

Search Off the Record - 58th episode

[00:00:00] ♪ [music] ♪

[00:00:10] **Martin Splitt:** [00:00:10] Hello, and welcome to another episode of Search Off the Record, a podcast coming to you from the Google Search team, discussing all things Search and having some fun along the way. My name is Martin and I'm from the Search Relations team, and today, we have two very special guests: Edu Pereda who has worked with Gary for over a decade, and most recently has been working on Search Open Sourcing together with Gary. We've discussed this in the previous episode, in this podcast, if you're curious. And Pascal, who has been working a lot on modern content formats such as web stories, which we also discuss on this podcast in episode 10 with Pascal, in fact.

[00:00:49] In today's episode, we are going to talk a little bit about JavaScript and TypeScript and everyone's famous loved topic, I think we should have Gary here because he loves JavaScript just so, so much, but it always surprises me when people ask me, "Why is JavaScript such a big topic?" And I think there's a bunch of projects within and outside of Google that are using JavaScript heavily, and I know that both of you are involved in some. So, would you like to introduce real quick what you're working on and what your JavaScript projects look like? Let's start with Pascal. What do you work on?

[00:01:27] **Pascal Birchler:** [00:01:27] My day-to-day work is building a visual web stories editor, like a JavaScript-based editor that runs in the browser, or basically within a WordPress plugin. And that is basically my daily battle with JavaScript, working on that editor.

[00:01:44] **Martin Splitt:** [00:01:44] Right, and that sounds like a super interactive application, and as you said, like a visual editor which is pretty advanced stuff, I would say. That's not easy to do without JavaScript at all, is it?

[00:01:56] **Pascal Birchler:** [00:01:56] Without JavaScript, this editor wouldn't really be possible, because it's what allows us to make this web page interactive, and dragging things around on a page and interacting with browser APIs. It allows us to talk to your webcam, basically, so that we can allow you to record yourself with a webcam. That's all JavaScript.

[00:02:19] **Martin Splitt:** [00:02:19] Absolutely. And what are you using it for, Edu?

[00:02:22] **Edu Pereda:** [00:02:22] Hey. I've been building for a long time internal applications for debugging stuff. I worked for a long time in Search where we were building internal applications for debugging the different parts of Search. And then, I switched to security a bit before the pandemic, and then continued building basically web applications that internal users use. Let's say they are tools in general, without going into any specifics. And we do use JavaScript because... Well, TypeScript, actually because it's basically their tools that you have to make dynamic tools in the browser.

[00:02:59] **Martin Splitt:** [00:02:59] Hmm, interesting. And both of you are working on tools. So you're mostly working with applications rather than standard off-the-cuff websites, I would say. Why do you think people have such a, say, strong emotion towards JavaScript? There's a bit of a hate/love relationship thing happening out there, I think. Any idea why that is?

[00:03:21] **Edu Pereda:** [00:03:21] I think I have a feeling of it. Applications written in JavaScript, if you're going to be using them to just show content, the loading of that content is going to take longer, just because it needs to load this extra JavaScript. But I'm guessing that there are so many good frameworks today to make web applications dynamically with JavaScript that for developers, it's a natural way to start there because it makes it so easy to make very dynamic, and you can put whatever you want, be it content or any other functionality. So I guess some developers love it because of that, and then some users hate it because they cannot get away from the past where everything would load instantly. I don't know, that's my feeling. I don't know, Pascal, what do you think?

[00:04:09] **Pascal Birchler:** [00:04:09] It's actually a great point. I think JavaScript definitely had a lot of negativity connected with it in the past because of larger bundle sizes of web page sizes. One part I do like about JavaScript and you can't really do without JavaScript is all the tooling involved with creating websites nowadays.

[00:04:31] Just the other day I wanted to create a small landing page and I wanted to have some build step,

like minifying CSS and then making sure the fonts work, etc. It's really hard to do that with another programming language. All the tooling for creating websites is basically JavaScript-based.

[00:04:50] **Martin Splitt:** [00:04:50] That is true, that's absolutely true. And I like the angle of users are dissatisfied with things that take longer to load and JavaScript tends to be that, and then developers are stuck to it because it's the way to pretty much build anything on the web today. And also the tooling, as you say, the minification, everything that you want in a modern website is very JavaScript-focused and very JavaScript-heavy and that's interesting.

[00:05:16] The other thing that I personally find tricky is that with JavaScript, you get very powerful possibilities, but at the same time, you also have to reinvent things that the browser already has. For instance, React has this virtual DOM that basically more or less reinvents what they say is a better version of the Document Object Model built into the browser. A lot of accessibility concerns come into mind when it comes to making sure that screen readers and keyboard input still works as the browser would by default give us working input facilities there. And I think that's something that might also contribute to this weird, strained relationship, I guess.

[00:06:00] But I heard that some developers are also not super fond of JavaScript to begin with. And it sounded a little bit like that when you, Pascal, said, "Oh, I just wanted to build this simple thing and I needed to deal with all this JavaScript tooling." [laughs] So I guess developing JavaScript comes with its own [sighs] "challenges," let's put them that way.

[00:06:22] **Edu Pereda:** [00:06:22] Wait, may I say something controversial first?

[00:06:25] **Martin Splitt:** [00:06:25] Oh!

[00:06:25] **Edu Pereda:** [00:06:25] I think a lot of the heat has to do with Google itself and the fact that it took a long time for Google to render dynamic pages that were using heavily JavaScript. It took a long time for Google to adapt to that. And then people were like, "Oh should I use JavaScript in my website that just showed content, or should I not?" And then like, "How will Google do that, how will Google render it?" So I think there is a lot of that as well for writers of applications.

[00:06:55] **Martin Splitt:** [00:06:55] I mean out there, SEOs downright often fear JavaScript, and without acknowledging that a lot of things are not even possible. And it's not that every JavaScript that you use actually destroys performance for the users or makes it harder for search engines. An example of that would be service workers that actually allow websites to work offline or make better use of the network in a faster way, where they basically load cached content pretty much instantly and then rely on the network to give them more up-to-date data.

[00:07:24] So, does the challenge that search engines, and I wouldn't just say Google, I would say in general every search engine faces the challenge that a lot of websites are using JavaScript to load content, and such engines just need to go with a web or go where the web is going and just support that and it has taken us a tremendous amount of time, that is true. However, we do have a modern Chromium rendering things now, so that has been a pretty big undertaking, so that's pretty cool.

[00:07:56] **Edu Pereda:** [00:07:56] JavaScript has improved a lot in and it was super bad in the past. A lot of people that are against it have not seen the modern JavaScript or even TypeScript, how different it is from how it was in the past. But a few years ago, it was horrible to work with JavaScript, there was no tooling, there was no types, it was like the wild west out there because you could... JavaScript itself is so flexible and you can do so many things that it is very, very easy to shoot yourself in the foot. And a lot of the modern tooling around Typescript, and all the development of this ECMAScript standards that have come through, and the TypeScript usually adopts early. Those have helped a lot to make the language more readable, more easy to maintain. I think those things had made the language...

[00:08:46] I used to hate JavaScript, just to be clear, and now I love it, I like it, I cannot do without it.

[00:08:52] **Martin Splitt:** [00:08:52] That's interesting. And I feel the same, having been a JavaScript developer for a very long time. It felt like you are basically riding a kiddie bike with training wheels, until a bunch of language features have been added to the language that made it a lot more mature and capable and powerful. And it's quite funny we had this episode on the podcast, I'm not sure what the episode number has been, but where Gary and I talked about JavaScript and he was surprised to hear what can be done and is different now in JavaScript that he wasn't aware of. So I think that is actually true for a lot of people.

[00:09:26] And what I would like to go back to, because you said a bunch of things like TypeScript and types and stuff like that, before we get into the nitty-gritty details, I'm actually interested and intrigued by the fact that I have two people who are working on very different kinds of applications. I know that you, Edu, are working on internal-facing applications, which I feel has different requirements and different development priorities than what Pascal's working on, which is an open source interactive application that may be deployed to millions or billions of users at some point.

[00:10:01] So, how does internal development and external-facing development look like? What is the... What are your requirements, what are your challenges in the language, and what are things that you do to address these challenges with the language? Let's start with internal applications because I feel like that's a good starting point that then we can build upon.

[00:10:25] **Edu Pereda:** [00:10:25] For internal applications, there are a lot of things that you just don't need to focus your efforts on. First, you will have a lot fewer users. There are a lot of things that you will not care about because also, they are internal users, googlers, basically, and they all speak English. Right from the start, you don't need to care about internationalization; you just write your application in one language. Everybody will be using these apps in desktops or laptops, so right from the start, there, you don't really need to care about mobile. There are sections, but most of these internal applications--

[00:11:01] **Martin Splitt:** [00:11:01] is that true? Is that still mostly--

[00:11:03] **Edu Pereda:** [00:11:03] It is, because for security constraints. At least the apps that I've been working on, you cannot really access them from a phone. Policies from the phone will let you access public Google applications like a Google Drive and Docs and stuff like that. But then for the internal ones, there are a lot of security restrictions and you can only access them if you have a fully corporate phone with a lot more policies and constraints and whatnot. And not everybody had those. Making sure application works well everywhere takes time, and then you don't always have the time. So it's very frequent that we don't even focus on the mobile version of an app, because it's not going to be used anyway, or very, very seldom will be.

[00:11:48] And then what we want is to be able to develop fast. So when we focus on fast development cycle, and basically, you touch your code and then you get it refreshed on the web page instantly or as fast as possible so that you can try new things and continue developing. That's the biggest, biggest thing that an app needs to have.

[00:12:10] And then for an external app, you need them to be scalable, you need them to load fast. There's going to be users accessing them with very bad connections, and there is a ton of things that you need to think that you don't need to care when you're writing internal apps. And also, you get a lot more flexibility because you don't follow as many guidelines...

[00:12:33] I don't really know if it's true, but I can imagine that if you want to change the font in Gmail, there's going to be a lot of red tape that you'll have cut there, whereas in an internal app, you just do it, and then maybe somebody complains and maybe somebody doesn't, I don't know. That's my experience. And then we have different frameworks suited for different things. I don't know if Pascal, you want to say some bits before I continue, because I can continue rambling forever.

[00:12:58] **Martin Splitt:** [00:12:58] [laughs]

[00:13:00] **Pascal Birchler:** [00:13:00] [laughs] Well, it makes me want to work on internal applications, actually, because as you briefly mentioned, working on an external application and an open-source program like I'm working on, you have so many more constraints, users with different browsers, weird setups, edge cases that don't work. It's even more complicated by the fact that this web story editor that we are building, we actually ship it as part of a WordPress plugin. So it's not even like a standalone single page app or whatever, but it's embedded within a WordPress admin. So you have to deal with all sorts of conflicts from other WordPress plugins or the site configuration you have to use. For example, the internationalization layer from WordPress, you can't really roll your own. So a lot of challenges come directly from working on such a project.

[00:13:57] **Martin Splitt:** [00:13:57] Interesting. And I think that's generally one of the bigger challenges for people getting into frontend web development with JavaScript, that especially when you're coming from a more internal world, or let's make it even more extreme, a backend world where you have control over everything, you know what kind of server the thing is running on, you know what kind of requests you're getting, you can ignore a lot of things, you can just say like, "I always have a fast connection to this piece" and "I always have this much CPU available." Whereas in frontend development, especially when it's

external, you know pretty much nothing; you can assume nothing and you have to defend against everything, be it a too small of a screen, be it a very slow network connection or a flaky network connection that might fail every couple of minutes or something.

[00:14:47] So it must be interesting to be dealing with that, but challenging at times as well. And I imagine embedding as part of WordPress has probably its own big set of challenges, isn't?

[00:14:58] **Edu Pereda:** [00:14:58] Yeah, I think you're absolutely right, yeah.

[00:15:00] **Pascal Birchler:** [00:15:00] Absolutely. I think it also starts with just what kind of web APIs or JavaScript APIs are available in your browser, because not every browser supports the same set of APIs or features, or maybe only to some degree. That makes it also difficult when you want to build some applications like, "Oh, this doesn't actually... it's not supported in Chrome, I have to use a polyfill or whatever."

[00:15:25] **Martin Splitt:** [00:15:25] And I feel that's really interesting. The entire note on how the frontend side of things, especially with external applications has to be so defensive. But the other thing with the availability or not availability of APIs is one thing, the other thing really is that the language, I feel to this day, even with all the additions that have been made to the language in the past couple of years, as Edu has pointed out, I think the biggest one was probably ECMAScript 2015, and then everything was added afterwards in yearly steps.

[00:16:02] So that fantastic language has been evolving greatly since the last couple of years. But it hasn't been designed from the get-go for larger-scale applications. It is a scripting language, which means you have a website, you have things in the DOM, in the Document. It hasn't been designed to deal with large applications from the get-go. So it is meant for scripting, which means you have a bit of content on the website and you want to respond to actions or events on it like clicking on a button or adding something when the user scrolls or something like that. It hasn't really necessarily been primarily built for large scale or middle scale applications that are fully interactive applications built in the browser.

[00:16:47] So I'm guessing both of you are dealing with the fallout from that as well. Don't you feel that the larger the application gets, the more complex the application gets, the harder it gets to actually make the different parts work together, and how do you deal with that? Pascal probably with a very visual tool that might be even trickier because you're dealing with everything from storing data, loading data from the network, and then also the whole user interactions layer, how do you structure your application? How do you make sure that the different parts play together well?

[00:17:19] **Pascal Birchler:** [00:17:19] That's an excellent question, Martin. And you're definitely right, JavaScript has come a long way from a scripting language to now having to support all these rich Internet applications. Luckily, I think the tooling has come a long, long way as you say, like ES 2015, but also the bundlers, the frameworks and libraries like React, etc., to make it relatively easy to make sure that your application is performant and doesn't use too much memory, for example. Browser APIs have also come a long way in that perspective. JavaScript engines have become much, much faster as well, but I think a lot is also now due to improved documentation and learnings from other people. So that nowadays, I think it's pretty obvious that you can write these applications in JavaScript in a performant way.

[00:18:15] **Martin Splitt:** [00:18:15] Okay, so then the performance angle is covered, but how do you make sure that all the different bits and pieces, because I know that for instance, there's this infamous media playback thing where you can ask the browser if you can playback a certain type of audio file, and it comes back with strings, and the strings are probably, maybe I think empty strings or something like that. It's a really, really weird way of dealing with this, and I can see so many potential problems from that if that is used in one part of the application and then another part of the application needs to deal with it, because if you're not 100% sure about the type of data that you're getting from it and what it means, then you might run into these really weird problems.

[00:19:00] I recently ordered something in the Web shop that was using a client-side framework to render out its entire shop system and in the cart, I saw the infamous "Not a number."

[00:19:14] **Pascal Birchler:** [00:19:14] [laughs]

[00:19:15] **Martin Splitt:** [00:19:15] In the price field of an ad, and it was bananas. So you buy this for 20 bucks, you buy this for 10 bucks, which gives us a total of "Not a number!" And I was like, "Oh..." And that's a

classical JavaScript problem where one part of the application didn't know what the data was that the other part expected and then gave it in the wrong way, and that imploded. Is there a better way to do this? Is there something that JavaScript allows you to do to avoid these kind of situations where the left hand doesn't know what the right hand's doing?

[00:19:45] **Edu Pereda:** [00:19:45] I think TypeScript has made that thing a lot better. Because you can enforce that a lot of things don't go wrong at compile time, so you don't need to wait for them. I think if you use strictly all TypeScript with all the possible checks, which sometimes is very annoying to work with because you're like, "I know this is going to work, why don't you let me compile it?" And then if instead of finding the right types, you end up using this any type that TypeScript has. That is basically an escape hatch to say, "I'm going to use JavaScript here for a bit, trust me."

[00:20:21] **Martin Splitt:** [00:20:21] [laughs]

[00:20:22] **Edu Pereda:** [00:20:22] So, as long as you don't do that, or if you do, you really know what you're doing, I think those errors are part of the past. I don't see them now, if you're using TypeScript correctly. So maybe that website that you saw is older or they don't use TypeScript, or...

[00:20:41] **Pascal Birchler:** [00:20:41] Maybe don't they don't have any tests for the UI, it could also be possible.

[00:20:46] **Martin Splitt:** [00:20:46] But Pascal said that he's using a mix of JavaScript and TypeScript, so why would you do that then. It sounds like it's a very clear need for just using TypeScript, and all the problems would be solved.

[00:20:58] **Pascal Birchler:** [00:20:58] The reason why we have this mix of JS and TS is because we started building the application in purely JavaScript. There was like-- a few years ago already, it allowed us to iterate quickly, because we didn't have to deal with any types. Also I think, the big reason was that not every team member was familiar with TypeScript, not all the tooling was at the level it is today. But I think last year was when I started migrating actually part of the code base TypeScript step by step. And that actually helped uncover a lot of bugs, as Edu mentioned. It could even be silly bugs like not passing the right arguments to some function and not getting the results that you would expect. So yeah, the type safety with TypeScript definitely helped to make our application safer and less buggy.

[00:21:53] **Martin Splitt:** [00:21:53] That is nice, so you would say TypeScript is a natural improvement for anyone building larger scale applications with JavaScript?

[00:22:02] **Pascal Birchler:** [00:22:02] I would say yes, absolutely. And it just makes sense to use types with JavaScript as well, because if you look at other programming languages... For example, I used to work a lot PHP because of WordPress, and you have types there. You have types in Java, which is probably the language you learn most at university. So why not use types with JavaScript as well?

[00:22:28] **Martin Splitt:** [00:22:28] True, true.

[00:22:30] **Edu Pereda:** [00:22:30] You know what to me was a great sign that TypeScript was the right way of improving JavaScript? When Google adopted it, Because it was created by Microsoft and Google went full on TypeScript. So it has to be good for a company like Google to go with something built by one of the biggest competitors.

[00:22:49] **Martin Splitt:** [00:22:49] True, true.

[00:22:50] **Edu Pereda:** [00:22:50] I mean, it's a good sign because we are basically choosing the right tool and not avoiding it just because it'd come from some other company. I loved TypeScript the moment I started reading about it, because it was JavaScript done right. I became a fan boy!

[00:23:04] **Martin Splitt:** [00:23:04] I think that's fine. And I really love that the power of open-source means that one company uses the stuff from another company without prejudice. I think that's a really beautiful aspect of open-source. And what has your journey been? I mean both of you have been writing JavaScript beforehand and Pascal, you also have experience with PHP probably from the time when types weren't a thing in PHP because I remember that. So, how did you feel the journey went? What were your big key moments? Was there a big "Aha!" moment in the journey from JavaScript to TypeScript? And is there anything that you would like to tell developers out there who probably haven't touched TypeScript yet?

[00:23:46] **Pascal Birchler:** [00:23:46] Oh, that's a good question, because I do remember that I looked at TypeScript and had tried it for some small projects a few years ago. And it was not where it is today. It didn't have many features and you're like, "Oh, I want to make types for this thing" but actually, it couldn't do what you wanted. So maybe you resorted to any... like why bother? I go back to JavaScript. So I would say if you have tried TypeScript as well in the past, it might be worth trying it again. If you've never tried TypeScript, I would probably start checking out the documentation, like getting started guides, because it's actually quite easy to get going with TypeScript. You can adopt it gradually for a project, maybe on a five by five basis. If you're not sure what to type, you can maybe use any... even if it's dangerous. [laughs]

[00:24:46] **Edu Pereda:** [00:24:46] But I think that's the perfect usage for it, for migrations. Internally at Google, we had-- I don't know if we still have it. There was this alias type any for migration, so that the lint would not complain, because you're not actually using any, but any for migration.

[00:25:02] [everyone laughs]

[00:25:05] **Pascal Birchler:** [00:25:05] I hope you have removed all the occurrences of that by now.

[00:25:09] **Edu Pereda:** [00:25:09] Oh! I was hunting them down. I avoid any like the plague. I don't like it in my code base.

[00:25:16] **Martin Splitt:** [00:25:16] [laughs]

[00:25:17] **Edu Pereda:** [00:25:17] So in my case, I used to work with the GWT, Google Web Toolkit, which was a way that somebody in Google invented many, many years ago to write Java and convert that into JavaScript. So it was one big attempt to add ties to JavaScript. Because you could write code in Java with some limitations and get that converted into JavaScript. We were using that in the main internal application I worked when I started in Google for years.

[00:25:48] And eventually, that thing became deprecated because basically, JavaScript became good and there was no need for this conversion from Java to JavaScript. And when I learned about TypeScript, there was this software engineer based in Munich called Martin Probst that he brought... I think he's the one that brought TypeScript to Google, and he gave a talk in the Zurich office. And I was like, "Whoa! I need to learn this." And I started. I fell in love with the language right away. So whoever has not tried it and is using JavaScript today, they should go ahead and give it a try because it's amazing. And Gary, if you're listening, you should try it as well.

[00:26:30] **Martin Splitt:** [00:26:30] [laughs] I'll report back if I can coerce or convince Google, and I'll report back if we can convince Gary to actually try it.

[00:26:45] **Edu Pereda:** [00:26:45] I know.

[00:26:46] **Martin Splitt:** [00:26:46] He loves JavaScript so much. I'm sure we can't coax him away from that. [laughs] We can try though, we can try, I'm with you on that. Excellent.

[00:26:57] So cool. And is there anything else exciting? I get the vibe that TypeScript was a huge step forward from JavaScript being a bit of a bag of surprises towards something more predictable. Is there something else that you find exciting about developing for the web these days? What about WebAssembly?

[00:27:18] **Pascal Birchler:** [00:27:18] WebAssembly, I love it. I'm not like super familiar... I have never written something and then compile it to WebAssembly, but I'm a heavy user of their tool, because WebAssembly allows us to bring, let's say C++ code and run it in the browser, and it's pretty cool. And in our use case, we use it, for example, for transcoding or compressing videos that you upload to your site, but in the browser. You don't have to do it on the server, maybe the server is not capable of doing that. Again, we're in a WordPress/PHP context, so there's no video library or whatever installed on that server. But we can do it all in the browser with JavaScript running this WebAssembly program.

[00:28:14] **Martin Splitt:** [00:28:14] We should probably explain what WebAssembly is. You already said you're doing video compression but does that mean that you can do just a very specific kind of JavaScript, or how does WebAssembly work in very rough terms, you don't have to go into the super nitty gritty details, but what's WebAssembly?

[00:28:32] **Pascal Birchler:** [00:28:32] Basically, WebAssembly is a low-level assembly language that is supported by I think all major browsers nowadays. It allows you to compile binary programs into

WebAssembly that you can then talk to via JavaScript.

[00:28:51] **Martin Splitt**: [00:28:51] Nice. So allows you to basically compile code in languages that are not supported in the browser, like C++ or Rust or whatever into a kind of code that runs in the browser as well without having to go through JavaScript. But you can call it from JavaScript, I guess.

[00:29:08] **Pascal Birchler**: [00:29:08] Exactly, yeah, you can call it from JavaScript.

[00:29:11] **Martin Splitt**: [00:29:11] That's really nice. That's really cool. So you can write parts of your program in a different language and then use that from JavaScript. That's really interesting.

[00:29:19] **Edu Pereda**: [00:29:19] Okay, so I know nothing, nothing, except from higher level what is... but it sounds like you can enhance the browser APIs and then call them from JavaScript, I guess.

[00:29:31] **Martin Splitt**: [00:29:31] Yeah, to a certain degree. I don't think you can break out of the sandbox, but I know that I wrote a C++ program that basically was a function that took two numbers and added them, which you can totally do in JavaScript, but the point was to see if I can...

[00:29:45] **Pascal Birchler**: [00:29:45] [laughs]

[00:29:46] **Martin Splitt**: [00:29:46] I know, that's ground-breaking stuff.

[00:29:48] **Edu Pereda**: [00:29:48] You can add them up without "Not a number."

[00:29:50] **Martin Splitt**: [00:29:50] I know, but we could hypothetically take our C++ robots.txt parser, make that as a function, compile that function into WebAssembly and then call that from a JavaScript interface. So you could build a web interface that uses the same source code fundamentally as the C++ tool that we built and that works in the browser, and I think that's really exciting.

[00:30:15] **Edu Pereda**: [00:30:15] Would it be like GWT, Google Web Toolkit for C++?

[00:30:19] **Pascal Birchler**: [00:30:19] [laughs]

[00:30:20] **Edu Pereda**: [00:30:20] Sort of. [laughs]

[00:30:22] **Martin Splitt**: [00:30:22] Kind of. Kind of, I think.

[00:30:26] **Pascal Birchler**: [00:30:26] Even like a note. Do you know whatever also supports WebAssembly, nowadays? So it's a really portable format, very useful.

[00:30:36] **Martin Splitt**: [00:30:36] It's basically like a little app that you can call from whatever JavaScript runtime supports WebAssembly. I think that's pretty interesting.

[00:30:42] **Pascal Birchler**: [00:30:42] And I think I also saw some examples where people basically put a JavaScript runtime into WebAssembly so you can execute JavaScript in WebAssembly. It's mind-blowing.

[00:30:54] **Martin Splitt**: [00:30:54] Okay, that's inception-level stuff. All right, before you put more of a knot into my and other people's brains, thank you so, so much for the conversation. It has been really, really exciting to hear about all the things and to also take a look into the future. And I know that we don't necessarily do that very often, but let's do one last look into our crystal ball. Do you guys think that WebAssembly will replace TypeScript then?

[00:31:21] **Edu Pereda**: [00:31:21] I have no idea, but I don't think so. Because let's introduce the caveat that I have no idea about [duh json], but from the description we just gave, it sounds like it serves a different purpose. I don't know.

[00:31:34] **Martin Splitt**: [00:31:34] True, true. That's true. Pascal, what's your take?

[00:31:37] **Pascal Birchler**: [00:31:37] I know that there are still some performance limitations with WebAssembly, because when you have to pass data back and forth between JavaScript and WebAssembly, it does a lot of overhead. And I think that some people would say that yes, WebAssembly could take over once these limitations are resolved, but I'm not sure where that would lead us.

[00:32:02] **Martin Splitt:** [00:32:02] Hmm, okay.

[00:32:03] **Edu Pereda:** [00:32:03] But if you could use just WebAssembly, then you don't have that problem of passing the data between JavaScript and WebAssembly. I don't know, could you have, in the same way as you include a JavaScript file from an HTML, could you include something in WebAssembly as well?

[00:32:18] **Martin Splitt:** [00:32:18] I think that's not how that works because as far as I understand, WebAssembly runs in its own virtual machine, and you would still have to, once you want to do something with the UI for instance, you would have to pass out of that virtual machine into the browser, and that's probably where the data transfer bottleneck comes in, I am guessing. But maybe I'm wrong. I think we can wrap that up as the definite maybe.

[00:32:46] [guests laugh]

[00:32:48] **Martin Splitt:** [00:32:48] And let people see what the future holds if we have been right, wrong or just in the middle. But before I send everybody off, Pascal and Edu, where can people find you?

[00:33:00] **Edu Pereda:** [00:33:00] I'm at home.

[00:33:01] [guests laugh]

[00:33:03] **Martin Splitt:** [00:33:03] I mean online. How can people... Give them their specific address so they can bring you chocolate, yes. But are you on Twitter, are you on LinkedIn?

[00:33:13] **Pascal Birchler:** [00:33:13] My Twitter handle is @epere4. So e from Edu, pere from Pereda and the number 4.

[00:33:19] **Martin Splitt:** [00:33:19] [laughs]

[00:33:20] **Pascal Birchler:** [00:33:20] And there, you can find me.

[00:33:21] **Martin Splitt:** [00:33:21] Okay, excellent. Pascal, how about you?

[00:33:23] **Pascal Birchler:** [00:33:23] My Twitter handle is @swisspidy or swiss-spidy. It's like swiss and then spidy with three "s" in the middle.

[00:33:31] **Martin Splitt:** [00:33:31] Excellent. You two, it has been such a great time. I do hope that you enjoyed it as much as I did. I think this has been really interesting, and this could go on for hours, but let's wrap it up here.

[00:33:42] ♪ [music] ♪

[00:33:46] **Martin Splitt:** [00:33:46] We've been having fun with these podcast episodes and we hope that you, the listener, have found them both entertaining and insightful too. Feel free to drop us a note on Twitter at @googlesearchc, or chat with us at one of the next upcoming events that we go to if you have any thoughts. And, of course, don't forget to like and subscribe. Thank you so much and goodbye!

[00:34:09] **Edu Pereda:** [00:34:09] Bye!

[00:34:09] **Pascal Birchler:** [00:34:09] Bye!

[00:34:13] ♪ [music ends] ♪