PROGRESSIVE SESSIONS

Jonni Bidwell talks progressive web apps with Enonic's co-founder **Thomas Sigdestad**

Thomas Sigdestad is CTO and co-founder of Enonic, a company that's grown from humble beginnings in a garage in downtown Oslo to one of Norway's most successful open source companies. It had the bold idea of providing people with something useful and usable during the dotcom boom, and that remains very much part of its strategy today. We caught up with him at the O'Reilly Software Architecture 2017 conference in London to talk about the future of cross platform applications: progressive web apps. You may not have heard of them, but the chances are, if you've gone anywhere near a major website recently, that you've already used one.

New technologies enable web browsers to do all kinds of things – things that hitherto were strictly the preserve of native applications. This could mean many exciting developments: truly cross-platform applications, an end to the scourge of app stores, perhaps even Linux becoming a viable option for people bound by application requirements on proprietary OSes. A brave new world awaits, so read on...

Thomas co-launched Enomic back in 2000.

Linux Format: Hi...

Thomas Sigdestad: So, you're the same people that made *Amiga Format*? I still remember looking forward to the disks and the first review of the latest game. It was pretty expensive in Norway though – £7 or £8 I guess.

LXF: That's us... well, not us personally, but our company. I loved that mag though. My mum used to go mental that I'd spend £3 on a magazine. She didn't understand the amazing value that represented. It wasn't just the quality journalism – the cover disk was a big deal: 880K of PD games and demos that would keep me occupied for, well four weeks. Ahh, nostalgia, best not get me started. How did you get into Linux?

TS: When I was studying I bought a Red Hat box set from the university bookshop. I spent a lot of time investigating that. I remember thinking what was really cool, and what I still think is really cool, is network booting. You just start the machine, and it goes off and fetches an image. This was amazing in 1997.

LXF: I used Red Hat in the early 2000s. As soon as I had a computer capable of running Windows XP, I discovered that I didn't like it. Not one bit. I'm less militant now, but Linux has been my primary OS ever since then.

TS: That's funny. Our product is called Enonic XP. The XP is short for eXperience Platform, but it's nothing to do with Windows XP. When we released it two years ago we referred to it as a "web operating system." If you're building an Android app, then you know how to write to disk and how the graphics work and things like that. There aren't any other dependencies, just an Android version. We had the same idea for servers: when you're building an Enonic app, you only had to specify which version of XP you were using. So this is an operating system concept at a much higher level – a server distributed approach. We went with the XP name for marketing reasons, but maybe in a few years "web operating systems" will catch on.

LXF: Tell me about your company, Enonic.

TS: We started back in 2000. We wanted to be a software company, so we worked for a year making our web portal, and then we launched and surprisingly quickly ended up with some interesting customers. We've used Linux for various purposes and in various guises from day one. We've always used Linux for our hosting.

What we did that made us unique was that we built everything using Java. We had an idea – sort of inspired by Bill Gates – that Microsoft's success came from piggybacking off IBM. We more or less did the same thing with our application server and database. We basically got customers that had invested in those architectures who wanted our solutions. So stone by stone we built our business over five years. We made a neat CMS, and saw that things were changing in many ways. We had application teams working on one stack, database teams working on another stack and website teams working on yet another stack. But they were all ultimately working to create some user experience and would have to wire these stacks together somehow. We then realised that this can be done differently.

When you invest in a CMS you have to have a full stack, including the database and everything else. So we built everything from the ground up, in a single piece of

IF IT WORKED FOR MICROSOFT...

"We had an idea – inspired by Bill Gates – that Microsoft's success came from piggybacking off IBM. We more or less did the same thing with our application server and database."

»

software. And the CMS is the surface layer of that stack, so now we're an application platform with a CMS as an optional feature of the stack. But typically that's what our customers use. Even though we could make many interesting solutions, we have chosen to focus on three key areas: digital experiences, progressive web apps and back-end services.

LXF: Let's talk about progressive web apps. Forgive my ignorance, but what exactly are they?
TS: The short version is they're web apps where you get the experience of a native app, but built with web technology. You can see an example at **http://officeleague.rocks**. It's a foosball app that enables anyone to start their own league. It'll work on any modern browser, desktop or mobile. Everything is driven by JavaScript, so there aren't conventional page reloads which download the application code every time. You get an Elo rating like in chess, and you can watch live games, which is pretty cool.

What's also cool is that if, for example, I'm playing on my phone and I turn on airplane mode, then the web page detects that I'm offline, but I can still play a game. The app records it, and the moment I go back online the app sends the data back to the server. Beyond this, the first time you run it you'll be prompted if you want to add it to your device. If you agree, you'll get an icon on the launcher, just like a regular application. And just like a regular app, you can look at its information in Android's app settings, and uninstall it. You have access to lots of different hardware features that you wouldn't usually expect with a web app.

So a progressive web app isn't a particular piece of technology, it's more a pattern of what these apps can do and how they work. To summarise: they feel like native apps, they're cross platform so you only need to code it once, rather than building for Android, iOS, web and Windows separately. They're also responsive, so you can have different screen sizes, and they have offline support, so there's resilience as regards network conditions. The reason they're called progressive is that the core parts of this foosball app, for example, work on an iPhone, even if they don't support the offline mode or some other features. So you can have progressive enhancements of your app depending on the capabilities of the client.

LXF: What about video acceleration? Do they use webGL or other trickery to give a smooth experience?
TS: Yeah, they can basically use whatever technology your browser is capable of on the device and platform you're using. It's up to the developer whether or not that's used, though. If part of making that app is that you want hardcore performance, then you might use WebGL or maybe even WebAssembly. That's just one use case though, and not everyone cares about that. Most business cases only require standard web technologies to create apps.

LXF: How easy is it to write these things?
TS: It's fairly simple to use as a developer, and we believe that two years from now native app development will become a niche thing, only used when there are specific requirements. For example, Google just announced the WebUSB and Payment Request APIs for Chrome (see **https://developer.chrome.com/apps/app_usb and https://developers.google.com/web/fundamentals/payments/**). These are pretty awesome. For example, if you're PayPal, then you can create a web or native app and register as a payment provider in a given device. Then a website can just call the payment request API, and the user will be given a list of payment providers – PayPal, Apple Pay, or whatever – so this removes much of the complexity in carrying out the actual transaction.

Likewise WebUSB opens other new capabilities—so everything from controlling the camera (every aspect of it), the microphone, access permissions, all of these are embedded in the web now. The web was always involved behind the scenes, but now it's come to the forefront, it's absolutely a viable alternative to native applications.

LXF: App stores have become common the standard way of getting apps. Even Gnome's Software tool pretty much conforms to this norm. How do these work for progressive web apps?
TS: Well, when you visit a web page there isn't an app store involved, so the whole "purchase/download from Google Play", or whatever step is skipped. So that's good for developers and good for users, because it speeds things up. Twitter is probably the best example of a progressive web app. It's three per cent of the size of the native Android app. There are no updates: you're always running the latest version whenever you visit the website.

So this way you get the best of the traditional web app world, combined with the best of the native app world. It's just a question of getting web browsers to support the necessary features. This initiative is being pushed hardest by Google through *Chrome* and *Chromium*.

» HOW ENONIC CAN SAVE YOU TIME…

LXF: So you started Enonic in the dotcom crash, right? But you had a good idea, unlike a lot of companies who were just putting the word "Internet" into wonkybusiness plans.
TS: Yup, we started in summer 2000 and one month later the whole world fell down. We went straight into platforming, which isn't the easiest thing to do. In a way we were lucky, because we were thinking, "Let's make things dead simple to use", so we wanted to make a CMS so simple that anyone can use it.

We had to give users form-based input, and to present this requires quite extensive coding. It turns out that no small company wanted to invest in this level of coding, but the big companies wanted this, and since we were already involved with Java we were a nice match for them. We were pretty young when we started, but we made a product that we thought was reasonable, but at the time we didn't really know who needed what we were building. We have a better idea of that now, but still things are changing very fast, especially with the cloud.

These days our offering is essentially "You have an idea. We have a platform. Run it in any cloud if you build on our platform". For typical cases we replace SQL/NoSQL, search, app engine, and at least parts of the web server. We have an identity system, you can plug in others, and we have our CMS. So we offer maybe five systems in one. Traditionally, people have been used to putting all that together first, before you write a single line of code. And setting that up, getting all those components to talk to one another, can take a lot of time. If you're happy with how we've set them up, then you can save all that time. And you don't have to worry about maintaining and updating and scaling all of those components, either.

Thomas explains that progressive web apps feel like native apps.

Firefox is doing a really good job too (see **https://developer.mozilla.org/en-US/Apps/Progressive**). Microsoft is involved too, so it's another chance for it to become relevant on the client side. The company's going to have progressive web apps on the Windows Store.

LXF: **What about Linux? Presumably progressive web apps can run there as well as anywhere else?**
TS: They do, and that's an interesting point. Linux dominates virtually every other field – the cloud, mobile (if we permit Android to be Linux) – but it never made it to the mainstream desktop. But with progressive web apps, desktop Linux could be a viable alternative to macOS and Windows.

LXF: **It's a bit like the browser is becoming the new operating system then?**
TS: Yes, at least if you think about it as the runtime for applications. They've become the foundation for client applications. It's a little like what Java did for coding server side applications in the late 90s.

LXF: **It's funny – back in the day whenever I saw "Starting Java..." in the status bar of *Netscape* it sent a shiver down my spine. I knew it was likely my machine was going to be brought to its knees while whatever applet was loaded. But that was on the client side, and probably my 486 was underpowered even for those times. On the server side Java enabled all kinds of things to be bigger and faster. I've heard this is why Twitter was able to scale like it did.**
TS: Yes, you could say that, most of the huge social media sites are running Java now. Java alone is good because it helps you with multithreading and stuff like that. But the community on top of Java, and in particularly the Linux side of that, creates all these cool open source things that make it even better. Twitter had some teething problems initially when it started to get big, but then it started using some of these things to improve its search, for example, and this really helped them to scale out. But yes, Java is pretty ubiquitous – our platform is based on Java. We've seen a new trend though, one related to progressive web apps and the web in general: JavaScript. When you build apps on our platform, you code with JavaScript, both server side and client side.

STAYING UP TO DATE IS EASY...

"Twitter is the best example of a progressive web app. It's three per cent of the size of the native Android app. There are no updates: you're always running the latest version..."

LXF: **That's an interesting point. A lot of people aren't familiar with server-side JavaScript, I'm used to clumsily dumping JS code in an HTML file, for example, but**

»

things like Node.js confuse me. Can you explain how that works?
TS: Essentially, JavaScript was created by Netscape to do some fancy tricks in the browser. But when you run JS in the browser, it's a single UI thread, and typically you also have this DOM (document object model) that you manipulate. So that's typically what you see: there's an event, some JS kicks in, it reacts and changes something in the DOM.

On the server side there is no DOM, so then there's a few different approaches on the server side. There's the language itself and then there's a huge movement around **Node.js**, which works pretty much the same as in the UI: it's a single thread, so you end up starting lots of small servers and you have to route the traffic into them, which is fine, and it's also stateless.

It has this same even principle: something comes in, an event is kicked off, and then you can do whatever asynchronous trickery you want between server and browser. However, we wanted people to be able to use all the investments that had gone into Java, all the multithreading, and the ability to fully utilise the hardware. In addition, multithreaded code is easier than async, it's simpler to debug and things like that. So we enable you to code server-side JavaScript: you get an initial request, a piece of your code is invoked, and then the JS itself is run.

Then there's also an effort called commonJS, which lets you do includes, basically. Standard JS doesn't support that. These concepts are the same for the client and the server, but in node.js you have just this single thread. Our platform has a threaded model. The idea is the same as with any scripting language though, you kick off a script and start doing stuff programatically. And you can take advantage of all the libraries or utilities out there to help you do stuff.

There's another nice feature too: When you're coding client side stuff, in *Chrome* you can go to Developer Options and find a debug console which makes life easier. With server-side development, you're often at the mercy of the client, so debugging is harder. But on our platform, you can use that same debug console to follow the code into the server, which is a nice feature for developers. The Javascript framework we use in Enonic XP is also available as a standalone project called PurpleJS (**http://purplejs.io**).

LXF: What's the tech community like in Norway?
TS: Oh it's really big. One of the biggest developer conferences, JavaZone (**https://javazone.no**), is held in Oslo. I think there were 2,500 developers there this year. One thing that's a bit sad though in Norway, is that 90 per cent of what's going on [in the tech world] is consultancy. The government and big companies hire people through consultancies. So they pick up all the talent, and the culture for product development isn't that strong. Nowadays there's more of a push for people to get involved with startups and such, but there's still too much of a focus on consultancy.

But some important companies have come out of Norway: Fast Search & Transfer (FAST, now part of Microsoft), Qt, obviously a big one for Linux, Trolltech (bought by Nokia). So in Norway at least, the successful companies tend to get bought up. So maybe Norwegian investors haven't quite realised the potential of digital. But there's definitely lots of skilled people – there's no shortage of talent.

WHY NORWAY'S THE PLACE TO BE...

"The tech community in Norway is really big. One of the biggest developer conferences, JavaZone, is held in Oslo. I think there were 2,500 developers there this year"

LXF: Tell us about your talk. I missed it because I have to spend all my time talking to people smarter than me.
TS: Well, I wrapped progressive web apps into a Star Wars-themed discussion. Apple was the dark side. It's good to make these things at least a bit entertaining. The message is that we're trying to build a platform that helps people create powerful web apps. We think progressive web apps are going to be massive, but most people aren't aware of that yet. It's only a matter of time before things kick off. And definitely for native Linux distros, this is going to be really great.

LXF: Awesome, well thanks so much for your time. I think James our photographer is now going to orchestrate a hero shot out in the lobby
TS: "Hero Shot?!" Oh no, I didn't bring my latex outfit! LXF

Thomas, seen here without his superhero costume, but still heroically helping people to create web apps.