EPISODE 619

[INTRODUCTION]

[0:00:00.3] JM: Chromium is an open source browser that shares code with the Chrome browser from Google. A browser is a large piece of software with engineering challenges around threading, rendering, resource management and networking. To add to the complexity, chrome runs on iOS, Android, Mac OS X, Windows and other platforms. Chrome OS is an operating system based on Chrome. There's also Chromium OS, which is the open source version of Chrome OS. The chrome and Chromium operating systems are based off of the Linux kernel.

There's a whole lot of complexity in all these different projects and a lot of overlap between them. Chrome and Chrome OS and Chromium and Chromium OS and all of the deployments that they maintain on the different platforms on iOS, Android, the other operating systems I mentioned. To add to that, throughout the entire episode the line between browser and operating system is blurry in today's episode. There's so much resource management involved in the Chrome browser, that the Chrome browser actually has its own task manager.

For many people including myself, the browser is the main application that I'm interfacing with throughout the day. It has all my business applications in the browser, my e-mail, even many of my desktop apps that are not actually in Chrome the browser itself are running on electron, which is a framework for building cross-platform apps that uses Chromium. These apps like Slack, or Github, the Github app for Mac OS, these things are essentially built using Chrome. Chrome runs a ton of your applications.

David Bokan is an engineer on the Chromium team at Google and he joins the show to describe the engineering of Chrome and the development and the release process. David also gives his thoughts on future developments for browsers, apps and the internet. It was great talking to David and I'm hoping to do more shows on browsers and Chrome in particular, because I am really unfamiliar with this territory. I hope you like this episode nonetheless and I'm looking forward to doing more of these flavor of shows.

Before we get started, I want to mention that we are looking for writers and we're looking for a videographer for Software Engineering Daily. We've got a couple other roles and you can find those as well at softwareengineeringdaily.com/jobs and you can apply to those jobs if you think you would be a good fit, or if you know somebody who might be. If you're interested in getting involved in Software Engineering Daily on a lower commitment basis than an employment basis, you can go to our Github repo and get involved with the open-source community. That's github.com/softwareengineeringdaily.

We'd love to have you involved. We've got mobile apps, we've got a browser-based app and those apps are in the App Store. The browser-based app is softwaredaily.com. Again, we'd love to have you in the open-source community at github.com/softwareengineeringdaily. With that, let's get on with the episode.

[SPONSOR MESSAGE]

[0:03:23.7] JM: At Software Engineering Daily, we have a web app, we have an iOS app, an Android app and a back-end that serves all of these frontends. Our code has a lot of surface area and we need visibility into problems that occur across all of these different surfaces. When a user's mobile app crashes while playing a podcast, or reading an article, Airbrake alerts us in real-time and gives us the diagnostics that let us identify and fix the problem in minutes, instead of hours.

Check out airbrake.io/sedaily to start monitoring your apps free for 30 days. Setup takes only a few minutes. There's no complicated configuration needed. Airbrake integrates with all of your communication tools, from Slack, to Github, to Jira and it enhances your current workflow rather than disrupting it. You can try out Airbrake today at airbrake.io/sedaily. If you want to monitor and get visibility into the problems that may be occurring across your application, check out Airbrake at airbrake.io/sedaily.

Thank you to Airbrake.

[INTERVIEW]

[0:04:43.3] JM: David Bokan is an engineer on the Chromium team. David, welcome to Software Engineering Daily.

[0:04:47.7] DB: Happy to be here. Thanks.

[0:04:49.1] JM: Chromium is an open-source operating system. There's also Chrome OS. Well, I guess Chromium is both a browser and an operating system, I think Chromium OS would be the operating system version of it. How did the Chromium project get started?

[0:05:08.0] DB: I haven't been around since the beginning, so maybe I'm not the best person to speak to that, but basically around 2008 I think maybe a little bit before I went public, Google started working on a browser. I think the Chrome OS project started a little bit after that and it was basically to fulfill the vision of computing completely on the web. Chrome OS is these days I guess, it has its own local apps, but it's everything is running in the browser. The history, I'm not super caught up on, so –

[0:05:38.9] JM: No problem. We can talk about the contemporary state. What's the relationship between Chromium, Chromium OS, Chrome, Chrome OS? There's four different projects here we can multiplex this conversation across. Can you level set us with the relationship between these projects?

[0:05:58.0] DB: Sure. I've seen the confusion here quite a bit. Chromium is the open source bits of everything that we work on. That includes pretty much everything these days. The only things that aren't included in Chromium would be the proprietary bits of things, like the Flash Player and I guess some of the branding.

From Chromium, when Google actually goes and builds the browser, Chrome browser, so they add all those proprietary bits on at some of the branding and then we call it Chrome. That's just the browser. Now from that, there's also Chromium and Chrome OS and it's the same relationship, where Chromium is all the open-source bits, which you can go and download and build yourself. Then Chrome OS is the Google branded version of that.

[0:06:41.2] JM: Okay. If we're just talking about the operating system side of things, how does the user experience of Chromium, so for people who are totally unfamiliar with this operating system, the Chromium operating system, how does it compare to a Linux operating system, like an Ubuntu?

[0:07:01.3] DB: Chromium is still running a Linux-like kernel as far as I understand. The main difference would be that you don't go and install binary apps. For example, if you're running a Linux system you might go and grab some packages from your distro, or you might go and build some things yourself. On a Chrome OS system, everything is basically a web app. There's some technicalities nowadays. I guess, you can run Android apps and things like that in it, but the vision of it is that everything you do is basically just a browser window, and we tried to dress it up really nicely so it doesn't look like it's necessarily a browser, but you can install a web app or PWAs now are the big thing. You're not really running binaries. Everything is often in the cloud.

[0:07:44.2] JM: Can you define the terms browser and the term operating system from your point of view? Because I feel like Chrome and Chromium blur the lines between what is a browser and what is an operating system.

[0:07:58.5] DB: Yeah, definitely. I think the browser is an operating system in a way. It's the operating system for the web. Really just need something to run your apps. If you go to your Gmail, I mean, that's really an app which 10 or 15 years ago that might have been an app on your Windows machine or something, but these days it's hosted in the browser. Really the line is quite blurry. I guess, on a Chrome OS system the difference would be the kernel and how we interface with the hardware, which so the browser abstracts a lot of that. We're an OS on top of an OS in a way.

[0:08:34.2] JM: The Chrome OS perspective is that the browser is so integral to the user's workflow that we might as well integrate it closely with the operating system, or as we've explored, blur the lines with the operating system. In Chrome OS, how does the relationship between the browser and the operating system compare to if I was just using Chrome on my MacBook, for example?

[0:09:02.3] DB: For example, like the task launcher, everything there it's just launching basically a wrapper a browser. On your Mac, if you go and you launch Chrome, you get the browser window with a lot of the Chrome, like the URL bar and everything like that. In the Chrome OS experience, sometimes if you install the app which really it's still just a webpage with some fancy dressing below the hood, but you get what looks like a native app. If you were to run something on your OS X system, it doesn't look like it's a browser. It's the same idea. We want a webpage to not be the traditional way that you associate webpages. We want it to just the browser to get out of the way essentially.

[0:09:42.8] JM: The three core goals of the Chrome project are stability, speed and security.

[0:09:50.0] DB: Four actually, it's also simplicity. It's the Four S's.

[0:09:53.2] JM: Simplicity. Okay, I didn't know it had been expanded to four. If we're just talking about it from a browser point of view, let's just talk about in terms of the Chrome browser, if you want a browser that is stable, has speed, has security, has simplicity, one aspect that you want is a multi-process browser. Can you explain what it means for a browser to be multi-process?

[0:10:18.1] DB: Sure. I think Firefox is maybe the example of the single-process version of this. If you go to several webpages and different tabs in Firefox, they all look like they're running independently, but really they're all hosted in one big process, and that it also includes the browser UI. The difference in chrome would be if you go to two different domains in two different tabs, each of those has its own process. What that means is that we're using the operating system's isolation for each of those tabs, so they don't share the same memory space, there's no way for them to talk to each other in a direct way. The Chrome browser UI is also running in its own separate process.

This allows us to do sandboxing. For example, those tabs themselves don't have access to really powerful primitives on your system, so they can't go and read the hard disk themselves, they can't go and access some of the hardware devices. They have to go and ask the browser process to do that on their behalf.

[0:11:16.2] JM: Okay, so Chrome is a browser, which assumes it's always connected to the internet. All of the Google apps assume that they were online in the past, but an operating system has to be able to run stuff offline and apps for Chromebook for example, increasingly have offline functionality. What are the engineering constraints around Chrome apps today? Do they still need to be online, or is there offline functionality that's required?

[0:11:45.8] DB: Recently, there's been a much bigger push to allow this offline functionality, because you can't always be connected to the internet. It was great having a Chromebook and being able to use it everywhere until you got onto a flight and you needed to do some work. Really recently, there's been some really nice advances in web API. Serviceworker is a really big one, which allows you to do all sorts of local caching and basically really build apps that work very well offline. I think that the landscape in that sense is changing a little bit.

[0:12:17.2] JM: There are some principles that Chrome shares with other network application platforms. For example, when I think about Android I remember doing some Android development in the past and one principle is you never want to block the UI thread, and you also want your different network calls to be able to execute asynchronously, so you don't block when there's calls across the network. If you're in a browser, or if you're on a heavily networked Android application, both these cases you want your network calls to be asynchronous so that you can make a call out to the network and then continue processing locally and then wait for the call back across the network. Can you give some more details on how to architect the threading model around a heavily networked application, like a browser?

[0:13:08.1] DB: Yeah. This has been a problem traditionally and it goes back to the legacy of the web, which started out as a document platform and it got this application framework bolted on after the fact. Really, there is no threading in web apps. It's generally, everything is basically on your UI thread. This has been quite the challenge for people who are trying to build these high-performance modern apps. These days, the recommended pattern is to really chunk up all of your work and to really make sure that you're not running anything long on the JavaScript thread, because that is your UI thread.

If you're executing any cross-site network requests, really you should be doing those asynchronously and with promises and not waiting on the response for that. We have seen, like

sometimes when we go and do a performance analysis on a webpage and we'll notice like, “Oh, they’re doing a synchronous request here,” yeah, that's going to block up. You can't scroll all of a sudden, you can't do anything.

We've also tried to mitigate some of these issues. One of the patterns that's used in all browsers actually today is this off-thread user interaction. When you go to scroll, when you do pinch zooming, whenever we can we try to do that on a separate thread, and there's an enormous amount of logic and complexity around that and we sometimes have to fall back to the main thread anyway, but there's a lot that happens in the browser to make sure that we're staying off that really busy UI thread.

[0:14:31.0] JM: Chromium is the open source version of Chrome. How closely does the closed source development, how does that development track the Chromium development?

[0:14:43.1] DB: I've been on the Chrome team now for about five years, and I don't think I've ever touched anything in the internal repo. It's really a small part of Chrome. I think these, like the PDF plugin for example was open sourced at some point. It's really been moving more and more to get more of the bits of Chrome open sourced. Really these days, I think is just a couple of things that I think codex and some proprietary bits like that.

[0:15:06.9] JM: Okay. Can you talk about your interaction with the release process a little bit more, getting the Chromium browser from when it's just in its open source instantiation to how that code makes its way into production for Chrome users?

[0:15:23.8] DB: Sure. Chrome ships every six weeks, which is really nice. We try to keep an evergreen model where users are always up to date. How that works is generally we'll do development on the tip of tree, on our trunk and after six weeks release cycle. Some point, we’ll cut a branch, and so that will become the beta branch at some point. For a couple of days after that branch is cut, we do a lot of work to try and stabilize that branch, any issues that we find, we land first in the tip of tree and then we merge those patches back to the branch. Once that branch is getting to a place where we can actually ship it out to lots of users, we’ll promote that to a beta and beta has – I'm not sure about the numbers, but there's quite a significant number of users that run in beta.

I'm not sure exactly the time, but it's a couple of weeks, beta will actually go and we do that same stabilization process where we'll land patches and the tip of tree, verify that they're actually fixing issues and then merge them back to beta. Then after a couple of weeks of this once everything is stabilized and there's no major blocking issues, that beta will be promoted to stable and pushed out to all of the live users.

[0:16:32.4] JM: WebKit is a browser engine that is in Chrome. What is the purpose of WebKit? Can you describe what WebKit is?

[0:16:40.1] DB: Yeah, so actually these days Chrome is running on Blink and Blink is a fork of WebKit. Long ago, Apple created the WebKit project, which was the rendering engine for Safari. Initially, when chrome first released it was using that same rendering engine. Around 2013, we forked Blink from WebKit. I have been using that ever since. The rendering engine is basically everything that interacts with the web. What Blink will do is go out and grab the HTML, the JavaScript, the CSS, do all the parsing, build Dom and then basically put pixels on the screen.

It's generally responsible for everything that's not the browser Chrome. Your URL bar and the tab strip and everything, that's all more Chromium. Then everything that's actually content is Blink. That includes things like user interaction and input events. It doesn't include JavaScript. That's handled by V8, but it basically – the main thing that Blink does is take HTML and spit pixels out on the screen.

[SPONSOR MESSAGE]

[0:17:48.8] JM: Flatiron School is an outcomes focused coding bootcamp that trains people to launch a new career in technology in as little as 15 weeks. Flatiron School trains students across the full stack, from the front end to the back end and everything in between over 15 rigorous weeks. Students learn to think and build like software engineers, from developing coding skills to gaining an understanding of how products are designed and managed. Flatiron school has both online and offline courses. If you want to get started with an online course, you can go to flatironschool.com/sedaily and get $500 off your first month of Flatiron's online web developer program to get started with a career in software engineering.

If you like that online course, you might enjoy checking out Flatiron Schools in-person courses that are the 15-week immersive courses. On their website, Flatiron School has a jobs report that details the outcomes of students who have gone through Flatiron School and they report a very high percentage of graduates getting a job offer and accepting a job offer after their curriculum at Flatiron School.

Check it out at flatironschool.com/sedaily. If you want to get started with the online web developer program, you can get $500 off your first month and get a jumpstart in software engineering. Thanks to Flatiron School for being a sponsor of Software Engineering Daily.

[INTERVIEW CONTINUED]

[0:19:34.7] JM: A browser is a sandboxed way of accessing web applications. We've historically needed a lot of security between the browser and the operating system, but our files and other sensitive information are increasingly in the cloud. In some sense, the security model around our files is becoming different, arguably more secure. Does that change how we can think about the security barriers between the operating system in the browser? If our files are no longer as much relegated to the local OS, how does that change security barriers and what our applications can do?

[0:20:18.4] DB: I mean, you still have to store things locally. I think you still do really care about local security. That's one of the main reasons we do the sandboxing Chrome is you don't want – if a webpage compromises the process that it's in, you don't want it to suddenly be able to start reading and modifying files on your hard drive.

[0:20:36.9] JM: Although, we have things like Google Drive, or Dropbox, or name your cloud file system, you still do need a sandboxed model to some degree. You don't want your browser to have uninhibited access to the operating system. I should have just asked you what are the security barriers that need to be in place between the operating system in the browser?

[0:21:01.4] DB: Basically, access to any hardware or things like cameras and microphones obviously, you don't want just any webpage to be using without the user's permission. Basically

it's about what the user expects. The browser is delegating I guess the permissions between the OS and the user. There's a little bit of tension there, because apps always want to do interesting things, like you have a microphone and a camera on your computer for a reason, but you don't necessarily want every single webpage you visit to be using those. It's more of a gatekeeper than actual complete box around your OS.

[0:21:36.7] JM: Can you talk about the security model of Chrome just in more detail, some principals and some security mechanisms?

[0:21:43.8] DB: I'm not super well caught up on those, so maybe just in a general sense. I think the process model is really the main thing and it's in the forefront these days with the recent Spectre and spec hammer attacks. One of the big overarching projects currently going on in Chrome right now is the site isolation project. What that's doing is we already have process isolation between tabs. Again, if you go to two different domains and two different tabs, generally they'll get their own processes, so that if you go to eagle.com and it manages to compromise the render and break out of its security box, basically it can't go and start reading data from any other webpage.

We want to extend this to work on iframes as well. Today, if you go to any webpage, there's potentially dozens of different domains being embedded in any webpage, be it ads, or some other embed. We want to keep those all also in separate processes for the – mostly for the security benefits, there are some other potential performance implications that might be good. Yeah.

[0:22:45.4] JM: What are the threat vectors that you worry about the most? What are the security vectors that could potentially come through the browser and affect the operating system?

[0:22:57.0] DB: Yeah. I think definitely the isolation aspect is a big one of it, because again, if you're browsing, I might be just doing some random browsing when I have my bank website open in another tab. Really these days, since everything is so much in the cloud I may be less worried about things on my computer and I'm more worried about webpages being able to go and exfiltrate my bank data, or steal some of my passwords from a password manager and

things like that. I think the isolation and keeping everything in its own separate box is really the goal.

[0:23:29.7] JM: When I walk away from my browser, it may need to update and the way it works today is at least on my version of Chrome, I have to manually approve an update. How does the update system work in Chrome and in Chromium? Are there any forced updates you do? Also, what happens when an update actually executes?

[0:23:56.1] DB: Updates are automatic. I'm not sure if you have maybe some a special setting, but in general your browser will update in the background and you'll just get a notification that your browser is updated and you have to restart it. The idea there is very often, like people, if you pester people to go and actually do the update themselves, very often they just won't, because they don't know, or they – it's just that you're not really interested in the mechanics of the browser and updating applications. This is bad for security and all sorts of other reasons.

We really want to make sure that everybody is running on a fresh version so that they can browse the web confidently. We do our best to fix any new security issues that come out. Getting those rolled out to everybody as quickly as possible has been a really good thing, I think.

[0:24:45.2] JM: Yeah. Now, I think I realized it's that the color of the arrow changes, so I have – you see the green arrow first and that just means, “Hey, the browser has updated and you should restart it.” Then when it turns yellow and eventually red, it's saying, “Hey, this is actually more impaired if you really need to restart your browser.”

[0:25:05.4] DB: Right. Yeah, so what that's saying is it's already downloaded everything that you need in the background and all you need to do is restart. I see those arrows sometimes too and it is a little bit frustrating, but it is a little bit of peace of mind when you update and you know that at least all the known security vulnerabilities are hopefully already patched.

[0:25:23.9] JM: There are Chrome extensions, these different features you can add to Chrome. There's also, I think there's Chrome apps, right? There's actual apps that are running somehow

through Chrome. Can you talk about the different ways that people can run applications that are custom built for Chrome/Chromium?

[0:25:46.2] DB: I only really know a little bit about the extensions, even less so I think about apps. Basically extensions just allows you to go and write some HTML and JavaScript that can plug into to the webpages that you're browsing and that allows you to go and do all sorts of things like modify the page that you're currently on, or get some information out of it, or give you some extra actions. You can think of it as just plugging in a little bit of extra code to the webpage.

Applications, I know this only maybe at a high-level. I think it's basically just packaging up all of the JavaScript and HTML that you would normally have on a webpage. In the case of Chrome OS for example, you can add a manifest and some additional permissions. If a user actually installs it, and so they signal that they actually trust this app, it might get some more security permissions that the traditional desktop application might get as well. For example, like accessing a camera and well, these days I think you can do that from the web. Generally, like writing to disk for example.

[0:26:44.9] JM: Do you spend much time with the Chrome OS team? Are you mostly just thinking about things from the browser's point of view?

[0:26:52.9] DB: Interestingly, I started off technically on the Chrome OS team, but very quickly got shuffled into working more on the web platform. I've been working almost primarily on Blink for the last five years. Now and then I'll interact a little bit with Chrome OS folks. It's generally, they are very much plugged into the web platform, so everything that they do is running on the same platform that we're providing. Examples of this are scroll bars in Chrome OS, are the same scroll bars that we use on the web, so that's the thing I work on in Blink, and so I've had to interact with folks on the Chrome OS team to get them to work the way that they would want to for example.

[0:27:32.0] JM: What's the hardest thing about building a scroll bar?

[0:27:34.7] DB: They're very different depending on which platform you're on. Having to maintain the different behavior between Mac OS X, or Windows and Linux and Android, it's just a really big, big challenge to make sure that we can support every different behavior for every different UX. Then make sure that again, we have that whole complicated machinery that I mentioned before for having that off-thread user input.

We have separate paths for when you go and you do a scroll and it happens on that off-thread to make sure that it's nice and smooth, versus when you do it on that busy UI thread. Having all these different modalities and different UX and what looks a really simple issue actually ends up being tons and tons of maintenance work and headaches.

[0:28:19.6] JM: It sounds like a lot of that is because of the cross-platform aspect of it. The fact that you do have to be compliant with OS X and Android and Windows and I was looking at some of the best practices within the Chromium team around how these things are shipped, how you ship operating-specific, platform-specific notions of code for each of these different platforms. From what I saw, you have just these different files in a given directory. For the Chromium project, you might have eight different files and each of them execute only on a specific operating system version, or some specific user agent version.

[0:29:06.6] DB: Yeah. This has been one of the most humbling things. I think starting on Chrome is realizing just how difficult multi-platform development is, because again, you'll have these different versions of a feature and it turns out that like, oh, these might be really well tested on Windows and Linux, but because Mac works somewhat differently, it's a little bit hard to write tests for that, and so then you end up getting regressions and bugs on Mac.

It has been getting better over time. I think that we've been putting a lot of effort into making sure that things work as close to each other on different platforms as we can, but like for example, scroll bars are still an example of something that is really difficult to make it common and consistent across all the platforms.

[0:29:48.6] JM: You said you were doing most of your work these days on Blink, which you said that's the rendering engine?

[0:29:54.4] DB: Yes.

[0:29:56.1] JM: Do you have to think about the future of what the Chrome application runtime is going to be, or the Chromebook world, because there's the fact that Google is making Android apps run a bowl on Chrome and as we talked about, there's the Chrome extensions and the chrome applications. Do you need to know about any of this, or can you just stay in your lane and focus on the Blink side of things?

[0:30:25.7] DB: I keep up mainly a little bit out of interest in curiosity. For the most part, I focus on the web aspect of. I think again, if you think of it as an OS, I think those are maybe more like the OS for Android apps and the OS for some other apps. We're really working on the OS for the web, so Blink is the web – we considered the web platform. We're building the OS for web apps. For the most part, that's very separate from all the other things that are going on in Chrome OS.

[0:30:54.9] JM: What language is Chrome written in?

[0:30:56.9] DB: Most of it is in C++, that's the vast majority. We have some scripts and tooling that that is done in Python. Then the Android version is a little bit of Java as well for things that are UI, but primarily C++.

[0:31:12.5] JM: Has there been any assessment of can we do any of this in Go, maybe use Rust? Have other languages been considered?

[0:31:20.8] DB: The Chrome codebase is enormously large, so I think at this point the ship is pretty much sailed. I have just been keeping tabs on Firefox and their efforts with Rust and writing – they were writing that new engine in Rust, and I think –

[0:31:35.4] JM: Servo?

[0:31:36.4] DB: Servo, right. That's super interesting and very cool technology. I think the challenge of bringing something the size of Chrome all the way into something new like that would be pretty daunting.

[0:31:48.7] JM: When you look at that browser that they built with Rust, or the browser engine, what are the advantages of Rust from a language level? Are you familiar enough with the language to know what the idea? Is there anything you envy that you – abstractions that you wish you had in C++?

[0:32:06.5] DB: I don't know much about it. The thing I've heard a lot it is I think like memory safety. In C and C++, it's very easy to shoot yourself in the foot and start reading and writing memory past what you intend. From what I understand, that's more difficult if not impossible in Rust, so I guess that's the really nice thing like they touted the security benefits of it. Yeah, I don't know it in depth enough to actually have a more nuanced position.

[0:32:31.6] JM: What challenges around memory management do you encounter in writing a browser? Because if your browser has a memory leak, it's really unfortunate because I never close my browser. That's one of the reasons I see those red arrows that tell me to close Chrome and reopen it. If there were a memory leak, then this would just be a catastrophic application for me.

[0:32:54.0] DB: Yeah. Well one of the biggest challenges is often the memory leak is not necessarily in Chrome, but it's in one of the web apps. For example, I keep my Gmail tab open for months at a time. Once in a while, we'll notice that like, “Hey, my Gmail tab is using 4 gigs of RAM.” It's also possible that the apps are actually misbehaving as well. Again, Chrome is a little bit of an OS of its own, so it's not necessarily that there's a memory leak in Chrome, but it could be a memory leak in the web app.

The other challenge there is that Chrome is such an enormous project with so many different pieces and components and features that it does tend to balloon fairly quickly. We do our best to try and minimize how much memory it uses, but it's a constant battle. We're treading water and then every once in a while do a little bit better, but the arrow is a slow creep up.

[0:33:45.5] JM: What constraints can you put on different tabs? For example, if there's some tab I have open to a recipe site and that recipe site just has a million different JavaScript add tags. I don't find that it just completely destroys my operating system performance, so clearly

there's either some sandboxed amount of memory that it's allocated. It's not able to leak into dominating my entire memory space on my computer. Are there some constraints around how much memory a given page can take?

[0:34:21.4] DB: Yeah. Today, everything on that one page would be in its own process. Your operating system is probably doing a bit of work there. I don't think that we have an explicit memory limit on that, but that does mean that the page itself is running really badly, right? One of the interesting ideas was with this site isolation project. If you take all of those extra ad, ad iframes or who knows, a Bitcoin miner maybe, and then just throw it into its own process and maybe throttle that process and give it lower priority, that would actually make the main application run much better.

[0:34:54.2] JM: Can you talk more about how that works? I know you touched on it earlier, but explain how site isolation works.

[0:34:59.6] DB: Right. When you load a webpage today, and it might go and it might embed a bunch of iframes from different domains, all of those go into one process in Chrome. What would happen in site isolation is if we notice that the iframe is from a different domain, we'd want to put that into its own process. That way it can't talk directly to the actual page. The challenge there is that a lot of the web relies on this being able to – it changes the timing for example of how a webpage will load, so you might be able to post message from your application to some iframe. Now because that iframe is in a different process, it might not have loaded yet, or things work a little bit differently. Internally basically, it's similar to as if you were to open that iframe in its own tab. It has less of a connection to the parent tab.

[0:35:49.0] JM: Understood. Can you talk more about the resource management of Chrome from the point of view of the relationship between Chrome and the operating system? For example, when I open up task manager on my, or whatever it's called, activity monitor on my Mac and I see four or five different Chrome processes that are running. What are those processes that are running? What are those different things doing? Is that is there some mapping from those processes to webpages, or two different things running in Chrome? Can you help me understand what's going on there?

[0:36:29.5] DB: Sure. It's a little bit complicated. A bunch of those will be, like those tabs, so I mentioned that often if you open a new tab then that that page will get its own process. You'll actually see that show up in the activity monitor. That doesn't necessarily always happen, so if you've got two or three tabs and they're all in the same domain, those will generally be I think in their own process. They'll have one process for the domain.

This varies between platforms, so on Android for example because of memory constraints. For the most part, everything gets shoved into one process like a traditional browser. There's also that browser process I mentioned, so that's the main overseeing process and that does all of the privileged work that needs to happen like writing to disk and then doing that request and things like that.

There is also a process for the GPU. For example, your process for the webpage will actually send a bunch of information to the browser process and then that will get put together with all the different information like your UI tab strip, and all the other things and then that gets all sent to a GPU process to actually get pushed out to the graphics card. It's really a big machine with lots of complicated moving pieces, so you're seeing the internal guts there. If you're interested, you can also – there's a – in Chrome, Chrome's got its own task manager. If you go to the little hamburger menu and I think in more tools, there's a – I forget the exact name, but it's something like a task manager, and you can see all of the different processes that are running in Chrome and what they are and how much CPU they're using and things like that. It's a little bit more informative than just seeing Chrome help or process in the Mac task manager.

[0:38:03.5] JM: It's actually called task manager. I just opened it up. I'm looking at it. It's got a lot of sub-frames.

[0:38:10.9] DB: Yeah. That's site isolation I think. Some site isolation experiments going on right now. Well, sorry it actually shipped on desktops in Chrome 67, I think, if I'm right on that. Yeah, so on desktops you will actually get isolated sub-frames now. You'll see a process for iframes on a page.

[SPONSOR MESSAGE]

[0:38:37.1] JM: This episode of Software Engineering Daily is sponsored by Datadog. Datadog integrates seamlessly with container technologies like Docker and Kubernetes, so you can monitor your entire container cluster in real-time. See across all of your servers, containers, apps and services in one place with powerful visualizations, sophisticated alerting, distributed tracing and APM. Now Datadog has application performance monitoring for Java.

Start monitoring your microservices today with a free trial. As a bonus, Datadog will send you a free t-shirt. You can get both of those things by going to softwareengineeringdaily.com/datadog. That's softwareengineeringdaily.com/datadog. Thank you, Datadog.

[INTERVIEW CONTINUED]

[0:39:31.4] JM: I'm looking at this task manager and I sorted by CPU consumption and memory footprint among all of the different tasks that are running across Chrome. At the top, it looks like the CPU is – the browser itself is dominating in terms of CPU. What are the most CPU-intensive tasks that the browser itself has to do?

[0:39:59.6] DB: That's a little surprising to me. Usually, I would think that the actual tab processes would be the ones that are fairly CPU-intensive, particularly if you're doing any video or audio like we are now, I would expect that to actually be doing quite a bit of work.

[0:40:14.2] JM: Zencastr is up there. We're recording over Zencastr, Zencastr is taking up 62 whatever CPU units these are, and then the browser is it got 121 CPU units.

[0:40:29.4] DB: Right. Okay, that does make sense I guess, because we're recording some audio and so it's got to be sending data between the browser and the tab process, so both of those would actually be in heavy use.

[0:40:40.3] JM: Fascinating. Yeah, and the other things I'm seeing are tabs, apps and then a bunch of, like I said subframes under accounts.google.com etc. If I were to go to my computer's, my operating system task activity monitor itself, I would see something different. I think you mentioned, so if I've got a bunch of things that are in the same domain, so for example, a bunch

of Google Docs processes, would those get co-scheduled into some task manager notion that will just show up as one single process under the activity monitor of my operating system?

[0:41:21.2] DB: It depends. There's a lot of flexibility in which how we split up different tabs and different pages between processes. I'm not super caught up on how we actually do that today. I know that there's been talk if your system is under memory pressure, then perhaps we'll squash things into fewer processes or not. In general, if it's on a different domain, I think it should be in its own process. If it's from the same domain, I think it should be in the same. Yeah, it's been a while since I've looked at that, so I'm not sure what that is today.

[0:41:53.4] JM: No problem. What about memory management and garbage collection more broadly? How much time do you have to spend thinking about garbage collection?

[0:42:03.1] DB: Luckily not too much. There's two garbage collectors at least that I'm aware of in Chrome. The first is the V8 garbage collector, and that's for everything JavaScript. That's more to do with page authors and then the web apps. Then internally inside of Blink, we actually a few years ago launched our own garbage collector for things going on inside of Blink. When I'm writing code for the most part, I just have to make sure that when I initialize things, I hold them in the right handles and it basically figures out when the object is no longer actually in use and make sure that it tears everything down.

Blink used to use reference counting, which was a little bit trickier and would often get things like cycles, so then you leak memory like that, but I think it's been a really – over time I think that the garbage collection has shown that it's actually been really nice thing to have.

[0:42:50.9] JM: I want to take a step back and discuss the team aspect more and the interaction and how development works on the Chrome team. Can you describe the structure of the different teams that are working within Chrome?

[0:43:08.4] DB: Right. We have the web platform team, and so that's the team I'm on. That's responsible for Blink, but also generally things like spec work and working with other browser vendors on making sure that the web is interoperable between different browsers and how we evolved the platform in the APIs.

Outside of that, there's more like I guess Chrome browser team, which would work on the actual UI and various features like autofill and things like that. Beyond that, I guess Chrome OS would be its own separate team. Just from what I know within the web platform team for example, we break up into various small sub teams. I work on the input and scrolling team. There's different teams for things like painting, for layouts, for style. It's quite a large team and it breaks down based on I think responsibilities and then ownership of code.

[0:43:58.4] JM: Specifically, the Blink engine and the scrolling feature that you're working on right now, that's a core piece of functionality that's really important to all the different browsers that get shipped across all the different user agents as we've already touched on. What's the process for being able to test all those different platforms?

[0:44:22.0] DB: Right. Testing is always a bit of a challenge, again, just because we do have so many different configurations and options. We do have a large repository of tests. I think one of the most exciting things going on recently is this push for web platform tests. What that is a common repository shared between all the different browser vendors, and so anybody can go and they can add tests there and this is the preferred method. I think now whenever we can, we try to add web platform tests for things, because that means automatically we get coverage across not just Chrome, but also Firefox and Edge and Safari.

When we find a bug, if we go and we fix it and we create a web platform test for it, we can see on a dashboard, okay this bug exists. This bug is now fixed in Chrome, maybe it never occurred in Edge, then it's still broken in Firefox and Safari. It also helps with new feature development, so when we build new features we add web platform tests and then that really lowers the barrier of entry for other browser vendors to go do it, because they can very quickly write up an implementation, see whether or not – where it's interoperable and where there might be issues.

[0:45:25.7] JM: How does the release process work when you've got all the – the tests are approved, can you just go ahead and ship your code and just continuously release it, or is there – there's a release cycle to different versions of Chrome?

[0:45:43.2] DB: Right. The major milestone versions is based on a six-week release cycle as we've talked about. Every time that we land a piece of code, it goes through a set of try bots that we have, so it goes and runs almost all the tests in our test suite, and it will only land in the Git repository if it actually passes those tests. Past that, before we ship a Chrome version, like everything is constantly building and constantly running tests, so hopefully tests should always be passing. Generally, where we do see failures and bugs, it's usually things that we were missing a test, or some new feature and we didn't quite have that Edge case tested.

[0:46:18.4] JM: There's a running debate around to what degree browsers should be able to block ads, or how they should regulate web experiences. I'll just say for my first-hand experience, I'm at the point where if I see a link to a domain name that I'm not familiar with, if I'm on my phone, I will usually just not click it, because the loading experiences for so many of these sites are just so offensive and just make my phone run really slow and it doesn't work very well. What's your perspective on to what degree of browsers should regulate the minimum tolerable web experience?

[0:47:03.9] DB: I'll just clarify. I'm speaking on personal capacity and not Google, or anybody.

[0:47:07.4] JM: Yeah. Yeah, of course.

[0:47:09.3] DB: There is a tension there, because like you said, the experience does really suck. If you just let everything in, then people will just stop using the web, because the experience will be so much better elsewhere. At the same time, I think if you block everything that will really undermine the business model of the entire web. That might be good in the short-term, but then if you take a longer view, there won't be any content made for the web, because it will be very difficult for people to make a living off of it.

I think the ideal thing that you want is to strike a bit of a balance, where you really block out the most egregious ads and make the experience good enough that people don't mind the fact that there are ads on the page, because it doesn't really affect the experience of the page.

[0:47:49.1] JM: Yeah. I tend to agree with you. When we think about all these different runtimes for networked applications, we've got mobile apps, we've got web apps, we've got that like we

said, browser extensions and Chrome apps and electron apps and all these different surfaces for applications. Then there's also things like the instant, or what's it? Progressive web apps, these kinds of things. There's a lot of different experiments that are being run. Then you've got things like react native that make it easier for these cross-platform applications to be written. Do you have a belief for – we've got all these different application surfaces are going to go. Do you have a strong belief in terms of how things are going to work out, or do you just feel we're running a bunch of different experiments and time will tell?

[0:48:45.2] DB: Yeah. I mean, I'm probably a little bit biased and that I think and at least hope that the web wins out. My personal feeling on this is that it's maybe we're seeing a similar curve we did with when desktops first came to prominence, where everything was just a binary that you install, and then eventually the web caught up with the capabilities of PCs. Then you started to see more and more things shift on the web. With mobile, we're seeing a similar thing where for a long time you really couldn't do very much on the web, and so everything was an app. I think we're starting to see a bit of an inflection here, where you can do more and more things on the web where what you needed an app like an Android or an iOS app to do a few years ago, you can now do on the web.

Personally, I think that the web has a lot of really nice advantages, just like everything is indexable. You can you can go find it. If I go to a restaurant or something and they have some special deal, you don't have to install an app that might be tens of megabytes large and you have to go through a Play Store. You can just go and type in a URL and everything is ephemeral. It's not changing like your system state, or adding anything to it. That's personally where my loyalty would lie. I know people differ there. Android is doing a lot of work to bring some of those advantages of the web to Android apps. I suspect we'll see a little bit of a more of a convergence, where Android apps become a little bit more like the web and the web becomes a little bit more app-like.

[0:50:06.6] JM: Yeah, somebody was telling me about something called slices, where I think you can run some portion of an Android app and you could open that potentially from a browser. I know as you said, you're not super familiar with the crossover between Android and Chrome.

[0:50:21.8] DB: That's one of the nice advantages of the web is that you don't have to download the entire app before you run it. If you go to a webpage, you're only really taking what you need at the current time and everything just downloads in the background. Some of the comparisons between the progressive web apps these days, which is really basically just a packaged up webpage. The difference in size is an order of magnitude.

[0:50:44.6] JM: Yeah. The experience there is if you open a progressive web app on your phone, for many of at least the simpler apps, which a lot of business applications you can get a long way with “simple apps,” they feel they might as well be an Android app, but it's a progressive web app, so it's basically like you're opening a browser tab, right? But it's a domain-specific browser tab?

[0:51:11.0] DB: Yeah, basically. Like you're running the browser, but we do everything to not make it feel like the browser. For example, we might remove the URL bar and let the page customize things a little bit more to how it should feel. I think the best examples of this are where you can open an app and not even realize that it's a web app. I know, I think the Twitter Lite app was actually remarkably well done and it feels very native and then it's just a great example of what you can actually do with today's web technologies.

[0:51:37.0] JM: You think that performance is just going to get better and better over time and eventually this will be – things will go in that direction, or that would just be your preference that things would go in that direction?

[0:51:50.0] DB: If I had a crystal ball, I'd feel a little more comfortable making predictions. Yeah, I do think we have been adding capabilities and missing features to the web to let these things evolve this way. I think we're going to continue to do that, so I do think things will get better and better. Like I said, at the same time it's coming at it from two directions, so like native apps will also start to become more lightweight and more web-like and then get a lot of these advantages as well.

[0:52:15.5] JM: When do you think you can get assistant into the browser? Because the Google assistance is quite useful.

[0:52:21.8] DB: Yeah. I don't know. It is a really interesting feature in Android. It would be nice to see it integrated more with the web, but that's hard to do. I'm not really even sure how we would approach that.

[0:52:33.1] JM: Okay. Last question, I don't know to what degree you can talk about this, but there's this fuchsia project within Google and it's got its own kernel called zircon, it's got its own graphics engine called skia, it uses dart. Do you have any perspective, or any knowledge, or anything you can talk about around that operating system?

[0:52:53.5] DB: Not super much. I don't exactly keep up with what's going on there. It does seem like it's a project where they're building out a lot of new technologies and trying things out. It seems really exciting from that point of view, but I don't actually –I don't know the details there.

[0:53:08.3] JM: Okay. Fair enough. Have you looked at dart at all?

[0:53:10.8] DB: Not in depth. I know it was a replacement for JavaScript than it was meant to be, like fix a lot of the things that are wrong with JavaScript. Unfortunate, I think the way that the web evolves it's very difficult to take something that's already in use in practice and just wholesale replace it. Dart had that very much going against it, because it was just trying to replace JavaScript. The way that the web has grown, it's always been slow incremental changes until you take a look at the progress that's been made over 10 years and it looks almost a completely different thing now. Yeah.

[0:53:40.6] JM: Actually, okay real last question, is there any way that web assembly has changed how you think about browsers?

[0:53:47.6] DB: Web assembly is really interesting because it basically now allows you to take almost native code and compile it into the browser in a really performant way. I think you're going to start seeing more and more high-performance type apps be feasible to do in a browser. Recently, we notice I think Sketchup, which is a 3D modeling sketching tool thing has replaced their app with a web assembly version in the browser, and it runs surprisingly well. That's a really exciting new development.

[0:54:16.7] JM: All right. Well I'll have to inspect that further. I wonder when they come out with a progressive web app. David, thanks for making the time to come on Software Engineering Daily. It's been really fun talking to you.

[0:54:25.4] DB: Thanks very much. The pleasure's mine.

[END OF INTERVIEW]

[0:54:29.7] JM: If you are building a product for software engineers, or you are hiring software engineers, Software Engineering Daily is accepting sponsorships for 2018. Send me an e-mail jeff@softwareengineeringdaily.com if you're interested.

With 23,000 people listening Monday through Friday and the content being fairly selective for a technical listener, Software Engineering Daily is a great way to reach top engineers. I know that the listeners of Software Engineering Daily are great engineers, because I talk to them all the time. I hear from CTOs, CEOs, Directors of engineering who listen to the show regularly. I also hear about many newer, hungry software engineers who are looking to level up quickly and prove themselves.

To find out more about sponsoring the show, you can send me an e-mail or tell your marketing director to send me an e-mail jeff@softwareengineeringdaily.com. If you're a listener to the show, thank you so much for supporting it through your audienceship. That is quite enough, but if you're interested in taking your support of the show to the next level, then look at sponsoring the show through your company.

Send me an e-mail at jeff@softwareengineeringdaily.com. Thank you.

[END]