rethink.fm Episode 13 with Gary Jones

Jackie D'Elia: Hey, everybody. It's Jackie D'Elia with another episode of Rethink.fm for you and today I have Gary Jones who is a UK resident and a Genesis expert in our community.

Hey, Gary.

Gary Jones: Hi. How are you doing, Jackie?

Jackie D'Elia: Very good. Thanks for joining me on the first episode for season 2, which is kind of awesome to have you here.

Gary Jones: Thank you very much.

Jackie D'Elia: For anyone who doesn't know who you are in the Genesis community, would you tell us a little bit about yourself?

Gary Jones: Absolutely. My name's Gary Jones. I'm based in Basingstoke in the UK. I run a small virtual agency creating technical WordPress solutions for clients, including integrations with their business critical systems. I'm a contributor to the Genesis Framework and WordPress core and many plugins and open source projects. I'm cohost of the UK Genesis podcast, although we haven't done many recordings recently, general translation editor for the British English locale in WordPress and organizer of WorkCamp London, father of five-year-old twins, and my background is teaching in schools and prisons.

Jackie D'Elia: Wow. That was awesome. You just went right through the whole thing. That is great.

Gary Jones: It helps being prepared ahead of time and actually writing my introduction down this time.

Jackie D'Elia: Yeah. If you're working in Genesis and you're in GitHub at all, it's hard not to see your name all around as a contributor for, like you said, numerous plugins, many open source projects. I see your name as a contributor on a lot of things.

Gary Jones: Yeah. A lot of the things that I contribute to are effectually under other people's names. It's under their repo or their GitHub name. One of the things I like doing is helping others. Irrespective of the context that might be helping clients or, in this case, helping other developers. If I can contribute something to their plugin then I'll go ahead and fork it and make the change and send it back to them. If it makes their plugin better and people recognize that there's a slightly better plugin or slightly better package that has got their name on it, that's fine by me. I'm actually fine with that.

Jackie D'Elia: I have to say, whenever I see a plugin if I need something in the GitHub repo and I see your name associated with it, I usually think it's okay to use because I kind of ...

Gary Jones: Probably, yes.

Jackie D'Elia: I know you're very thorough and you've got a really good background on one of the things we were chatting about before the shows is that you do some code audits and I've known that you've done that in the past for other developers, for businesses, for individuals.

That would be my first question I think I want to ask you is how did you kind of formalize an approach to do code audits? How did you evolve into that?

Gary Jones: I enjoy reading code. It's as simple as that. My background is teaching, in education and helping others, so if I could combine sitting there reading code and analyzing and look to see where it can be done better with somebody whose efforts, maybe it's for a free plugin, maybe it's for some client work, maybe it's for a premium product that they're trying to sell. If I could look through the code, make suggestions and educate that will use it as a tool to educate them so they become a better developer with the benefit that they have a slightly better product at the end of it, then I enjoy doing that.

I enjoy sitting down, sharing the screens for two hours and going through line by line, say "Have you thought about doing this? Have you thought about adding support for internationalization or making it more accessible or adding support for right to left scripts?" Or just reorganizing the code in such a way that it will reduce bugs and has a conflict driven approach that makes future edits even easier to do. Kind of easier maintenance.

There's lots of reasons for doing the code audit and it's just something that I found that I enjoyed. Word of mouth was that people would send other people to me to do that. Then the formalization process was very simple. It was, kind of, an hourly rate to a fixed price for a couple of hours of going through the product line by line and then at the end of it the person who we're doing the code audit for can take all those notes and implement whichever they agreed with in their own time.

It's not necessarily me just fixing up their code and them learning nothing. It really is effectively teaching them how to fish so that they can then know for next time, so when they do their second theme or an extension to their plugin, they've got that knowledge and everybody, WordPress users, WordPress ecosystem in general, is better off for it because they're plugin is now more accessible. Every plugin they do is more accessible, for instance.

It was one of my ways to contribute back and just brought a little bit of side income in as well.

Jackie D'Elia: Awesome.

How did you stay up to date with coding standards so that you know how to advise people about ways to do things and maybe refactoring and new ways to do things? How do you stay on top of all of that?

Gary Jones: In terms of the formal code standards, there is the WordPress Coding Standards. There's a document on the make.wordpress.org site that covers the actual coding standards. There's also a set of sniffs, they call them, for PHP CodeSniffer and I'm involved in that as well. So I've got [inaudible 00:05:38] access to that and involved in the view in all of the decisions that go through on that.

That's kind of the formal side of things about whether you should have commas after every line in an array and whether associative arrays should be multi line or whether certain functions should be escaped or not escaped or recognized and highlighted and so on. There's all of that sort of side of things.

Beyond that, it's just my own experience and just conveying that. If you've got a function call that accepts an array, instead of just dropping that array right into that function call. It just separates out the data from the logic, so that you perhaps, halfway through your development you can do something else with that array. You can extract it out into somewhere completely separate so that your function or your logic now will accept any array. So that when you come down to doing some unit testing, you can say, "well let's throw in a duff array that's got completely random data. Does our logic still work? Does it handle the edge cases and the errors and the exceptions and so on?" It's just my experience in getting that across and saying, "if you do this, then this is the benefit you'll get." To your end users, they won't notice any difference, but [00:07:03] you'll get some benefit from unit testing or by your editor being able to and therefore autosuggest how to auto complete some of the code. It's just experience that I've learned over the last 18 years or so that's conveying to them to make them better developers.

Jackie D'Elia: Okay, what code editor are you using currently?

Gary Jones: I use PHP Storm. For a long time I was ... What did I start off with? I think it was just basic Notepad to start with. Then I moved on to Sublime Text, Sublime Text 2 and then number 3. I love that. I like the fact that you could add in individual plugins or extensions to the Sublime Text system and you could have your code linter in there. You could auto-arrange, auto-align different things. It's fantastic. I still use Sublime Text for either very basic code editing or more likely for writing markdown documentation and so on. [inaudible 00:08:00] PHP Storm. I tried it once and it was too much. I couldn't get to grips with it. There was too much going on. Didn't like it. There's a free course by a gentleman called Jeffrey Way that's under the [inaudible 00:08:20] Podcast system things that he does. There's a free course which I recommend completely and totally for anybody who is even considering moving to PHP Storm is to get PHP Storm. You can get the free evaluation edition to run through this course. Pretty much the first session is: Turn this off, that off, and the other off and really simplify down to just an editor.

The actual power of PHP Storm ... Yes, you can integrate your version control and command line and all your other tools. Actually some of it comes from the auto-complete which you still get even if everything else is turned off. Running through that, when I found out the second time, then I've stuck with it ever since. I know that I don't use PHP Storm to its full potential. Every now and then I say, "well, is there a new feature?" Kind of re-map some of the key commands to Sublime Text so that I haven't got to learn two different sets of keyboard shortcuts. Between the two of them, everything gets written. PHP Storm is good in so much as there's an extension for it: EA Coding, I think it's called. That will highlight potentially bad practices with the PHP. Even if I've just written the same code it will highlight, "oh, did you mean to do this? Did you mean to do that? Have you thought about this?" It will highlight potential errors that Sublime Text wouldn't, and that I wouldn't have necessarily caught manually. It actually helps me to write code in several different ways.

Jackie D'Elia: Have you tried Atom?

Gary Jones: No, I've not. I've seen it around a lot. I've seen lots of praise for it. I might have loaded the website, but not gotten any further than that with it.

Jackie D'Elia: I've tried Atom as well as PHP Storm and I did go through the [inaudible 00:10:17] free courses, the videos on that and I found them very helpful in configuring and setting up Atom. I mean PHP Storm. I've also tried Atom because I've seen that used a lot now on Lynda courses and LinkedIn Learning courses that they're teaching. A lot of the developers that are instructors on there are using Atom. Now Atom's free, so that's a great choice for somebody who's looking for a free code editor and it does have the sniffer packages that you can install. I use that now in mine so that when I am writing code or when I save a specific PHP file or an SESS file it automatically formats it and lets me know if there's any coding errors in the file. Very similar to how PHP Storm works. That is another option for folks. PHP Storm does have a yearly cost to it, and Atom is another choice. I used Sublime Text as well earlier on and I found Sublime and Atom to be very similar. I think Atom is easier to install packages and manage that process than Sublime Text was. That's another option for folks if you wanted to do that. For some developers that are just starting off or are not what they should be doing coding in the WordPress space, you said that there's WordPress coding standards. I know there's a PHP sniffer that you can install that will monitor whether or not you're using the WordPress coding standards. Can you talk about that, and your involvement in it, and how would somebody go about implementing that?

Gary Jones: Yep, so there's a command line tool called PHP code sniffer. It's by a company called Squizz Labs, so it's an unusual name, but that makes it very easy to find.
[00:12:10] They support various different coding standards, the coding standards, PSR2, obviously WordPress, and so on. That's just the tool which comes with a few sniffs and a few checks. The WordPress coding standards is a separate repository that is [inaudible 00:12:27] in PHP so as a tool for checking your PHP, it's easy to see and understand how this tool is built. It will look through your files and it will break all the PHP code down into separate tokens. Could be a token of an open bracket followed by the word 'echo' would be a separate token. It's all broken down. Then it looks at combinations. Is there a bracket followed by a croat? Perhaps you've got an array or something like that. Also,
[00:13:04] hang on a minute, the WordPress coding standard says you need to space in
[00:13:11] between there. For a lot of these individuals checks, all of these different sniffs, for WordPress coding standards. They're trying to increase more is actually kind of automatic fixes as well. You could run this command line tool over your code and it'll say, "okay, you've got a thousand errors." It could then say, "well actually, I could fix 800 of these automatically." So you run a slightly different tool and there you go, it's fixed most of them. Give it a once over. Commit it to
[00:13:36] then go ahead and check out these 200 other errors. As well as individual white space errors or whether you're using short syntax or long syntax for arrays, it will also check things for WordPress coding standards. It'll check things like: are you escaping your output? It will say, "Well, you've got echo something." It'll say, "well hang on a minute, that's not escaped. I was expecting escape HTML in that context." Or you're doing in an attribute you're just doing 'get permalink'. Well it will say, "hang on a minute, I'm expecting that to be escaped." So there's lots of things there that will be able to help with the escaping and the sanitization and potentially validation. It's good coding practice. The WordPress coding standards are split into WordPress core, WordPress VIP which counts as being on the VIP with automatic, and WordPress Extra. The combination of core and extra means you not only get whatever is documented in that make.wordpress.org handbook, but the extra stuff is also things that we think you should be doing as best practices.

You can configure all of these to say, "well I want these checks, but I don't that checked." Or "Want it to be a bit stricter or a bit more lenient." So you can configure how these checks are done against your existing code base or how you want it for your new plugin or your new thing that you're doing. It's just very flexible, but at the same time there's a de facto, default behavior that pretty much all developers in the WordPress space should be following. As I said, you can add in your own exceptions if there's good reasons for that, but if you've got no existing code practices, coding standards for your company or at the company level. This is a very good way to make sure that the code you're writing can be readable and understandable and fixable without any obvious noticeable difference in code style between you and somebody else.

Jackie D'Elia: I think what you were talking about is maybe the PHP code beautifier and fixer? When I configured Atom, and I think it's also in PHP Storm too. It can automatically do a lot of these fixes for you. I did notice in Genesis sometimes with the escaping there's some Genesis functions that you might get a warning on and you might have to do some exceptions in your configuration file to say, "hey, it's okay. This has been escaped, but you just don't think it has."

You had some notes in a repo somewhere, I could have sworn I read that about. You had that same challenge. I was getting the same error and I was like, "Okay I don't understand this part. How am I going to fix this?" And then I saw that actually there was a way to put some exceptions in there. Say, "no, this is okay. You can ignore that one."

Gary Jones: [00:16:37] Yeah, the particular function is Genesis_attr for Genesis attributes. If you're doing a Genesis theme chances are you want that as an exception in your conflict file.

Jackie D'Elia: If you want to choose a base coding standard for WordPress, just the WordPress would be the one to choose if you were going to pick your HPCF, right? You'd want to go with just WordPress?

Gary Jones: For most things, because you'd have exceptions like that. You're conflict file would be in your plugin or in your theme. I can basically say, yeah, choose WordPress probably exclude the VIP. That would give you the WordPress Core, and the WordPress Extra and the WordPress docs, it's actually a separate rule set, but it all kind of falls under the same thing.

Jackie D'Elia: So, you could actually configure like, three rule sets and then just exclude the VIP. Then you could also do some exceptions of your own in that file? I think it's like a ... Is it an XML file if I'm just ...?

Gary Jones: Yeah, PHPCS.xml. , and then probably if it's a plugin or theme that you're distributing you'd put .dist on the end. The advantage of that one is that, that is the conflict file that you distribute, but if somebody wants to override it locally they can have PHPCS.XML which you would put in your 'get ignored' so that anything you've got isn't distributed as well.

That would then just override the conflict file that you distributed, so you'd actually get the best of both worlds, distributed and local configuration.

Jackie D'Elia: So, that's a great starter if you are not using any coding standards, and you're just writing PHP code in WordPress. It will work with all of these code editors you can just find the package that goes with the code editor that you're using. You install it. You can immediately ... You can either run it from the command line, you can have it go through all your PHP files at one time and make all of these corrections and give you a notification of all the problems and errors. I do it in mine. Every time I edit and save a file, it just automatically does the PHP CBF Beautify. So, it reorganizes the file so that it corrects for most of the things. Then anything that's left is going to show up in the bottom of my window. It says, "hey I found six errors." It underlines them and shows you where they are.

Gary Jones: Yeah, and you can also tie into . . If you're doing any kind of continuous integration, so if you're using Travis or Scrutinizer or Bit Bucket Pipeline which is what I've been using a lot at the moment. You can set those up so that it will say, "hey pull down the WordPress coding standards, set it up, and run PHPCS

[00:19:11] against your repo and your newest . Then you'll get a parcel file that says, "hey this is being checked." Instead of running it locally, you you can it on every commit as well.

Jackie D'Elia: Oh, that's very good. All right.

So, we've kind of covered the things that I wanted to ask you about as far as coding standards and code editors and things like that. Just want to roll back to Genesis in general.

I know you're involved in the Genesis Slack channel. That's where a lot of conversations go if you're a Genesis developer and you haven't found that, or you're using Genesis. It's a great resource to join the Slack channel.

Do you still use Genesis on most of your projects that you do for client work?

Gary Jones: Yep, absolutely, yeah. If there's a theme involved we will push them to use Genesis even if it's a custom child theme that we build from scratch or adapt an

[00:20:05] existing one or theme out there. Yes, it'll always be Genesis. That's just what I'm familiar with and what my team are familiar with.

Jackie D'Elia: Why do you like Genesis, in general over maybe some other frameworks if you really want to call Genesis a framework or just a parent theme? I'm not really sure. You have a couple of options. Why did you choose Genesis and why do you continue to use it?

Gary Jones: I originally stepped into Genesis, before that I was using a different parent/ stroke framework, and I try to contribute back. The lead developer there just wasn't interested at all. It was like, "okay, well yeah. Your theme is okay. I think things could be done better, but you're not willing to accept." I got introduced to Brian Gardner via email and he made the fatal mistake of saying, " oh, if there's anything you think could be improved, let me know." For those who know my reputation, you'll know that I'm not afraid to say, "yeah, there's certain things you could do better." So I sent him a big email saying, "yeah there's certain things you could do better." He was like, "wow, okay. We need to get you involved here somehow."

That's how I stepped into it. From there just making contributions to the Genesis Core. There's certain things and some areas like the breadcrumb stuff is stuff that I completely rewrote. It's been adapted and updated and probably improved and fixed since then. There's certain areas that I know that predominantly were my original contributions, and that's great.

Beyond that, for a few reasons, familiarity with the code. Its the ability to still be allowed to contribute even if I know that some of the contributions I make, not necessarily wind people up, but are trying to push Genesis in the direction that others might feel that it doesn't need to go. My whole goal is to bring Genesis kicking and screaming into a modern development workflow that everybody can be proud of to use on the development of Genesis side of things. Beyond that, it's then the community, there's fantastic resources, as a tool itself, if I'm doing multiple websites then I've only got one code base to code against. If we've got clients who've got multiple websites they've only got
[00:22:38] one tool to learn, because obviously they don't interact too much with the actual theme itself just the features that Genesis has got. Lots of plugins for it. Its a whole community in itself. A community of people, and tools and support and tutorials. I enjoy that community. I can contribute to it in different ways. I like the whole setup of everything there. I've got no plans to move away from it.

Jackie D'Elia: Do you find it's easier to customize or develop a child theme in Genesis versus in WordPress Core, just do a child theme there? I'm asking that because I watch a lot of videos for instruction, especially on Lynda. Many times I'll see Morton Rand Hendricksen, he'll do a video and he's not using Genesis. He's doing things like, just for a typical child theme in WordPress. To me it looks sometimes more complex. A little bit more difficult to work in there. The beauty that I think in Genesis is the hooks and the filters make it very easy to intervene, make your changes, and produce a result in a very logical fashion versus having to go in and edit a bunch of individual files. Then you gotta remember where all of those edits are. To me the beauty of the way the Genesis is structures is, even if I create a lot of partial PHP files for my ... Break apart all of my code specific functions like all of my theme setup can go in one file. All of my widget setups can go in another file. My default for the themes can go in another one. Then I've got another PHP script for handling all of loading of all of the assets, whether they're the JS or anything else. Your fonts, everything can be all in one place.

The beauty of that is that those are more like configuration files to me. It just seems to much easier to understand to be able to adapt ad quickly change things versus having to go in and like edit your header.php file and your footer.php file. There's lots of people who do it that way and love that way that that works. I just scratch my head a lot of times. I don't understand why you would want to do it that way versus this nice configuration that's already available to use.

Gary Jones: No, I completely agree. The use of the action hooks and filters within Genesis means that if you can write a plugin, but you've never touched a theme before, well you can edit and customize a Genesis child theme. It's exactly the same. If you understand there's certain hook points and there's a callback and there's a priority for that and maybe some passed in, you understand all you need to know. Combine that with the conditional function so it's home is "front page" or is front, is archive, is single and all those. You can configure everything you want and as you said, it allows much more flexibility in terms of splitting up the child theme into an array or several files worth of array configurations plus some logic that handle those so that they can be class files, plus then some views that has got your html with a php values, escapes and echoes into it. For me, in my mindset, that works far better than having templates or partial templates that are mixture of some logic and some data and the views side of things. Yeah, I can't see the benefit of learning something new when actually I'm writing enough plugin themes that I understand the hooks and filters and can make use of it within a child theme as well. I realize it's different to how many of the child themes work out there, but I love that level of abstractness. That works for me and I can build on top of that in a way that partial templates and some of the other features of 2017 and 2016 and the like don't have.

[00:25:27]

Jackie D'Elia: Well, we covered what we like about Genesis, so next question would be: What would you like to see changed and you briefly touched on the direction you'd like to see Genesis move into. What's your vision for the future for that? What would you like to see?

Gary Jones: Probably three things that I'd like generally addressed in no particular order. First would be, code standards. There's aspects which we're talking about the code standards that Genesis still has issues with. Genesis 2.5 will be much better than Genesis 2.4. there's still room for improvement in there. Its not necessarily has it's own coding standards so much as there's aspects of it that just doesn't follow the WordPress coding standards. When you run the WordPress coding standards sniffs against Genesis you might get hundreds of errors. If we could clear out most of those by adhering to the WordPress coding standards, I will show up with things like, "oh certain values aren't escaped." Or certain things you're using, the post data or getdata without actually sanitizing it for instance. It will show up the more important, critical ones if we can get rid of the fluff coding standards reports that are coming in.

Second one would be more config and class driven so that at the moment, much of Genesis is procedural code. There's all functions in there. There's a new thing within Genesis 2.5. I'm not sure if we can say about it yet, but basically its split up into a config and a couple of classes so that if, as, and when the config values, the data needs to change, that can be done without even touching the logic for that within the actual classes.

That then leads nicely onto the third item which would be to have more unit tests. Yes, you can test functional code, but with classes, when you're just dealing with that logic you can say, "right, if I give you a certain value into your class method, does it produce the right result that I'm looking for?" You can test
[00:29:03] all the different code branches, well if you give it duff data, do you get an exception, and so on. So by splitting that the procedural code out into that config and that class you allow it to be more unit testable. Genesis does have unit tests. Its not distributed so most people probably won't know that they're out there. There's certainly a low percentage of code coverage at the moment of which we're trying to do those kind of two, not the code status as much but do that second and that third item on the agenda so that we do get a high level of code coverage. We can be more confident that when we write some, re-write some code so that the procedural codes are just wrappers for the class-based code. We've got the unit test to say, "actually yeah, you're still producing exactly the same behavior." Or, "no, you're meant to be returning an array, but now you're returning an array of integers instead of strings" or whatever it might be.

So, certainly that's where I'm trying to push Genesis at the moment. It's to just take little bits of pieces. Let's take the Genesis update. At the moment, the Genesis updates ... If you were to update from Genesis 1, probably not that far back, but say Genesis 1.0 the code would end up running through a series of functions that do version based updates and then one function that does a whole load of updates for particular versions all in one go. We could extract that into a series of class, so we could have Genesis 2.3 update, Genesis 2.2 update, Genesis 2.1 update and just kickstart the method in each of that so that we could then test that in unit tests, do integration tests for that individual update to say, "if we run this, does our database now show it's got an extra value in it or not an extra value?" Or whatever the change to the database might be. We'd be able to test that far easier than what we've got at the moment which predominantly comes down to manual testing. The more we can move to unit tests and integration tests with Genesis the far happier and confident everyone will be about how things are working as they should be working.

Jackie D'Elia: How about code base in Genesis? The XHTML code in there. Do you see any need to maybe remove that at some point? Or separate the versions where you leave the XHTML code and maybe just do security updates for it? At some point, let's just say you've got HTML5 and then you have the next standard that comes out after that. How long do you think its feasible to leave all of that code in there to support all of these different versions?

Gary Jones: [00:31:54] I think predominantly that's going to come down to a business decision. That's well outside my remit. I don't work for StudioPress or Copyblocker or Digital. So that's for Brian and Nathan to decide ultimately. Much of the markups only for 2.4/2.5 goes through what we call the Genesis markup API. It's not an API in the strict sense, but basically any time there's markup to be output it goes through a particular function which accepts what the markup should be whether it's in the HTML5 mode or the XHTML mode. That's the way they've gone forward in that. I proposed a different scenario where as we try to move more things to views, which Genesis 2.5 will have a views folder and the actual markup in there where possible. I suggested, let's have HTML5 views and XHTML views directory. Then basically on the earliest point you work out, are we supporting one mode or the other and just set the directory path to whichever one was relevant. That way then you get a whole load of files in a directory that basically aren't being used. It's not loading them. It's not affecting performance and so on.

Whereas at the moment it is a bit of a mixture. At some point in the future, they'll say, "right, at version 3 or in 2018 or at some point that they feel is a good business decision. Yes, we're now going to take it all out and simplify it all down so we just have it in the HTML5 markup in there.

Jackie D'Elia: What have you been rethinking in your business and how you work this year versus last year or the year before. How is that affecting what you're planning to do?

Gary Jones: Over the last couple years I've been in an odd situation. I've got one long term client who just keeps giving more work and more work and more work and they are fantastic clients in every which way. That's kind of led me to the odd situation where I've basically done no marketing. What I'm trying to push for this year is to effectively sort out my websites, sort out the positioning that I've got which is something that I've never really established. I've just been a generalist. To sort out the market in terms of what my reputation is with the Genesis community or with the WordPress community with the developer community shall we say. Which isn't going to be advantageous or useful when I'm trying to market to business with large budget who need a partner to work with them on creating a new website or creating new features or whatever they might need. Part of that is working out my positioning and working out I can convey that within the content of my website. Just the whole brand, really. Which for a company who is going to be 9 years old is quite an unusual step. I've never had to dive into it as much as I've done so far. I'm trying to balance between the code, the individual projects that I'm coding at the moment, plus project management between the couple of main clients that I've got and my team. And sorting out this marketing so that I can move forward in the long term and get more leads in of the type of business that I'm after. Potentially then take on a bigger team to handle that work and grow the company that way. The rethinking is all about establishing the business and not taking the fact into account that I'm 8 years old already. How can I grow? Where do I want to grow? Which direction do I want to grow? Putting myself out there. Ultimately, I think it comes down to 'monetizing my knowledge' is the phrase that I keep saying to myself. I've spent hours and hours and hours learning about the code and establishing a reputation, good, bad, or otherwise within certain communities. It's, "how can I turn that now into money in my pocket from the businesses who need that skillset, or who need that experience and that advice and that guidance?" It's all about the business now. I've definitely changed phases in the last six months, I'd say, from ...The thing that kick started it is my main client had their business arrangement was such that they were potentially being taken over and said, "well for quarter 1 of 2017 we might need to cut down on the amount of work that we can send your say. Just for three months, just so that our books look better. So that when we're taken over, we can get a higher share price and higher dividend price and so on." Because of that and obviously, not as heavily supporting the people in my team as such, but wanting to make sure there's sufficient work to send their way and support me as well. I really need some extra leads. I can't have all my eggs in one basket. We're starting to get more leads in. Starting to diversify. To do that properly we need proper marketing and to really set the business on the right path to growing in a way that it really should be growing.

Jackie D'Elia: Any plans to have some more episodes on the UK Genesis Podcast? I loved listening to it. I must listen to every episode that should you guys recorded. I miss it.

Gary Jones: Yeah, yeah we did, I think up to 25 episodes. We did quite a few. We recorded them every two weeks or so. The reason why we stopped is that we effectively dried up of the people who I knew were willing to come on as UK based Genesis users/developers/designers/Genesis professionals. We got to a stage where we dried up and just that's it for now. Now we've left it probably a year and a half, something like that. It's been awhile since we've done one. Obviously we do know more people in the Genesis UK community so it's just a case of finding some time. Joe and I have only recently finished the Wordcamp London. Having [inaudible 00:38:13] organized last year. I was the organizer last year and I was the organizer this year. So now that's done, maybe there's some more free time amongst all that project management and marketing and all the other bits and pieces.

it would be great to do that. I think it provides value to those ... The reason why we originally started it was that I was getting leads and Joe was getting leads saying, "right. I'll use Genesis", or "I want to use Genesis, but I want to develop for in the UK." Timezone considerations and so on. It's like, "well we know a few, but we don't know that many. There must be a community out there." So we set up the UK Genesis community, as it were. We wanted to have a directory of contact details so we could say, "hey go look at this. There's a whole load over there. Go and make contact with each of these people who you think might be suitable." Work on your business that way. Ultimately, we ended up just having the podcast which were hour long interviews with the individuals and just find out about their business, how they work, and again the tools they use. Then some quickfire questions which I think was really useful for showcasing them. Those 25 hours do act like a showcase that they could use or direct people towards. We even had Mike Little on there. We know he's a Genesis user. He's a co-founder of WordPress based in the UK. So as a Genesis user we very fortunately managed to get him on for an hour. I'd love to do it again. I do enjoy the podcast. It's just finding the time and finding what value it can give to me. I contribute to lots of different things. There's usually an ulterior motive of to why I'm contributing to a certain plugin. Its because, "oh, I'm using it myself and I need a bug fixed". Or, "I'm using Genesis.", or "I want Genesis improved." Or same for WordPress Core. Its for the UK Genesis podcast, it's "okay, what value is it giving to me and my business in terms of where I want to push it?" If that all matches up or if I can find the reason there to join it up then, yeah, absolutely we'll start contacting some UK people and say, "right, let's set a date. You come on and do a live show for an hour. Talk about yourself." Which most people can usually do without too much practice beforehand. Let's get you an hour on there.

Jackie D'Elia: Is there anything else you wanted to share before we wrap up?

Gary Jones: No, I think we're good. For me, everything is rethinking at the moment for my business and moving forward. It's good to be busy. There's lots going on. I probably need to cut down on some things, but yeah. I think everyone just needs to keep going exactly what they're doing. If they find it's not working, change it in some way and move forwards.

Jackie D'Elia: All right, well thank you Gary for joining me on the show. I hope everybody got a lot out of it. I know I did. If folks want to reach out to you and follow you or touch base with you, how can they reach you?

Gary Jones: I'm on several different Slacks. The Genesis Slack and the WordPress Slack. I'm on there as GaryJ. I'm also on the UK WordPress Slack under the same name as well. Main website would be gamajo.com, so either look for gamajo or GaryJ or Gary Jones on most profile sites. Twitter as well is GaryJ.

Jackie D'Elia: All right, well thank you again, Gary. Thanks to everybody else for listening and tune in to the next episode.