Chris Strahl:

Hi, and welcome to the Design Systems Podcast. This podcast is about the place where design and development overlap. We talk with experts to get their point of view about trends in design, code, and how it relates to the world around us. As always, this podcast is brought to you by Knapsack. Check us out at Knapsack.cloud. If you want to get in touch with the show, ask some questions, or generally tell us what you think, go ahead and tweet us @TheDSPod, we'd love to hear from you.

Hey everybody. Welcome to the Design Systems Podcast. I'm your host, Chris Strahl. Today we're going to do something a little bit different. We don't really talk a lot about Knapsack on this show, and the reason why is because we want to make this about the community. We want to have the stories, the voices, the people that are in the design system community to be those that we elevate. What's been interesting is, we have had a lot of people ask, "Hey, what is it actually that Knapsack does?" So as a part of talking about our own voice in the community, I'm going to have my co-founder Evan on here, Evan Lovely, and we're going to talk a little bit about what we do at Knapsack. Evan, welcome to the program.

Evan Lovely:

It's a pleasure to be here.

Chris Strahl:

Evan is somewhat responsible for this company, because we were both working together in an agency, gosh, a long time ago. And as we were sort of exiting that agency and figuring out what was next, Evan cornered me on a ski lift on Mount Hood, and basically said, "Hey, you remember that thing that we built, that was this systemized way of building websites? That's the future of the way the Internet's going to get built, and we should start a company that does that."

Evan Lovely:

It is pretty easy to corner someone on a ski lift. They really don't have anywhere to run, so you had to listen to the idea.

Chris Strahl:

Very captive audience. So from that, the coworkership turned friendship became a co-foundership out of that. And along the twisted pathway, we built this thing called Knapsack. I think, in my own words, the idea behind Knapsack is, we want to be the infrastructure and the tools for your design system. And that is an intention of streamlining a lot of the work that you have to do, just to stand up and make a design system work. Evan, you want to elaborate on, specifically, the early thoughts here, and what we were planning on?

Evan Lovely:

Yeah, of course. Well, I've got my work origins in being a front-end engineer, and had just experienced a lot of the pains that lead people to design systems, classically. It wasn't called Design Systems back then, but just trying to not repeat myself, trying not to pull my hair out, and trying to ship front-end UI at scale, so that had led me personally towards a lot of the approaches and practices around separating out front-end from its data models, so it's flexible and reusable. And naturally, of course, along that way, it's just like, "Okay, so how do you work in that? How do you view a card standing alone? There's a lot of necessary tooling."

And when Chris and I started working together, we were starting to help bigger companies put their design systems together. Over the course of multiple projects, we needed a lot of similar tooling and foundations. Actually, almost all design systems are incredibly unique and very different from each other, there's a lot of common needs. And like good design system enthusiasts we were, we did not want to repeat ourselves, so sharing that tooling between the projects and then building it up more and more, is how we ended up creating Knapsack. It's a collection of tools that pretty much everybody could use.

Chris Strahl:

So when somebody comes up and asks you what are the core foundational pieces of Knapsack, what do you say?

Evan Lovely:

Well, it's a platform for building out design systems, and it allows multiple personas to be able to collaborate together. So designers, developers, PMs, stakeholders, gives them that common ground. And the main things that you'll see in there, would be design tokens, documentation, but also of course components in multiple template languages, and being able to see them rendered in the medium it's destined for. So a web browser, because that's what users are ultimately going to see. So it gives you the chance to be able to actually experience your design the way that the end users will ultimately experience it.

Chris Strahl:

And you're also able to take things from Figma or other design tools and link them to those code patterns. And I think the final thing that is really important, is there's a workflow for publishing your design system. Where we create a design system package, we have a Git-based workflow that allows you to control how your design system is changed, and what actually goes out in that package. And then you, as a consumer, instead of having to go copy-paste something, or think about what you need to actually extract from the design system, we do that work for you.

And you can just include the Knapsack package in your downstream application bill. And I think that those early conversations paved the way for ultimately what represented a systematic approach to actually building design systems. And it's away from the content. I think that that's another interesting sort of thing to hit on, is it doesn't matter if there's material, or carbon, or your own custom thing inside of Knapsack, the entire intent was it could support lots of different things inside of your design system, but it was the scaffolding in the framework for how you get those things to be used and to work.

Evan Lovely:

Yep. I mean, doesn't matter what the buttons and cards look like, you need a place to be able to view them, to document them, to be able to resize them. And a lot of those common things that is needed across every design system team. And design system teams want to focus on making those components. They don't want to be focusing on having to make a props table, or an embeddable, resizable component playground, or anything like that. That's what Knapsack wants to be, is that platform for design systems, so people can get to their interesting work, which is what's unique to their system.

Chris Strahl:

Right, and I think it's also a little bit hard to justify if you're a member of a design systems team to think about, "I am tasked with building a design system component, but before I ever can actually build that component, I need to build some sort of resizable view port that allows me to actually see that component. Or I need to go and implement some other set of tooling." And I think a lot of the intent was to try to smooth that process, and make it something that was really integrated into the way that people work. I think that's also a big part of the reason why we didn't choose to just pick up Storybook, or to continue with Pattern Lab, because a lot of your early design system experience was in Pattern Lab. Why didn't we just make an extension of that?

Evan Lovely:

Well, I think that it's a developer-only tool really, and that's one reason. The other is just the component library, place to demonstrate it. But let me dive into that first one a little bit, is that it is if a designer wants to contribute towards it, they're going to have to contribute code. They're going to have to open up Markdown or MDX or just write it in a React component if they want to be able to add two Storybook or even to Pattern Lab as well. And so design systems are about more than just that one persona of the engineers. It is also about the designers. And so that's one reason.

And then the other is that it works great for basically saying, "Oh, here is a card with some props that showcases the image when it's on the left," or whatever, and then rendering it in the medium it's destined for in the browser. That part's great. Providing immediate feedback to an engineer who's trying to style it. But some of the other things like design tokens, it doesn't do as well. Back then what we were seeing a lot of times, is people would end up making a color swatch component, and then they would just loop over their colors to showcase that. And there's always a weird system in getting that there. There's also just the regular documentation, not just documentation associated with a card, but just full on docs pages. There's a lot that goes into a design system besides just components, and that's kind of what Storybook and Pattern Lab tends to focus on.

Chris Strahl:

Yeah, I remember you were the first person that I heard the words, "Design tokens," from. And I remember thinking about that as something that was core and foundational to our product from the beginning you talked about this idea of, "Hey, we need a bunch of key value pairs that essentially represent a brand." And I remember reading [inaudible 00:09:12] book about design systems, thinking about how those lined up to the perceptual patterns that are described there. Those foundational choices around Knapsack were pretty interesting because it was stuff that I'd never heard in industry before. What were some of the other things that we thought about early on, that were those sort of foundational decisions that have been important in shaping the way that Knapsack works for organizations?

Evan Lovely:

I remember there's always this decision of the spectrum that you want the design system to be on, about how flexible or opinionated it is. That's a really important one for teams to make, in my opinion. It can really be one of those principles that can help when you are facing, "Hey, we have a couple ways to do this," looking at that principle is going to help guide you. So if you want to be able to have a very flexible, unopinionated system, you might be able to stick the icon component in all sorts of places in other components, for example. And if you're really opinionated, you might basically only say, "Hey, tell me, the button would accept only what the icon name is, but we're going to put it right in that one spot so

we get some consistency," because maybe design doesn't want word-icon-word. Whereas in the opinionated, you could build it if you want, even though it might look a little weird. So that kind of flexibility and opinionated spectrum is an important one to make.

Chris Strahl:

And where would you say that Knapsack fell down on that? What was our take on that flexible versus opinionated side of things?

Evan Lovely:

See, this is the thing that has always been really interesting about Knapsack, is that there comes decisions that we simply can't make for other design systems. Like what I just mentioned there. There is no wrong answer there. What I encourage teams to do, is make sure that one component is not opinionated, and another one is unopinionated and flexible. But anyway, with Knapsack there, it's always really interesting on basically, "Okay, this is some decisions that we simply can't make, because we want to be flexible. We don't want to get in everyone's way." This was led us to basically the unopinionated functionality with the opinionated implementation, was the thing that we'd always did there. It's like, "Okay, so Knapsack needed to be really unopinionated and flexible." And it's funny, because I feel like we're starting to realize that people want a little bit more strong opinions on what are the best practices of how to do things, but we definitely got started by basically saying, "Let's not make too many decisions for people."

Chris Strahl:

Yeah, and I mean I think that's really lended ourselves towards the enterprise market. I mean that's where we are is we work with really big organizations that have lots of complexity and lots of scale. Because the intention of Knapsack was to streamline all the work you have to do, to actually get a functioning design system up and running, and then maintain it long term. And the ability to have that unopinionated way of thinking allows us a lot of flexibility to work with whatever environment, or whatever set-up a big enterprise customer has.

And I think that one of the things that [inaudible 00:12:33] really interesting, as a CPO of an organization told me a few months ago, is that it's very focused on a practical way of implementing a design system at scale. You want to have the ability to do multi-brand management, you want to have the ability to have lots of different user types. And have supportive workflows, and integrate it to your CI and all these other different things that Knapsack does, with the intent of it not just being a reference system that people look at, but actually being where a lot of the core development of your application takes place, is in the patterns that are ultimately the building blocks of those apps downstream.

Evan Lovely:

And I think that the height to which we're able to scale, is because of that unopinionated nature, that it's very flexible there. Sometimes it can be a little tricky to get going with our product, and sometimes that makes me wish that it was much more opinionated, because I think that it's a lot easier to get going. But I think that the height to which it can scale, is something that I wouldn't really want us to lose. And so it could be tricky to basically be easy in the beginning, and can handle complicated scenarios later. It's a tricky one to hit both for.

Chris Strahl:

Yeah, and I mean it's a big part of the reason why something like the ability to sign up for Knapsack online is a really difficult problem to solve, is because there's different Git-providers, and different workflows, and different structures, and different CIs that all need to be a part of the holistic solution.

Evan Lovely:

The choice of how coupled we are with a Git-repo is a very different one for a lot of SaaS apps, but I think it's one thing that is very interesting. And so we store basically all the content as JSON and YAML files inside the Git repository. It lives right next to the source code, you can use some NPM package links to have it set up in some different ways of course. But this is an interesting piece, because basically you're able to showcase the rendered components as they are for a given Git commit, which that can be confusing for some people, but it could be awesome for others.

What this basically means, is you get all the benefits of Git, is you could check out your design system a year ago, and take a look at it. Not just the components, but also the docs. People can branch and work in on different things at the same time. In fact, actually in Knapsack, when you start working in content, the first thing you do is make a branch, and then when designers basically propose some changes, that's proposing changes via poll request. That is kind of an interesting thing right there. It's nice to be able to have it, so designers can start thinking that same way with branching, but then actually using some of the time tested standards of Git, without scaring them with a terminal, or anything like that.

Chris Strahl:

Yeah, I mean the ability to have lots of different people working on the same system at the same time, and all making changes. And then being able to collect those changes and then publish that as a version controlled thing that then goes out to all the consumers of the design system, that's powerful for enterprises, because you need that ability to control how your design system evolves and changes at a pretty fine grain level.

Evan Lovely:

Absolutely, and I mean, again, some of these needs aren't really exposed until you hit some scale. And so if you had a design system that was used in a hundred different sites or apps, and you might be on version nine, there might be somebody still using version two out there, and they would be able to go and take a look at how the components were for version two, but also what the docs were and what the design tokens were, and all of the usage guidelines and everything else that goes with it, back on version two. And so to be able to support that kind of stuff, you're going to have people with, on so many different versions of your design system. And of course you want the most to update to the latest and greatest, but that's not, the reality is that can't always happen.

Chris Strahl:

Right, especially when you come into principles of inheritance. There's lots of big companies that have a core design system, like a foundations design system, and then they have all these systems of systems that are product design systems, or brand design systems that ultimately inherit something from that central system. And what they choose to inherit from, and how they choose to take that inheritance and build then their own set of content, is something that we also enable with Knapsack where you can basically say, "I can have a workspace in Knapsack that ultimately feeds another workspace, and provides a foundation for changing that individual product or brand system, based on that initial foundation." And that systems of systems concept is really difficult to model out, because you have so many different

versions that interact with each other, and so many layers of dependencies that it's actually really tricky to design around that.

Evan Lovely:

Yeah, and this is where we store our content as JSON, YAML, Markdown inside the Git repo really has an advantage, is because if you have one design system that has a dependency on the core design system, you can import and inherit the content as well. I mean you can be able to pull in a page which is just a JSON, or YAML file from the core system, and then pull it into yours, maybe override a few things, add a few things. But that flexibility is important, because that's what happens with design systems in packages and code. That's kind of the reality of where it's built, and how it's made.

Chris Strahl:

So when we think about how Knapsack helps an organization scale, we tend to think about this in three buckets. There's the group that has really mature design systems, that are out there that have been doing this for a while now, and that have a lot of tooling, and a lot of investment into what is predominantly custom systems. They might use Storybook here and there. They might leverage some other tools here and there, but on the whole these are things that they've built themselves. Then there's this sort of mid-tier, which is people that are transitioning from either a rebuild of a design system, or a new design system, that are companies that are trying to figure out how they actually scale their digital apps.

And then there's smaller organizations, start-ups, high growth companies, that sort of thing, that are trying to use design systems to tackle this innovation edge. We've mostly aimed at that middle tier, and some of that first tier, where we think about how do we capture the attention of people that are looking at a choice between, "Let's go build something out of either completely custom, or out of a series of tools that we integrate together," or, "Let's go pick something like Knapsack," which represents a more all-in-one kind of complete platform for your design system. How do you think about that decision process for those organizations?

Evan Lovely:

Usually a design system team wants to build a design system, not a space where a design system can get built. So focus on what you actually want to ship, which is actually going to get out to your end users, your buttons and cards and [inaudible 00:20:02] and things like that, that's actually going to get out to your final end users. A lot of people aren't going to end up seeing your design system documentation site. We all love material UI, and things like that, but very few big companies actually make their design system publicly available, and if they do, their goal is not for people to adopt and support it. So focus on what's going to shift towards your actual customers, and not make your own platform.

Chris Strahl:

And most people that are making these custom platforms, the investment that they're making, is one that looks at like, "How do I build up a bunch of tools, and how do I build essentially a stack that lets me support the content of my design system?" And what we're trying to take away, is the need to build and maintain that stack, that platform, which is where you want to actually put that content of that design system. So shifting gears, we talked a fair bit about what Knapsack is, why the use case we have is valuable, but you and I are both really fond of the saying, "The interesting stuff happens once you already have a design system."

And so if we think about what the next chapter of Knapsack really represents, and we have this great design system platform that's out there right now, it's really about what we can build on top of that platform that is interesting and exciting for the future of this whole systems and product concept. I want to have a chat about where we think this is all headed. Because there's lots here about generative design and AI, there's lots here about data, there's lots here about all these different stories, about all of the interesting things you can do to further this concept into a company to help them scale even faster, once these sorts of things are in place.

Evan Lovely:

Yep, so the interesting thing is that happens once you have a design system, obviously that's going to be a force multiplier inside that company. So if they're able to be able to get more out that's at a higher quality, I think then there's a different layer of challenges that come, is basically how do they manage that product within their company, the design system being the product. How do you handle contribution? How do you handle bug reports and feature requests? And how do you communicate a roadmap? Those are what we want to be able to solve for, because it's about maintaining that. And then also how's it getting used, how's your design system getting used? And being able to get feedback from that. And a lot of this stuff just echoes basically what it's like to run a product, shipping 100 components. Is there any component that is not used at all? What is our most used component?

We should probably stop supporting the component that no one else is using, because we're wasting our time. And we should probably double down on that most popular one. So it gets towards a classic data and analytics question a little bit, is basically how is my stuff getting used? I think that every design system wants to be able to answer the question, "What's the return on an investment here? I think that basically being able to say that to the people that are giving their team budget, is going to basically be the difference between expanding the team, or shutting the system down. So we want to be able to help provide that data, and that mechanism for being able to have feedback. Because you build something, you ship it, you get some feedback on it, and then you improve it.

Chris Strahl:

So I think about this also as shifting a metric away from like, "How much does a design system save you in terms of cost, to how effective a design system is at delivering better product." Everybody always talks about this efficiency side of design systems, where it saves you time, it lowers the cost of the product creation process. But one of the things that is, I guess, underserved in that conversation, is how much better it makes the product you build. So it's not just a, "Faster, cheaper," conversation. In my opinion, it's absolutely about the, "Better," conversation, and shipping more effective product. And the ability to put attention towards the hard problems and automate away the mundane is what we're all about. We're trying to basically figure out what data can we gather for a customer of Knapsack, that shows that that product is more effective?

And that can be accessibility data about, "Hey, we created a more accessible experience." It can be performance data about like, "Hey, these experiences that we're producing are all fitting within a performance budget." It can just be usage data about like, "Hey, this particular experience is really highly leveraged across our ecosystem of applications." But all of those also represent an automation story. If you're automating your accessibility testing, or you're automating the collection of data about your system, that's one less thing that you have to make a decision around, to decide how to collect, and use, and store, and manage that data, that ultimately reflects a return on the value of your design system.

Evan Lovely:

Yep, totally.

Chris Strahl:

I think that there's also a lot of interesting things to do, to think about, when we think about the future and the management of this contribution, you think about, more and more how this relates to collaboration software. And so I'm not really necessarily talking about Miro, or Whimsical, or Lucidchart, or anything like that. I'm more thinking about when you actually are getting work done, and you're all working in that same place, what does that workflow really look like to people? Is it something that follows a traditional paradigm of a bunch of stuff lives in design land for a really long time, and then there's this magical point where a handoff meeting occurs, and then it lives in engineering?

We're really trying to break that down. And we're trying to basically say, "Very early on in that design process, you should get something into the medium it's destined for." Design's function in the role and an ideal use case of Knapsack, is that you're only doing what's absolutely necessary to express the intent of what you're building. And then very quickly that gets into code, because then you can have a conversation, designer and engineer, about is that code acting, or behaving, or doing the things that we want to do in the medium it's destined for, in the same way that a user is going to see that code expressed?

And once you're there, you can iterate really quickly, inside of our system, based on this ability to represent what contribution needs to happen, and what is the next thing that needs to be done to improve that system. And I think that when we think about that contribution management, and that collaboration side of things, that's what we're talking about. Right now, so much collaboration happens in Figma, it's super overloaded, and there's lots of great tools that support it. I'm not shitting on Figma, I think there's an amazing amount of stuff that application does, but when you're actually sitting there trying to say, "I'm going to go figure out what the next thing I need to do to improve my component is," I don't think that's a conversation that should exist in Figma. I think that's a conversation that should exist next to where that actual code lives, that is representing that component.

Evan Lovely:

Well, yeah, because if it's in Figma, you still have the translation step. You still have basically the engineer having to code it. And it's not about basically like, "Oh, is design the source of truth? Or code the source of truth?" It's what the ultimate end user experiences, that's the source of truth. That's how people's brands are actually being experienced. And that's rendered code. I mean, unless you're a company that makes posters, then it's very different. Then the actual source of truth is like a PDF.

Chris Strahl:

I don't want to take anything away from designers in this conversation, right? Because there is a step that exists in Figma that is really important, and that is deciding how much intent I need to express? What level of fidelity do I need to get into? Because one of the things that's always been hard for me, and I come from more of an engineering background than I do from a design background, is seeing all of the limitations that we impose upon ourselves because our design tools don't express things the same way as a web browser, or as a native app.

And because of that, we actually limit our capacity, by creating really high fidelity things in things like Figma, because ultimately Figma does less than our web browsers and our cell phones. And I think that a

designer's job is going to start to shift a little bit here towards the idea of, how far do I need to take this so that I can express what I'm intending. But not actually saying, "This has to be exactly what that user experience looks like," inside of a different medium than the user will ever see.

Evan Lovely:

Yeah, and I'm with you there. Designs are great. They really give a lot of clarity to what can be in someone's mind. It is a crucial step for how you plan it. And I guess my point there, is don't spend hours in Figma making new designs when a 30-minute conversation can do the trick. A designer and a developer collaborating together, with a designer being able to say, "Can you make that animation just a little bit faster? Can the element when it moves, bounce a little bit? Okay, that's a little bit too ragged, can it feel a little bit more playful?"

Those things would be impossible to be able to tackle in Figma. And being able to see it in the medium it's destined for, and do rapid iteration on a tiny piece, that's what we're talking about. That's what we're advocating for. So don't go to After Effects and make a movie of how animation should be. Go ahead and collaborate with somebody in the meeting it's destined for. You're going to save it a lot of time, and it's going to be more accurate to what you envision, and what your end users actually experience.

Chris Strahl:

Or also have a prop that is like, "Animate this," and then be able to set an animation style and an animation timing token, and be empowered to do it yourself, without ever having to dive deeply into the code. I think that's another part of this that is interesting, is the empowerment that this brings to designers, product people, and to engineers. If you're able to build without having to know exactly what the right prop definition is, and what the right data structure is, and just be able to hit a bunch of drop-downs to make an experience look and feel the way you want, and to be able to snap a bunch of those experiences together, you can get really far just with the ability to stack components on top of each other

Evan Lovely:

Yep, and play with props and play with token values and all that can be done in Knapsack. And you have an engineer who basically sets it up so that the animation speed prop actually gets passed to the right thing, and then lo and behold, you have a [inaudible 00:30:51] that a designer can play with, to see what that's like, to see what is too fast, what is too slow on an animation.

Chris Strahl:

And the other interesting thing that is there, is there's lots of people that want to present a reference site for a design system. And they want to have that storefront workshop kind of mentality. I think that one of the things that we've done pretty successfully with our customers, is we've been able to figure out a way to actually publish that reference site publicly, or semi-publicly, that represents a really rich, interesting and engaging experience for people that are just exploring the design system. But also has that practical workshop of, "I actually need to get these components into my code."

So it's not just a beautiful reference site that people look at and refer to once in a while. It's that, and it's something that practically ends up in your products. So when we think about some of the more far-flung ideas, and some of the more interesting stuff about how Knapsack's adding value, when you take structured design tokens that are pretty universal and cross-platform, you take a bunch of JSON data

structures and components that are reading from live code, and all these other different things, and you put it all together, there's a lot you can measure about a design system. And not all of it is just about effectiveness and things like that.

You can feed that into learning models that actually help make a lot of decisions for you about the way you should be thinking about structuring a page, or component, or some other part of your design system. Now, this gets really technical and really heady really quick, but in the age of AI, in the past six months of all the explosion of this that we've seen, generative tools that automate away a lot of mundane things is where a lot of the value fits. How do you see that happening in Knapsack, Evan?

Evan Lovely:

Well, we've always believed in structured data, so we know all the props and slots that components can take, and anytime we're basically composing the different demos, so that might be, "Here's the button with its type set to primary, with its size set to large, with its text set to, 'Hello world,' that's actually going to render something you see in the browser," that is all structured data. So that works really well with AI being able to feed in what people have done before, and what you might want to do. So a lot of the mundane, exhaustive work, of basically making all of the demos, of, "Here's all the ways that my components might get used with some real content," that can be a way that AI could really help out. But also with composing components, because we do the same thing with, "All right, I want to show a card, but I want to put a button in it, and I want to put an icon inside the button."

And so if you just play that out, I mean that can be a whole page. It's just composed components. This isn't Dream Weaver, you have professionally crafted building blocks, components. But the composition of them, putting them together can easily be done by a machine. So that gets real interesting real quick. So if you combine this with being able to have some kind of analytics on components, be able to say, "All right, we want people to end up doing this desired action," you could easily be able to go, "All right, let's go ahead and play with different settings. All right, does a small button or a large button convert better?" That's obviously contrived and simple, but there's a lot of possibilities here. So you get to a point where you could actually start getting to data-driven design, where what gets shipped out, is the thing that converts the best.

And before any designer gets all scared about this, the thing is that designers already created, like, "These are the okay variations," and it's just picking which one of those. Design systems is all about limiting choices. If you think about a design token, and we have a font-size scale. Instead of saying, "Hey, pick any font size from one to 200, with 200 choices," we're saying, "No, go ahead and either pick 10 pixels, 12 pixels, 16, 18, 24, 32." And so about giving a finite number of choices. Same thing with color. Instead of saying, "Hey, pick one of 65 million," you're basically saying, "Hey, here is, let's just say our 20 colors," or whatever. And being able to say, "Pick within those." And so it's up to humans to be able to craft what those are. But to be able to pick the best combination of them, or an interesting combination of them, or a different one, especially to be able to do trials and feedback, there's a lot of benefit there to be able to have a machine running those experiments.

Chris Strahl:

Absolutely, and if you think about it, I'm going to pick something that is, I'm hoping a pretty mundane example. If you think about a pricing page, and you think about your average SaaS company's three column layout, where you have freemium or cheapium, medium, and then an enterprise, and then you have your company's nav and your header and your footer, there's not a lot of design mojo that goes into the design of that page. It's kind of samey for the vast majority of companies that are out there. And

I've intentionally picked a really samey example to not have the person that is the pricing page designer DM me on Twitter. But the idea is we probably don't need to redesign that page. That's probably not the thing that is going to 10X signups for our app. But the thing that might 10X signups for our app, is now something that can garner the attention of the humans that are making those design choices.

And so I would love to see a future where inside of Knapsack, you could say, "Hey, Knapsack, build me a pricing page with three columns that uses Card A, Card B, and Card C," and then have it just go, and build that for you. And then obviously, you still need to put content in it, and stuff like that. Or maybe you don't, maybe we figure that out too. But the idea is that saves us time and effort on something that is fairly mundane, so we can spend a lot of time on what does that actual app signup experience look like. Because that's the thing that's going to convert new users, not our pricing page.

Evan Lovely:

Yep, exactly. It's about standing on the shoulders of giants, so you can get to the interesting thing.

Chris Strahl:

Awesome. Well, hey, Evan, thanks so much for jamming with me. It's always great to chat about this stuff, and to get to do it in front of a mic. And I'm just really glad that I get to work with my friends, like you, to build this really awesome thing. I mean, I really do believe that Knapsack is changing the way that people think about building digital products, and it's great to just sit and jam about some visionary stuff.

Evan Lovely:

It feels good to be building out the tool that I wish that I had, and try to be able to help out other people that way. And of course, always a pleasure chatting with you.

Chris Strahl:

Hey, if this conversation was interesting to you and you want to learn more about Knapsack, you can always check us out at knapsack.cloud, or you can hit us up on the show. We [inaudible 00:38:10] to show you the product, talk things through with you, and see if it might be a good fit for you. Hope you have a great day.

That's all for today. This has been another episode of the Design Systems Podcast. Thanks for listening. If you have any questions or a topic you'd like to know more about, find us on Twitter at TheDSPod. We'd love to hear from you with show ideas, recommendations, questions, or comments. As always, this pod is brought to you by Knapsack, you can check us out at knapsack.cloud. Have a great day.