EPISODE 797

[INTRODUCTION]

[0:00:00.3] JM: Coding in the browser has been attempted several times in the last decade. Building a full development environment in the browser has numerous technical challenges. How does the code execute safely? How do you fit all the requirements of a development environment into a browser window? How do you get users to switch from their normal IDE, the interactive development environment to your new web-based environment.

CodeSandbox is an online code editor created by Ives van Hoorne and Bas Buursma. CodeSandbox allows users to program and run applications in the browser. It's a full development platform that allows users to install NPM modules, run their code and share their applications with other users.

The engineering problems within CodeSandbox are not simple. Building a web-based IDE is complicated, but CodeSandbox is also an exciting project because it lowers the barrier to entry for many newer programmers. The development experience for a new programmer is still a difficult on-ramp.

If you're an experienced developer, you have a workflow that you're comfortable with. It might involve Vim, or Emacs, or JetBrains IDEs or Eclipse, but newer developers can find these development environments confusing and hard to get started with. The development environments of today are integrated with build tools and git repositories and deployment platforms. This could be overwhelming for a newer developer.

CodeSandbox is a very visual tool and it makes it especially useful for new developers who learn through seeing examples running live in the browser quickly. CodeSandbox is also used by web developers who do have experience, but who want a modern shareable form of developing software. It's a renaissance in online web development right now with things like git workflows and new development platforms like Netlify and ZEIT and GraphQL middleware. We've done a lot of shows about this new form of somewhat serverless web development and CodeSandbox fits really nicely into that paradigm shift that's going.

Ives and Bas join the show to talk about the motivation for CodeSandbox and the engineering challenges that they have solved. I really enjoyed this conversation and I think you will too. We have a couple events coming up. April 3rd and April 6th. On April 3rd we have a meet up with Haseeb Qureshi at CloudFlare in San Francisco. We're going to be sitting down for a conversation with Haseeb about cryptocurrencies and venture capital and software engineering.

Haseeb is a good friend of mine and somebody who has been a frequently requested repeat guest on Software Engineering Daily. Then on April 6th, we have a hackathon for FindCollabs, the company that I'm building. You can find out about that by going to findcollabs.com/hackathon, or softwareengineeringdaily.com/hackathon. There's a $5,000 prize purse. To enter, you just have to add a project to FindCollabs. That project can be open source, it can be an art project, it can be a music project. Ideally, you'll find collaborators to help you build that project and to have fun working with.

The in-person meetup for FindCollabs will be April 6th at App Academy. We'll have some food. We'll hang out. We'll have a great time. I hope to see you there. If you do want to come, sign up for one of these events by going to softwareengineeringdaily.com/meetup, or softwareengineeringdaily.com/hackathon. The space for these events will probably run out, so I recommend checking them out soon. Thank you.

[SPONSOR MESSAGE]

[0:04:05.3] JM: Testing a mobile app is not easy. I know this from experience working on the SE Daily mobile application. We have an iOS client and an Android client and we get bug reports all the time from users that are on operating systems that we did not test. People have old iPhones, there are a thousand different versions of Android. With such a fragmented ecosystem, it's easy for a bug to occur in a system that you didn't test.

Bitbar is a platform for mobile app testing. If you've struggled to get the continuous delivery in your mobile application, check out bitbar.com/sedaily and get a free month of mobile app testing. Bitbar tests your app on real devices, no emulators, no virtual environments. Bitbar has real Android and iOS devices and the Bitbar testing tools integrate with Jenkins, Travis CI and other continuous integration tools.

Check out bitbar.com/sedaily and get a free month of unlimited mobile app testing. Bitbar also has an automated test bot, which is great for exploratory testing without writing a single line of code. You have a mobile app that your customers depend on and you need to test the target devices before your application updates rollout.

Go to bitbar.com/sedaily and find out more about Bitbar. You get a free month of mobile application testing on real devices and that's pretty useful. You can get that deal by going to bitbar.com/sedaily, get real testing on real devices, get help from the automated test bots so that you have some exploratory testing without writing any code. Thanks to Bitbar. You can check out bitbar.com/sedaily to get that free month and to support Software Engineering Daily.

[INTERVIEW]

[0:06:15.6] JM: Ives van Hoorne and Bas Buursma, you guys are the creators of CodeSandbox. Welcome to Software Engineering Daily.

[0:06:23.3] IvH: Hello. Thank you for having us.

[0:06:24.6] JM: CodeSandbox is a web-based IDE, or online code editor. We can talk about CodeSandbox eventually. First, let's just talk about the question of web-based development environments. Why do I need a code editor on the web? Why not just use my code editor that is local?

[0:06:47.2] IvH: That's a very good question. That's one that's we've been asked a lot before. The good thing about having an online web code editor is that you can – when you have for example a problem, you can easily share the editor with someone else and they can work with you, so there's a collaboration aspect. You also put the code editor away from machines. You can have multiple machines with the same environment.

Let's say you're in a different location, or you don't have access to your main machine, then you can still continue working because you can go to the website and start working from where you left off. Then you also have the development environments itself, like the server that runs your code; that one is external as well. You're also not tied to your system requirements if you need some development server that runs on 4 gigabytes of RAM. You don't need to use those 4 gigabytes of RAM on your own PC. There are a bunch of things that make it better to use an online code editor.

[0:07:44.7] JM: Why is it hard to build an online IDE? Because I completely agree with you that there are a lot of benefits to having one, but for the most part people are still using their local IDEs.

[0:07:58.1] IvH: Yeah. There are a lot of answers to this one. I think the main thing with online code editors is that they need to be reliable. It's quite recent that we have a lot of connectivity that we can do online editing. For example, the train, the network is getting better. If for example, a service goes down then you also lose your activity. This is a reason that lots of people are very wary of using online code editors. It's becoming more popular to use online tools for example for designing. Sketch recently announced that they are going to make the design tool on the web. The same for Figma is already on the web.

People also do text editing with for example, Google Docs, or Microsoft Office on the web. It's only since I think a couple of years that there are code editors that are completely based on web technologies, like Microsoft Visual Studio code. Lots of aspects have recently changed and more and more people are starting to use an online code editor for editing code.

[0:09:02.2] JM: We have some past examples of online IDEs. There's Cloud9, there's JSFiddle. What were the approaches of the online IDEs of the past?

[0:09:15.2] IvH: Yeah. Cloud9 is also Dutch, so that's also a really cool example.

[0:09:20.1] JM: By the way, I guess I shouldn't say of the past, because people are still using these today.

[0:09:24.5] IvH: Yeah, that's true. JSFiddle and Cloud9 are quite different in what they aim to achieve. For example, Cloud9 is aiming to replace your development environment, so that you do your primary development of all your services, or your code on Cloud9. While JSFiddle is more like a companion application, where you do write code in it, but it's mostly to either prototype something quickly, or it's to share the code with someone else, so that they can also look at it, like for example, bug reports are examples.

With CodeSandbox, we are trying to be more in the middle of those two. When I started with CodeSandbox, I found that I couldn't do as much as I wanted with JSFiddle or CodePen when I wanted to share an example. I also didn't have my laptop with me, so I wasn't able to use to see someone's code, or create code when I had question for my co-workers. That's one more of the idea of CodeSandbox game where it is useful for creating small applications, for creating bug reports, for examples. It's also used for workshops and job interviews, but it isn't aiming to replace the primary development environment yet, because that's a tricky one to handle.

[0:10:35.5] JM: Explain what CodeSandbox does.

[0:10:38.8] IvH: That might be a good one to explain. CodeSandbox is an online code editor and it's focused on web application development. At first when CodeSandbox came out, it was only focused on React application development, so people could share their React examples, or when they had a library they could share those examples with React to others. That's also our main focus. We want to have web application developments central in CodeSandbox, so that we can make that experience perfect.

We focus right now on that people can easily prototype and create examples, or bug reports to other people, because that's something that's becoming more and more a usage pattern for developers. For example, if you're a library author and you want to show an example of your library to someone else, then you can use CodeSandbox for that. Or if you want to quickly prototype something, or do some pair programming on something that you want to try out or test a function, then you can use CodeSandbox for that as well. Or if you want to create a small service, like a microservice then CodeSandbox would also be a good fit for that.

The main focus of us is to be focused on web application development, so we can for example, provide deployment services like now, or from site, or Netlify deployments. We are trying to make it effortless to start a new project and work on it.

[0:12:02.6] JM: Speaking more abstractly, what are the goals of CodeSandbox both as a project and as a company?

[0:12:10.7] IvH: When we started CodeSandbox, it became popular pretty quickly. We put out three core values for us to follow, so that when we had a feature request, we are arguing by these three values. The first value is lower the learning curve. When people start with web development, or with JavaScript development there is a huge amount of things you need to do before you can get started. You need to set up your terminal, code edits, you need to learn the terminal, you need to learn about package management like NPM or yarn. Then you need to set the whole project up.

With an editor that's already pre-configured with pre-configured projects in the browser, you can just go to the project and start editing text before you can see a change life without setting up anything like a development server. That's a really good advantage, because you will have this feeling that you're already doing something useful before you have to set this all up and when you think that you are actually going to do it more, then you can set up the whole environment.

The second value is that we want to make it very easy to share and discover project from others. When you create something on CodeSandbox, you can share it with the URL and people can open it in their own editor, they can fork it. It's a bit a very lightweight github, where people create projects and it's easily to fork and you can just continue working on it.

With the discovery part, we want to – we have a search. If you for example need help with the dependency, then you can search on CodeSandbox and filter on that dependency. You can see all the projects that use these dependencies. You can see that's this big knowledgebase of things that people create. The last value is that we want to make it easy for people to create applications and services. This is more close to the local editor experience.

We think that CodeSandbox should feel like a local code editor and it should work like a local code editor. Because if you have something that behaves differently, then you would have to learn two environments. The best case scenario is that if you go to CodeSandbox, it works exactly like your local code editor, so you don't have to learn a new way of working. We value this by everything that we do. Every feature that we build, or every change that we make to the website is put to test against these three values. If none of those values improve, then we decide to not build this feature. If one of these value is improved then we decide to build it.

[0:14:33.3] JM: One element of CodeSandbox I find appealing is the fact that this is something that appeals to both entry-level developers and to sophisticated developers. On the sophisticated developer front, that makes complete sense. You're a sophisticated developer and it would make sense, like everything you're describing about WYSIWYG editor, but just being able to develop in the same environment that I do everything else in, which is the browser and being able to have this really easy package installation experience and all the other things that you get out of CodeSandbox. It’s really quite a cool product. I encourage people to just check it out, because if you see it, you will understand why this is such a cool and popular project.

On the introductory developer side of things too, you seem to have a lot of empathy with people who are getting started developing software. I've seen this firsthand and I've heard about it a lot, like talking to Quincy from freeCodeCamp, why people drop off when they're starting to learn programming? I think a lot of it has to do with if it takes a long time to get to the moment where you feel some dopamine and you feel an element of excitement, like you've actually built something, or you've actually tweaked something and you understand why you tweaked it and why I had a cool outcome, if you don't get to that dopamine, then you're never going to become a programmer. You have to get to that initial – it used to be the hello world experience, but now the bar has been raised it seems. Is there some element? Did you have a bad experience learning to code or something, where –Is that informing your empathy for the inexperienced developer?

[0:16:17.7] IvH: Yeah, that hits pretty close to home. For me, it was – initially, when I try to do development, I stopped with it because I didn't really like it. I didn't get to the dopamine rush. Later when I picked it up because I wanted to try something new again, I got the dopamine rush and now for me development that's what excites me, this dopamine rush of making something work. If development is about this dopamine rush, then you should experience it as soon as possible, because based on that you can judge if you are going to like this or not.

This is something that I also notice with other students, so we still live on the university and there are lots of students here getting started with web development. If they have to set up this whole environment, they might drop off sooner because that's really the boring part. If my whole job would be to set up these environments for as a full-time job and I know that some people like it, but I'm not a big fan of it, then I wouldn't be interested in programming, but it's about building this stuff. That if you focus on actually building stuff, that's much more exciting.

If you can get this building phase as soon as possible, that is the best-case scenario. Yeah, it's already cool that if you can just edit some text and send hello world, that already gives this small dopamine rush.

[0:17:41.1] BB: It's the preview window that's just next to the editor that's also really nice to see, that when you open CodeSandbox and you see an example, you have a visual representation next to it that's executed.

[0:17:51.8] JM: It also tells you where you screwed something up. If you mess up something, it's not just – it doesn't just give you – well, depending on what the context is, but it'll give you the error message, but it also gives you suggestions and you can just one-click and have your code fixed in some context, which is pretty cool. If you forgot a semicolon somewhere, it won't just say you forgot your semicolon, it'll say click here and we'll just automatically fill in the missing semicolon for it, which is pretty nice.

[0:18:24.0] IvH: Yeah, exactly. That's the cool thing about having ownership over the whole environment, because our editor knows about the files, it knows about the code, how it executes the code. We can take advantage of this. One example is for example, because we focus so much on web development and initially on React development, like the early days there was a very specific React error that lots of people had. When they forgot to name a file with an extension, it would be imported as a string. If people –

[0:18:50.4] JM: It's like without .js.

[0:18:51.9] IvH: Yeah, exactly. Without .js. Then people expected it would be a component if it exports a component, but it's a string. React used to give a very strange error message, a very ambiguous error message of that it doesn't understand what it is as a component and people were having this issue. They send it in the discord of CodeSandbox that they had this also during job interviews and we can easily detect this.

If we see an import that's without an extension and it's used as a component, we can already see oh, yeah, this is should be a React component. We caught the error message and transformed it and added a button. If you click that button, it renames the component to .js, so it renames the file to .js. Those very small things can be very frustrating, but we can already catch them pretty easily.

[SPONSOR MESSAGE]

[0:19:49.3] JM: When I'm building a new product, G2i is the company that I call on to help me find a developer who can build the first version of my product. G2i is a hiring platform run by engineers that matches you with React, React Native, GraphQL and mobile engineers who you can trust.

Whether you are a new company building your first product like me, or an established company that wants additional engineering help, G2i has the talent that you need to accomplish your goals. Go to softwareengineeringdaily.com/g2i to learn more about what G2i has to offer.

We've also done several shows with the people who run G2i, Gabe Greenberg and the rest of his team. These are engineers who know about the React ecosystem, about the mobile ecosystem, about GraphQL, React Native. They know their stuff and they run a great organization.

In my personal experience, G2i has linked me up with experienced engineers that can fit my budget. The G2i staff are friendly and easy to work with. They know how product development works. They can help you find the perfect engineer for your stack and you can go to softwareengineeringdaily.com/g2i to learn more about G2i. Thank you to G2i for being a great supporter of Software Engineering Daily, both as listeners and also as people who have contributed code that have helped me out in my projects.

If you want to get some additional help for your engineering projects, go to softwareengineeringdaily.com/g2i.

[INTERVIEW CONTINUED]

[0:21:41.8] JM: I was watching a presentation you had given and one of the things that also made me think you might have had a bad experience programming early on was you took a subtle jab at learning Java in university, which I can completely relate to, because in college, most of our curriculum was focused around Java. I got pretty good at Java and all due respect to Java, Java has tons of benefits. The JVM turns out to be a really, really interesting ecosystem with millions of applications running and it's very useful.

For newer developers, for students, what's appealing, what's fun is the JavaScript world. It's sad that because of the way that universities work, I think they're caught in – they're caught in the Java, C++, maybe Python world and they're just completely missing out that JavaScript is the way that you should be learning programming now.

[0:22:45.9] IvH: Yeah, I agree. Java, I think my first language was Ruby, but then I started learning Java due to – well, it's for Minecraft actually. Java has a bunch of boilerplate. There are lots of – if you want to explain to someone how a piece of Java code works and they have never programmed before, then you have to – it's almost impossible to explain it if you don't want to skip some fundamentals, because there's so much boilerplate with Java.

I think JavaScript is actually very nice balance between simplicity and what you can do with it. Because there are no types, you don't have to explain the type system initially to someone and you can gradually start introducing all those different theories in the programming world. That's super nice. For example, if someone wants to learn how to write hello world with JavaScript, it's a one-liner. For Java, it would require some boilerplate too. I think that JavaScript is one of – is at least a very good language to start with. Yeah, and at our university we still start with the Java. That was my little stab at the Java boost room.

[0:23:56.5] JM: Yeah. I mean, I just remember my hello world experience with Java and just starting with public static void main string –

[0:24:04.1] IvH: Exactly.

[0:24:05.1] JM: - brackets, args, open brace, return, system .out .currentloop, it’s like, I know it from memory because I swear, I spent days trying to understand, so what's a static? Like static? What? What does that mean?

[0:24:25.1] IvH: Yeah, exactly.

[0:24:27.9] JM: The thing about the JavaScript world is there was in recent memory and I think, you could argue this still exists today, there is a steep learning curve to understanding what's actually going on in the JavaScript world, because you have – there's React and then how does that work. There's NPM modules and how do these things work. Then it's like, okay, am I using typescript? What is my build pipeline? What is this web pack thing? What's your feeling on the learning curve of JavaScript today?

[0:24:58.7] IvH: Yeah, that part is a huge learning curve. I think that the JavaScript world is not there yet with the tooling. Although, it is so good to see that there are so many people taking a lot of effort to make this better. When you see for example new butlers pop-up that have zero configuration, or very simple libraries that do the rendering for you. Still, it's a steep learning curve and that's one of the reasons that CodeSandbox was started, because we try to make as many choices as possible for you. Library called create-react-app was one of the inspiring factors for CodeSandbox. With create-react-app, you can start a project by running a single command and it will scaffold your projects, but it will hide all the configuration. That's on purpose, because they want to have simplicity.

For example, if someone says, “I'm using create-react-app,” then the person you're talking to if they know create-react-app, they already know everything about the build system and how it works. There's a lot of power to that. If you want to escape from this CLI, then you can call create-react-app eject and then you will be actually able to see the configuration files.

We wanted to do something similar for CodeSandbox, but then with the whole build tooling with editor setup and all these dependencies. When you start a new project, we already made a bunch of choices for you. For example, we decided that you use create-react-app, we decide that – you don't have to choose between MPM, or yarn. Even the package managers are – there are four different package managers now in the JavaScript ecosystem, and you don't have to choose an editor, or how to run the code. All those choices are made for you. With that, we hope to make it, so that you don't have to think about all these things. You can think about them later.

I find it pretty interesting to compare all these tools. If you actually want to build something, then all these choice gets in the way of getting started. Yeah, I agree with the JavaScript community, it's a good thing that there are so many tools, because tools are getting better. They are competitive, so they try to stay on top of each other. When you're getting started, it can be overwhelming. Then with tools like CodeSandbox, JSFiddle is also a good tool actually and code bin and JSBin as well. They already make choices for you and that makes it easier for you to learn.

[0:27:16.0] JM: How do developers use CodeSandbox today? Are they using it as their actual IDE?

[0:27:23.4] IvH: Yeah. There's a wide range of use cases. Someone built a Redis server on CodeSandbox. Most people use CodeSandbox either for documentation, bug reports to a prototype, to with other people. It's used for job interviews, it's used for workshops, it's also used for creating small applications. For example, if you have a back office tool that you need to make. We've seen that people use it for big applications like main developments, but it's not used that much for this use case yet. That's also because CodeSandbox has been focusing mostly on the smaller applications, but we want to make it possible to also build bigger applications with CodeSandbox and hope to see bigger things built on CodeSandbox.

Yeah, I think the core value for CodeSandbox is that’s it's an online code editor that's very easily shareable. You can just share a link with someone and they will see the exact same environment that you build something in. There are lots of different use cases that you can tie to it. I think we haven't found all the use cases yet. Job interviews was something that I never thought of with CodeSandbox, or bug reports was also something that I didn't think of when starting with CodeSandbox. Yeah, that's how the function is.

[0:28:38.3] JM: What? Bug reports? What's the bug reports use case?

[0:28:40.5] IvH: Oh, that's a really funny one. If you're an open source author, or you have a library and people send in a bug report, like when I give these parameters, this function doesn't work, or it throws an error, then it's much more useful to also give a reproducible. Previously, this was done using GitHub, like you give a GitHub repository, or you give a snippet of code that they have to run in their own editor to test it.

Nowadays, lots of open source library authors mostly in the React community, they have an issue template where they put in a CodeSandbox example and they ask people to put the bug in the CodeSandbox example and then link it in the issue, so that when library authors can just open the sandbox and see the error right away.

[0:29:25.1] JM: Interesting. What's the architecture of CodeSandbox? I open it up and CodeSandboxe is in my browser, what's going on under the hood?

[0:29:36.4] IvH: CodeSandbox consists of multiple – when I started with CodeSandbox, it was simple. It had React front-end and the backend was in Phoenix and Elixir. We have a PostgreSQL database and Redis database for caching. That's still very similar to how it is now. The front-end is completely in React. Then we have the backend, which just serves as an API and we have a database PostgreSQL and Redis. We now also have some microservices, node microservices.

The interesting thing that I think is pretty cool about CodeSandbox is that for the majority of the projects built on CodeSandbox, they are executed in the browser. We have built something that's very similar to web pack, not as advanced as web pack, but it works good enough for us. It's a bundler, but it's optimized to work in the browser. For example, for transpiling we optimize it by running all the translations in web worker, so we can do it in parallel. Its guest using a serviceworker, so it works offline. Whenever you open a project in CodeSandbox, you actually execute the code instead of us.

This was super useful, because when we started with CodeSandbox we were just students, so we didn't – we are not able to run all the code on servers. That would be super expensive. For the first year of CodeSandbox, we've run everything on a $40 fee PS at Fulcher. At some point, we had I don't know, I think we had 300,000 visitors in a month and we were still running everything on a $40 server, because we let the user execute all the code in the browser. It even got to the point where we also let the user generate the cache of a sandbox. Whenever a sandbox is executed, we generate a cache of all the computations that have been made and all the translations, all the CPU intensive work. Then we let the user upload this cache to our servers.

This way, we even let users generate and work on our cache. We found it pretty funny. It feels a bit like Bitcoin mining, but that's also not the same thing. It allowed us to keep our cost very low. Since November, we also have a different project on CodeSandbox called containers. There we run the code on Docker containers and that's for some templates applicable.

[0:31:52.2] JM: That's pretty cool. You mentioned service workers and web workers. I think this is actually a topic that we have not covered at all on Software Engineering Daily. Can you explain what a service worker and a web worker are?

[0:32:05.4] IvH: Yeah. The worker concept is a way for – JavaScript always runs in one thread. The worker concept is the answer of JavaScript for allowing – run code to run simultaneously at the same time. A web worker is a new process. You can see it as a new process that spawns and then executes your codes. It has a messaging protocol, so you can from the main thread, you can send messages to it. For example, transpile this code, and then it can send a message back with the answer. Because it runs off the main thread, the UI won't stall when it's doing heavy computation work, so that's a good advantage.

Then the service worker is very similar to a web worker, but it's much more powerful. Service worker has a lot of rights that web workers can't have. One thing for example, as a website you can register a service worker and you have one service worker per website. For example, if you have five tabs open, they share the same service worker. The service worker can intercept network requests for example and it can do a bunch of other stuff. It can show notifications. It can run in the background when the website is not open to do computation.

I always found the network intercepting very interesting, because with this you can make websites work offline. For example, if you don't have a network connection or you're offline, then with a service worker you can still go to the website and the service worker can handle the network requests and give a response back that is for example, cached or something that the service worker already knows. This way, it doesn't seem like the website is offline.

For example, with CodeSandbox when you have a project and you run it once on your machine, whenever you're offline you can still run it, because our service worker cached all the files that are used in the initial run and it will give back those files if it detects that you're offline.

[0:33:56.9] JM: I think on the backend, you mentioned it's Elixir?

[0:33:59.5] IvH: Yup, it’s Elixir. CodeSandbox started as a learning project for me. It was more of an escape from lectures at the university. I always wanted to learn Elixir, so this sounded the perfect fit to learn Elixir. I have to admit, I had to rewrite it three times before I was able to rock the functional programming paradigm completely. It's been super, super powerful until now.

For example, we have this functionality in CodeSandbox that allows you to edit code at the same time that you can – like Google Docs that you can see this cursor moving. This was all implemented in Elixir. It handles it super well. We were able to have I think 200 different users connected to live sessions at the same time. The server was just able to handle it. This was still around the time where we had this $40 fee PS. I was very impressed with how Elixir handles this. Elixir is based on Erlang, so it runs on the same VM as Erlang. Erlang is designed to be very concurrent.

Erlang has this very cool concept, where everything that – all computations are in separate processes, a bit like web workers, but these processes are super lightweight. When you do a computation, you can start a process and it has a messaging API and it will do the computations for you. The advantage is for example, if process crashes, then it can just – the supervisor, which handles the processes can just spawn a new process and it will behave normal. It's very isolated. It's also horizontally scalable.

In the case of us, if we have 200 people connected at the same time to our server, then it just spawns 200 processes that handle the incoming connections and then pass them to new processes that do the computation. This is super cool, because our system in theory would be infinitely scalable, because server is just a bunch of processes. If you add a new server and you make it think they are the same server, it's just one big pool of processes that can communicate with each other. Yeah, I'm super excited about this concept and it helped us already quite a bit.

[0:36:10.1] JM: CodeSandbox, is it entirely open source?

[0:36:14.2] IvH: No. For CodeSandbox, everything is open source, except for our API server. That's actually the Phoenix server and the Elixir part and our container structure. We're considering of open sourcing the container structure. Yeah, with CodeSandbox, we had the idea, like we wanted to keep the option of on-premise open when we started with CodeSandbox, so we decided not to open source the API server.

A lot has changed in the meantime, so I think we need to revisit this at some point. All the other stuff is open source, so the clients and that's actually where most of the code is because we try to do a lots of computation in the browser. All the microservices, or the node microservices are open source.

[0:36:57.2] JM: What's the functionality for the API server? Is that just when people are creating new CodeSandboxes and I guess, the database of all the different CodeSandboxes that people have created?

[0:37:09.2] IvH: Yeah, yeah. It functions as a database. When people create a new sandbox, it's stored using our API server in the PostgreSQL database. When people fork, it’s – you can see it as this data store. The API server is very light in the sense that it's like this gateway to the database. It does have some logic behind the live collaboration thing, but that's the heaviest part of the API server.

[0:37:33.3] JM: as a business, you have all these CodeSandboxes that people are creating in your database. What if a network affects to having all those CodeSandboxes?

[0:37:44.6] IvH: How do you mean with network, like bandwidth?

[0:37:47.7] JM: No. More like as a business. I mean, there must be some advantage to aggregating all these different sandboxes that people are playing around in. I guess more broadly, maybe you could just talk about what the vision is for the business.

[0:38:01.7] IvH: Yeah. For example, with saving all these sandboxes, we haven't done it enough yet, but we want to make it more easily for people to find new sandboxes. You can see some usage patterns. For example, filter by dependencies, that's something that we already do. Also, filter for example by version. As a company, we want to make it easier for people to – mostly for open source. We want to focus a lot on open source. We want it easier for people to share their documentation and examples.

I already described earlier the scenario of bug reports and documentation. We want to improve this with CodeSandbox by making the editor more configurable. For example, some websites for example the Babel REPL, they are building their own code editor to show transpiled code, because that's what Babel does, it transpiles code from JavaScript to a different form of JavaScript, so to say. They have this live editor where you can play with it.

I think that CodeSandbox would for example, be a very good fit for this demo behavior. The editor is not optimized to show transpiled code. For example in the preview, you don't see a transpiled code. It would be very cool if people have more configuration possibilities in the editor itself, they can for example set the linting rules, or they can say what preview is shown as default, they can install custom dev tools. That way, they can eat more easily show to other developers their APIs. That is something that we want to make much better as a business.

We also want to make it more easily for prototyping and their other use cases for example, job interviews is something that was not really – people use it for job interviews, but it's not really optimized for job interviews. By having a more configurable editor where you can for example, at your own deployment target, or you can add your own small UI that shows a timer, these small changes so that people don't only edit the code that they want to show, but they can also edit the editor experience that they want to show to others. I think that would be very valuable for all the different use cases that are already done by CodeSandbox, but also for creating new use cases.

[SPONSOR MESSAGE]

[0:40:21.7] JM: Logi Analytics is an embedded business intelligence tool. It allows you to make dashboards and reports embedded in your application. Create, deploy and constantly improve your analytic applications that engage users and drive revenue.

You focus on building at the best applications for your users, while Logi gets you there faster and keeps you competitive. Logi Analytics is used by over 1,800 teams, including Verizon, Cisco, GoDaddy and JPMorgan Chase.

Check it out by going to L-O-G-Ianalytics.com/datascience. That's logianalytics.com/datascience. Logi can be used to maintain your brand while keeping a consistent, familiar and branded user interface, so that your users don't feel they're out of place. It's an embedded analytics tool. You can extend your application with advanced API, so you can create custom experiences for all your users and you can deliver a platform that's tailored to meet specific customer needs. You could do all that with Logi Analytics. Logianalytics.com/datascience to find out more.

Thank you to Logi Analytics.

[INTERVIEW CONTINUED]

[0:41:49.2] JM: To touch a little more on the front-end architecture, so if you open up a CodeSandbox, you have as Bas said, you have the code, but you also have basically the output of the code, with the IDE alongside the rendering of the website that you're building in the IDE. I think that this requires. You talk about iframes. Basically, these two sections are in iframes and they're passing messages between each other. Can you just talk a little bit about the interaction on the browser between your editing space and what you see is what you get a space where it's rendering what your code does?

[0:42:37.1] IvH: Yeah, yeah. All the preview, everything that you create, the code is executed on a different domain, because if we would execute the code on the same domain, then the code that's written by the user would be able to access the cookies of the root domain and do potentially bad things with them.

The code is executed in an iframe under a different subdomain. We communicate with this iframe using post message. You can use post message to send messages to the iframe, the iframe can send messages back to the main window. That's also how we execute the code. In the client side evaluation, so that's how we call the buntler that runs in the browser. For that scenario, we send all the files of the projects to the buntler and we give the entry point and the dependencies. The buntler transpiles all this code and it installs the dependencies. Then it executes the code. Then it sends back either an error with the line and the column number, or it sends back that it's successfully transpiled.

There is a small API between the preview and the editor that allows us to control the editor from the preview. Not in a very substantial way. It's not very powerful, but it allows us to do things like with these suggestions where something is wrong with a component where it doesn't have an extension, these things can be done from the preview, then we can send the message or rename this file to this file.

[0:43:59.9] JM: You've talked a little bit about the opportunities of using web assembly to improve the experience of somebody who is using CodeSandbox. What opportunities do you see for web assembly?

[0:44:13.3] IvH: Oh, yeah. There is a wide range of stuff that we can do with web assembly. I found the story from Figma super interesting, where they had a bunch of performance improvements by writing their design tool in web assembly in parts. We already use web assembly for our syntax highlighting. CodeSandbox uses the facial studio code editor as the core editor, so that it's very similar to your local environment. We wanted to use the same syntax highlighting as special studio code, but they use rag expression library for reading the syntax files that's written in C. We try to convert it to JavaScript, but that was much slower. Then we created a web assembly version of the C library for reg expressions.

That was actually pretty fast. This way, we were able to use the same syntax highlighting as visual studio code, but in the browser. This is a very good example of how web assembly can be very powerful. You have a lot more control over memory management, over how code is executed. If you need to have a high-performing part of the code editor, then you can write it either in C, C++, Rust, create a web assembly from it, version from it and execute that.

I think we should explore and we can explore many more different versions of this. For example, maybe it would be possible in the future to write C code in CodeSandbox and we still run it in the browser, but in a web assembly version. That would be very interesting too.

[0:45:42.6] JM: Maybe even Java code.

[0:45:45.3] IvH: We go back to the Java. That would actually be pretty interesting.

[0:45:50.4] JM: What's the five-year vision for the company?

[0:45:52.7] IvH: The five-year vision is that people will use CodeSandbox actually as a default way of development. Right now, people still use their local code editor to do development. I understand that it's safer, but we want to build the functionality in CodeSandbox where we have – it's so much better to develop in CodeSandbox, it's much better focused, much better streamlined, it's much easier to share.

Source control is done very easily in CodeSandbox that it's more compelling to build something in CodeSandbox, because then you can easily work together with other people on it. That's our long-term vision and that's something that we are aiming to achieve in a very gradual ways. Like I said, that we are not focusing on being the primary editor right now, we are focusing on making it easier for the other use case of developers, because developers they don't only write code for applications, they write code for a lot of things. Like for simple prototyping, or for services, for bug reports, for quickly sharing ideas with others. We want to make those use cases, we want to fix those first.

[0:47:02.4] JM: Do you see other opportunities to improve developer experience with dashboards, or heads-up displays, or better visualizations, or just are there better ways to improve the developer experience with just a better UX? Because I felt when I was looking at CodeSandbox I was like, this is so different – such a different experience than – it's been a while since I've written code, but it's such a much friendlier user experience than a lot of the other development tools I've used in the past.

[0:47:36.4] IvH: Well, thank you very much. Yeah, it's –

[0:47:39.5] JM: Not that I used it extensively, so I'm – I'm definitely stroking your ego here, but I really did think it was beautiful. What are the other opportunities to improve developer experience?

[0:47:49.8] IvH: Yeah. People always are very visual-minded. We're a visual-minded bunch of people so to say. I think that we can make it far easier for people to see for example, if you have test results, it's far easier to see the test results in a UI. Or if you have an overview of how big your bundle size is, if you see the numbers, it doesn't say as much as if you see it in a visual way, for example on comparison of how much of your bundle is dependencies and your own code. It's much easier to see this comparison in a visual way.

I think that we can make it – make the whole editor experience far more visible, that you can very easily see the test results, that it automatically runs the tests when you change stuff, but then also shows it in a good UI. Also around the editor, the whole experience around the editor itself for example, if you have with a dashboard, you can see a list of projects that you have and you have a screenshot of every project, so of the preview of every project, so you can easily see which project belongs to which code.

That's something that developers are very terminal-minded and I think that we can make more tools that are more visible. It's I said once, interfacifize everything and I think that's what we've been trying to do with CodeSandbox a while.

[0:49:03.7] JM: Yeah. I remember in college, the cool way to do things was in sublime text. I was like, “Yeah.” I mean, I'm a fan of very basic text editors. Or I guess, Vim; people still really like Vim. Some people are Vim – I never got into that. I was like, this is just onboarding - onboarding is too steep. I guess, I'm not cool enough. You need some Vim plugin perhaps, or a Vim experience. Actually, there's probably nobody in that Venn diagram overlap, people that both want Vim and CodeSandbox.

[0:49:38.5] IvH: Well, we've seen some Vim support requests. We're deploying that on – actually this Monday, we're planning on a very big release this Monday with VS code extension supports and such. The Vim extension is also included.

[0:49:54.1] JM: If you have VS code extension support, then that means the same tweaks that people can make to their visual studio code environment, they can make to CodeSandbox?

[0:50:02.9] IvH: Yeah, yeah. That's what it comes down to. Initially, I wanted to support VS code extensions and I was thinking of building a bridge between the VS code API and the CodeSandbox API to make the extensions work better, but then I saw that it’s so much work that I took a stab at implementing more parts of VS code directly in CodeSandbox. That actually worked very well. This update, we're going to make VS codes the default editor in CodeSandbox for our core editing and it will allow for extensions. Then also for the customization that you have in VS code.

[0:50:37.3] JM: Very cool. As we begin to wrap up, I'd like to know a little bit about your division of labor and the dynamics of co-founders. What's the product development cycle like between you two?

[0:50:52.1] IvH: Bas and I, we've been working on – together on projects since we were 12, I think when we met in high school.

[0:50:58.9] BB: Yeah, a long time.

[0:51:01.0] IvH: Yeah. Normally it goes like this when we have an idea, we live in student dorms at the university, but we live at the same student flat. It's actually our rooms are across each other. I have a whiteboard. When we have an idea, then we just start talking about this idea and Bas, he starts drawing it all on my whiteboard and then we put it into the design tool. Then the implementation happens. That's just a product part. Bas does a lot of other stuff with [inaudible 0:51:28.7] as well.

[0:51:29.6] BB: Yeah, regarding the management of the business part. The patron, we have a patron feature, so people can support us and they get the ability to keep their sandbox as private, because if you're using CodeSandbox for free, you get all the features basically, but your sandboxes are public. This is so if you create something cool, other people can also see it and learn from it.

[0:51:49.8] JM: Cool. You guys are still in school?

[0:51:54.4] IvH: I'm not. I stopped with studying at some point, because I spoke – CodeSandbox got more popular, my grades went down. It was this correlated graph where I wasn't able to concentrate that much on studies anymore. Yeah, at some point I decided to drop out.

[0:52:09.1] BB: Yeah, you can say that we're –

[0:52:11.4] JM: Bas, yeah. I was going to say are you still attending classes?

[0:52:15.0] BB: I'm not attending class anymore, but I'm almost finished with my bachelor in industrial design actually. It's a whole different thing than focus on programming. Yeah.

[0:52:25.1] JM: That's probably more fun.

[0:52:27.6] IvH: Probably.

[0:52:28.9] BB: We still have Java classes. Some things are the same. Yeah, I'm almost done and stuff that I'm doing for my studies right now is CodeSandbox related.

[0:52:41.1] JM: Oh, cool.

[0:52:42.8] BB: Since beginning of 2019, we're both full-time on CodeSandbox. Yeah, you can say it's a true startup right now.

[0:52:48.6] JM: Very cool. Well guys, it's been really great talking. I look forward to the continued success of CodeSandbox.

[0:52:55.3] IvH: Yeah, thank you so much. I loved the conversation.

[0:52:58.9] JM: Yeah, likewise. Very good conversation.

[END OF INTERVIEW]

[0:53:05.0] JM: GoCD is a continuous delivery tool created by ThoughtWorks. It's open source, it's free to use and GoCD has all the features that you need for continuous delivery. You can model your deployment pipelines without installing any plugins. You can use the value stream map to visualize your end-to-end workflow. If you use Kubernetes, GoCD is a natural fit to add continuous delivery to your cloud native project.

With GoCD on Kubernetes, you define your build workflow, you let GoCD provision and scale your infrastructure on the fly and GoCD agents use Kubernetes to scale as needed. Check out gocd.org/sedaily and learn how you can get started. GoCD was built with the learnings of the ThoughtWorks engineering team, and they have talked in such detail about building the product in previous episodes of Software Engineering Daily. ThoughtWorks was very early to the continuous delivery trend and they know about continuous delivery as much as almost anybody in the industry.

It's great to always see continued progress on GoCD with new features, like Kubernetes integrations, so you know that you're investing in a continuous delivery tool that is built for the long-term. You can check it out for yourself at gocd.org/sedaily.

[END]