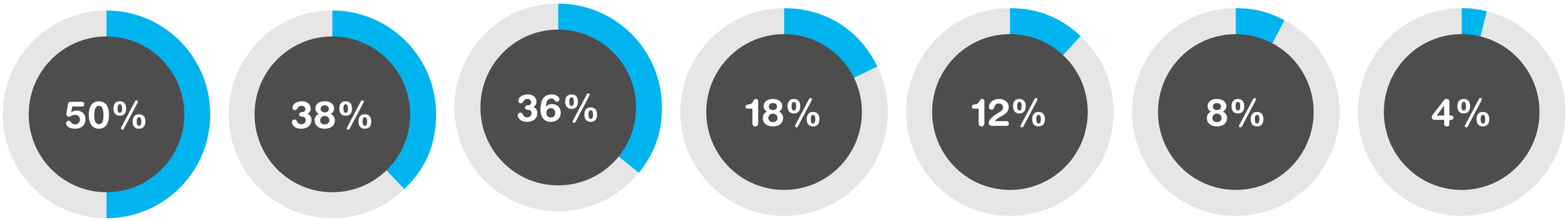
Joonas

3.3 Joining forces

In June this year, Google announced that it would open control over GWT to a steering committee, in hopes that it would blossom as Eclipse did after IBM relinquished control over the open source tools project.

We asked the community what they thought would happen to GWT now, under it's new stewards, and although a large group think that it's too early to tell, 36% think that GWT will get even better and 38% think that GWT will continue for a long time. Very few people think that GWT is dead (4%), or that GWT will die once Dart takes off (8%) -- so overall, the future looks bright for companies considering further use of Google Web Toolkit.

> I'm glad to see, as evidenced by this survey, that the GWT community is alive and well. But most importantly, that they have stuck with us through this intervening period. As GWT transitions into an even more open-source context, I hope we can breathe new vigour into the project and also provide easier ways for people in the community with time and ideas to help make GWT better.
>
> **Mike**



Too early to tell

GWT will continue for a long time

Once you've got 100k+ lines of code, you don't change overnight

GWT will get even better

The new stewards will succeed in making GWT prosper

Everything will continue as before

GWT will die once Dart takes off

GWT is already dead

Prediction on what happens now that a steering committee is beginning to lead the project

3.4 Your wishlist

In that light, we wanted to help guide future development, so we asked people to share their thoughts about the problems they are facing with GWT, missing features and extensions as well as their thoughts for the future killer features of GWT 3. And we got what we asked for - even more than we asked for! In fact, we got so many suggestions that it is virtually impossible to list them in this report on a page, two or even ten.

So instead of that, we decided to post the full wishlist of online.

Just go to vaadin.com/gwt/report-2012/wishlist to see it all, contribute your thoughts and vote the best ideas.

> I think its interesting that GWT's biggest strength (large, maintainable applications) turns out to be its biggest weakness (compile times). It seems to me that a big priority for the SC and the GWT community in general, should be the amelioration of the compile-time problem.
>
> **Mike**

> What I would like to see is in GWT is a CSS3 Parser, some more optimizations on CSS, reducing bolierplate with easier data binding, having great mobile support in gwt, more great looking widgets
>
> **Daniel**

> The most important missing feature in GWT, from my perspective in my selfish disposition as the Errai lead, is the lack of a good AST transformation API in the GWT generator framework.
>
> **Mike**

> GWT currently ignores the server side and how data is handled there, even though it is a feature of all applications. It would be great to see improvements in this part.
>
> **Artur**

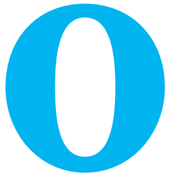
3.5 Browsers to support in 2012

IE 6/7

14%

Safari

18%



Opera

36%

IE 8

54%

IE 9

79%

IE 10

80%

Chrome

94%

Firefox

94%

Browsers developers expect to support in 2013

Conclusions

With over 20 pages of data, charts, stats, and commentary from some of the most well-respected folks in the world of GWT, we'd like to think that this report is the most complete survey of the GWT community to date. We've looked at everything from the composition of GWT teams to the worst features of GWT to predictions for the future but the one thing that stands out most about the Future of GWT is this: Now that the codebase for GWT is open for contributions, The Future of GWT is really in your hands, the hands of the community.

Here's how you can get involved:

Get your hands on the code at the GWT Repository
https://gwt.googlesource.com/gwt/

Follow the conversation of the Steering Committee
https://groups.google.com/forum/#!forum/gwt-steering

Report a Bug
http://code.google.com/p/google-web-toolkit/issues/list

Gerrit Voting
https://gwt-review.googlesource.com/

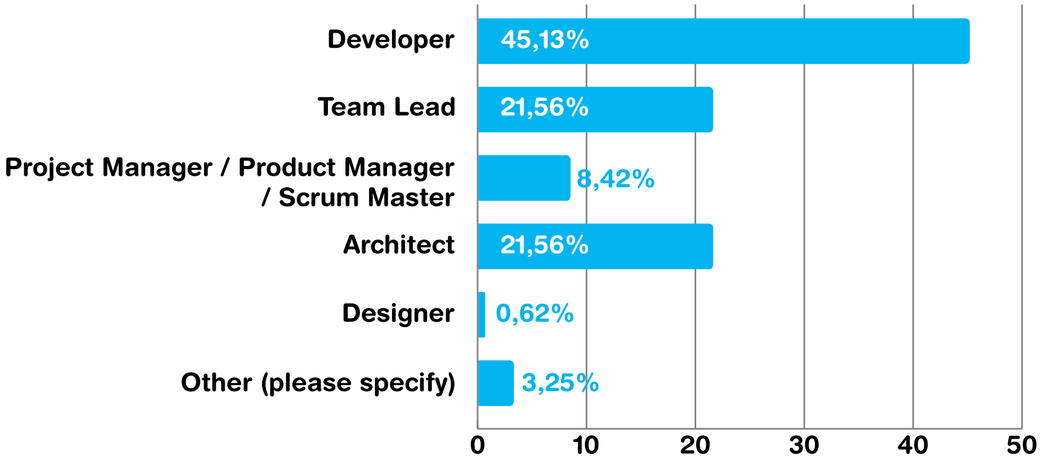
Share this report!
https://vaadin.com/gwt/report-2012

About the survey

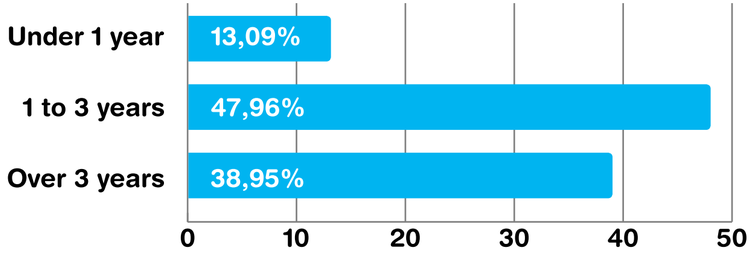
This survey has been created through the work of Vaadin, with the support and contributions of the GWT Steering Committee and GWT-minded individuals, most notably: Ray Cromwell (Google representative and acting GWT Steering Committee Chair), Artur Signell (VP of R&D, Vaadin Representative), Mike Brock (Project Lead, JBoss Errai, RedHat Representative), David Chandler (Developer Advocate, Google), Daniel Kurka (developer of mgwt, gwt-phonegap), Bhaskar Janakiraman (Google), Michael Vogt (Vaadin), David Booth, and Joonas Lehtinen (CEO, Vaadin).

We humbly recognize that there may be problems with some of our data, due to the sample size (n = 1349) and self-selecting nature of our respondents. None of our respondents were required to answer any question (no questions were mandatory), nor were they required to provide contact information. That being said, we'd like to graciously thank those who contributed their time and experiences into our 40+ question survey - and we look forward to creating a similar report next year, to track the changes that happen over the next 365 days.

If you're curious, here's a little info about our respondents:



Job roles of the respondents



Experience with GWT

vaadin }> thinking of U and I

Psssst!
Did you know that Vaadin offers commercial support and professional services for GWT?