Episode 6: Debugging by Design

There are still significant inequalities when it comes to who has access to computer science education. In this episode, In this episode, Dr. Sara Grimes (Director of the KMDI) chats with Dr. Deborah Fields, Associate Research Professor in the Instructional Technology and Learning Sciences Department at Utah State University, about her research on the relationship between identity, motivation and learning how to code among tweens and teens, and how to break down stereotypes about who can code and how.

Dr. Sara Grimes 0:00

When did you first learn how to code, how to design or build, or even just read computer programs? Maybe it was making that little logo turtle move in computer class, or figuring out how to copy lines of HTML from a favorite website so you could add it to your own. Maybe it was completing the hour of code challenge in Minecraft. Since the 1980s, a ton of emphasis has been put on teaching kids and grownups how to code and no wonder. We're surrounded by code by devices, systems and interactions running on or through computer programs. And as computing technologies spread across our lives and environments, knowing how they function, even if it's just at a basic level, is critical for understanding the world we live in. Despite the seeming prevalence of school computer classes and programming apps, and learn to code initiatives over the last several decades, we still don't all know how to code.

And part of that is due to inequitable access. According to a study published in 2021 by nonprofit code.org, only half of high schools in the US teach computer science. The rates aren't that much better in Canada, where less than half of the provinces and territories teach computer science is part of the elementary or middle school curriculum. Ongoing disparities in who exactly gets to develop computing skills and literacies follow existing lines of systemic oppression, including race, class, and gender, which are also embedded in the very designs of our computing technologies. But access is just part of the story. Even in well-equipped schools with plenty of supports for STEM education, after-school computer clubs and elective computer science classes tend to be dominated by white

boys. Getting more girls and more children of color into coding are high priorities for many educators, governments, and some parts of the tech industries, where women, BIPOC, and LGBTQ+ people are still very much underrepresented and marginalized.

The barriers to solving these issues are complicated. They're structural and cultural. They require an inclusive intersectional and child centered approach that resituates coding as an accessible and relevant part of diverse children's everyday lives. Dr. Deborah A Fields, Associate Research Professor in the Instructional Technology and Learning Sciences Department at Utah State University, is doing just that. Her research focuses on developing innovative workshops and cutting edge pedagogical approaches for teaching coding and other STEM subjects to teens and educators. Her work explores the relationships between learning, identity, and motivation. And her theories about these relationships and how to foster them in positive ways have had a game changing impact on the fields of education, technology, and media studies. She's also a key part of a world-renowned research team, led by Dr. Yasmin Kafai, that's designed a number of excellent online tools promoting coding and creativity to kids across K-12 and beyond, including the critically acclaimed visual program language, Scratch.

Dr. Fields' work appears in numerous publications, including the book Connected Play: Tweens in a Virtual World, co-authored with Dr. Kafai, published by the MIT press in 2013, and the the article Electronic Textiles as Disruptive Designs Supporting and Challenging Maker Activities and Schools. Co-authored with Drs. Kafai and Searle, and published in the Harvard Educational Review in 2014. Full disclosure: I've had the enormous privilege of collaborating with Dr. Fields myself on two major research projects, the Kids DIY Media Partnership, and the Kids Online White Paper for the Joan Ganz Cooney Center at the Sesame Workshop. Her most recent collaboration with Dr. Kafai, Luis Morales-Navarro, Justice Walker, and others focuses on the impact of bugs, mistakes, and other errors on kids' learning and experience of computer science. Dr. Fields and her colleagues are flipping the script about bugs, turning them into opportunities for kids to build community, develop a growth mindset, and start seeing themselves as coders.

The findings from this project have already been published in several journals and conference proceedings, most notably in an article entitled Debugging by Design, a Constructionist Approach to High School Students Crafting and Coding of Electronic

Styles as Failure Artifacts, published in 2021 in the British Journal of Educational Technology. I'm Sara Grimes, director of the Knowledge Media Design Institute at the University of Toronto and host of the Critical Technology podcast. Today I'll be speaking with Dr. Deborah Fields about the Debugging by Design project, and how to make coding more creative and fun for young people who don't yet see themselves as coders. So let's just jump right in. What is constructionism?

Dr. Deborah Fields 5:11

Constructionism is an educational philosophy that was developed by Seymour Papert in the 1980s, and the basic idea is that we learn best when we're making things. It's like the knowledge structures in our head fit together when we're making something in real life, whether that's a sand castle or a Lego structure, or putting together a kitchen appliance, or maybe even just writing that paper that you had to do. We usually remember these things more and they actually help us learn. I'm sure most people can sympathize with that idea, that you remember the things that you've created. But not only that, we learn when these things can be public. So when I can see what you're making and when you can see what I'm making, and we can have a conversation about it or get ideas from each other, so when we are in community and the stuff we are making is shareable and visible and it leads to conversations and it might lead to changes in what we make.

Dr. Sara Grimes 6:14

And why is it so important for kids to learn how to program or build circuit boards?

Dr. Deborah Fields 6:20

There are a number of reasons to argue for kids to learn to program. Some of them are jobs. It'll give them workforce opportunities and all that. I tend to be along the lines of the literacy side. Just like it's important for us to read and write because we're surrounded by texts and the world operates around text and images, and those sorts of things. Computation is now one of those things that surrounds us. Doesn't have to mean you're always on your computer. Computation is in the stoplights. It's in those doors that open automatically. It's in the purchases that we make. It's in the data that's being collected on us all the time. And it's really empowering for everyone, and especially kids, to be able to take part in at least a little bit of how that computation exists.

So if you can learn a program, it equips people in one of the basic literacies of modern life.

The other part is it's important for people to know that that is something that's possible for them. Just like reading and writing, programming is accessible to everyone. It hasn't always been made accessible in the way it's been taught, but it absolutely is possible for anyone to learn. One of my favorite things ever was teaching an 80-year old woman how to program a birthday card for her adult son, who was himself a professional programmer, but she'd never actually programmed. So everyone can learn, young or old.

Dr. Sara Grimes 7:42
One of your recent articles published last year in the British Journal of Educational Technology, coauthored with Kafai, Morales, Navarro, and Walker, introduces a new idea and pedagogical tool, which you call Debugging by Design. What is Debugging by Design?

Dr. Deborah Fields 8:00
So let me back up a little bit to answer what Debugging by Design is. First of all, if you've ever made anything at all, you know that it probably did not go right the first time. And so you had to do what we call debug it. The bugs would be the things that go wrong, the things that are little errors, unintentional or just normal. So by debugging things, we remove the bugs. It's a formal term in computer programming to debug a computer program to make it work better. So pretty much nothing goes well the first time we do it and we all have to debug as part of the process of creating something. However, one of the interesting things that we can do is give people practice in actually debugging. And so often what happens in schools is the teachers or the instructors create bugs for people to solve.

When you think about all the math problems you ever solved as a kid, those were designed by someone and you got to solve them. So think about creating a computer program or a craft project or an electronic thing. Bugs are going to happen, and as an instructor, we could design bugs that we think will help people learn by solving them. So what we wanted to do with Debugging by Design was not give that power of designing bugs to instructors, but give it to kids so that they could learn by creating bugs. And we did it in this social way, where we had kids create buggy projects for each other to solve. So it was part of this unit that we had on what we call electronic textiles, which are these really fun projects where we sew electronics on fabric with conductive thread. And then we sew a computer on there and then we can make these lights turn on and we can make it sensitive to touch or light or those sorts of things.

So these interactive fabric, computer objects, all those words together usually blow my mind, and we have bugs as part of them and the kids solve them. But the bugs that you have are not usually very fun to solve, because you don't know what the answers are. So instead, we put kids in charge of bugs, so they were no longer susceptible to these bugs that would come up that they couldn't predict. Instead we said, "Go on, make some bugs and then give it to your other friends in class, we've mixed and matched partners. And then you have an hour to solve them."

Dr. Sara Grimes 10:24
You mentioned just now that bugs aren't usually fun to solve. And that makes sense, since they're normally experienced or encountered as mistakes, as failures. What does the Debugging by Design project tell us the role of failure in kids learning?

Dr. Deborah Fields 10:39
Well, failure is a part of all of our learning, but we don't usually like to talk about it. Sometimes people are of the idea that, well, we should protect people from failure because we don't want them to get too discouraged. And yet studies have found that when we fail or think about it not as a terminal failure, but like a temporary failure. So when you run into a wall on something, say you run into a project you can't finish it. Or one of the areas that came up originally was in math education. And you encounter a problem that you don't know how to solve, and you don't finish it. So let's consider that kind of a failure. They found that when kids did that, when they were given a hard, difficult, unstructured problem that they were not fully capable of solving, they actually did better in their next problem set, compared to kids that were just really given all of the help they needed and all of the scaffolds and all of the supports.

So we know that failure can be productive for learning. And there's a whole genre of work called productive failure that's a part of this. So when we get into Debugging by Design, we kind of played with this idea, except we turned it on its head because we wanted kids to design those buggy projects for other people to solve, instead of the instructors designing really hard bugs for the kids to solve. So we use Debugging by Design in the middle of a unit where kids are already creating things. They have enough knowledge and frankly enough experience with bugs happening in their projects that they could use those for others. And one of the funnest things that happened is that a lot of the bugs kids designed were ones that came up for them when they were making their work.

So it's like, ah, this thing happened when I was making my own design. I am going to design this thing for someone else to solve. Ha. And that's almost therapeutic, I think, to give someone else a problem that you've encountered and now you suffer through this problem, but it also becomes a learning experience for both the designers and the recipients of the problems. So in terms of thinking about failure in kids' learning, it's just part of the learning process and we need to find ways to make sure that kids understand it, that you don't always solve every problem perfectly the first time, and that it's totally normal to run across things that are too difficult for you but that actually the process of coming across those problems that are too difficult for where you are right now, that will actually help you in the next step of your learning.

Dr. Sara Grimes 13:25
That leads really well into my next question about a concept discussed in the article, which you and your co-authors call socially meaningful failure artifacts. Would you mind telling us a little bit more about this term, what it refers to and why it's important?

Dr. Deborah Fields 13:40
I think what we mean by socially meaningful failure artifacts is making failure social, so making it obvious to other people and then making it meaningful, where this becomes part of a valued learning process. Some of the best teachers that I've worked with actually celebrate the problems that come up during class. So they'll actually call out, "Hey, guess what? So and so did?" This is my favorite failure of the day because it becomes a learning moment for the entire class. Think about it. When you make something, you run into problems and maybe hopefully you eventually solve those problems. And those problems are probably the things you remember the most, and they become some of your most visible, your most memorable learning moments.

But think about it, if you could learn not only from the problems that you encounter, but problems that I encounter, or that other people in the room encounter too, then we just multiply the learning opportunities because all of these problems, these mini failures become socially meaningful for us and become ways that, oh, maybe I can avoid the problem that you made since I know about it. You encountered it, you came up with a way to solve it. Maybe I can avoid it. Or maybe if I'm already running into it, you have a way to help me solve it. This is happening on forums around the world everywhere. We're just trying to help it happen in classrooms. And the only way you can make it happen is if you make it something that it's shareable, that it's okay to share that you ran into a mistake, even a silly mistake, because we all do that.

Dr. Sara Grimes 15:19

In this article, and actually in a lot of your work over the years, you argue that there's a meaningful connection between learning and creativity. What is the relationship between creativity and learning, and why is creativity such a crucial part of kids' engagement with STEM?

Dr. Deborah Fields 15:35

It's probably easiest. If I just give an example. When we were first studying the projects that kids made, these crazy syllable computational objects, we called e-textiles. One thing we noticed is that when kids were working on the parts where they really cared what it looked like, that were personal to them, we might call it the aesthetics of the thing they were making. And I don't mean aesthetics with a really big, heavy meaning. I just mean like, I wanted this to look like that college lecture that I want to go to, or I was making a cool belt that has lights on it. When we did that, they created problems for themselves that were unique. And by solving those problems, they created their own sort of mini specialization of learning. And it was so much better.

Their learning was so much more thorough. Their learning was so much deeper in those moments where they were making this thing they really cared about, that it took them way farther than if they had just looked at this, oh, here's this model for how to create this circuitry, ta da, go ahead and do that. When they were instead having to alter these ideas or customize the knowledge about coding and circuitry and crafting to make their particular object, we found that was when they learned the most. So that actually led us to train teachers in this really interesting idea that we like to call aesthetics first. It's basically like we make kids draw what they want to make before we tell them anything about the technicalities of how to make it. So they have to draw their idea for this textile object and where are the lights going to be and what is it going to do?

And of course, because they have no domain knowledge about how to make this, they come up with ideas way beyond their skill level, even at the end of the unit. But as part of having that, they have this idea in their head. And once they have ownership of that idea, then we can introduce them to some of the skills to begin to accomplish that. And their original idea will change. It changes in ways that make it more approachable and they learn a lot, and they keep ownership of it, versus here's a simple project that everyone can make. Go ahead and repeat what I have made. That does not lead to tons of learning. Obviously you'll learn something by doing what that person made, but if you're starting to adapt those ideas to something that's personal to you, that helps.

Dr. Sara Grimes 19:10

Another important theme in your work, and one that's already come up quite a bit in your responses so far today, is the key role that peers can play in constructionist learning experiences. What roles did peers play in the Debugging by Design study? And how did this compare to some of the other workshops you've led with children and teens?

Dr. Deborah Fields 19:28

Let me talk about this in two ways. One, everyone was with one or two partners in designing their buggy project, so there were peers as co-designers. And then of course these projects were intended for someone else to solve. So I'd say the way it connects to some of the other work that we've done before is this sort of co-designing the kids together working on things. I will say that one of the really fun things that we noticed in this particular context of designing buggy projects, that the kids were really encouraging to each other. So one of the funny things that happened is while they're designing bugs intentionally, they of course accidentally designed unintentional bugs. So this usually puts a wrench in things and they have to figure it out. And we saw kids just encouraging people to persevere, coming up with ideas, saying, ah, you got this, just keep on trying or suggesting ideas.

And also they had a lot of fun coming up with the ideas together. There is something really beautiful about creating a buggy project for someone else, the whole mischievousness that came about. So this starts to relate to the second part of the answer in terms of the audience. There was a recipient for these buggy projects, namely other kids in class. And it is so much fun to design bugs for someone else to solve. Like I said, it's a little cathartic, a little therapeutic and the mischievousness was something we've never seen before in any of the workshops we've done for close to a decade now. A couple things happened that were really interesting about this sort of audience, the peers as audience for these buggy projects. One of them was the mischievousness and that, oh hahaha, I am going to design this crazy bug or this bug that I encountered and I'm going to give it to this group of kids to solve.

And so there was that fun and these kids delighted in trying to write tricky code, that would be intentionally deceptional, which was just not something we ever thought of before we saw the kids actually do that. So there's a bit of that. And that was really fun. And it got them frankly, to think about code in new ways that they hadn't before in the project. And the other part is they actually were somewhat merciful towards their friends. I've emphasized the mischievous side, but we also told the students that they only had

one class period, so a little less than an hour to solve the projects. So we encouraged them to think about all right, so how many bugs is too much? Because you don't want people to be frustrated to the point of giving up. You want some frustration, but not too much.

So they actually really started to think about their projects as a whole, these buggy projects. How many bugs is too much? Is this going to be too hard? Is going to require them to completely resew this project? So they actually cut out some bugs that they had intended, refined them a little. It was really interesting to see the way that the audience or the recipients of these buggy projects shaped their design. And then of course, they all got to compare at the end, like which bugs did you solve? Which couldn't you find? Some kids found all of them, some were too difficult, some were beyond their knowledge and everyone had a lot of fun with it. And there's also a little bit of victory. You know, the designer's like, haha, you didn't discover this bug or wow, you fixed this one that I accidentally put in that I didn't realize that I put in. Those sorts of things happened.

Dr. Sara Grimes 23:13

One of the things that really struck me when I first read the article was how so many of the kids initially reported feeling frustrated with this part of the workshop and with the whole debugging process. But several weeks later, frustration was replaced by feelings of comfort and confidence. Can you please say a little more about the role or impact of frustration in this project and more generally in kids learning?

Dr. Deborah Fields 23:37

So first in relation to this particular project, when they were solving the buggy projects that they had just exchanged, that was the last part of this little mini unit. And so when we had them write about their feelings immediately afterwards, they were very, very aware that they had not solved all of the bugs or they were frustrated by the number of mistakes, because this is a project with a lot of mistakes all at once. And yet a few weeks later, they really only had positive things to say about this and some of the kids were like, "Can we please have more? Can we do this more often, like maybe earlier?" Because all of a sudden they had so much expertise with solving bugs. They had a thought process to apply like, okay, there's a bug. Let me think about what I know about it, where it might be taking place so that I can focus some of the sort of isolating the bug, figuring out where it is and then how to solve it.

They also had potential solutions, and it had just been incredibly visible to everyone that everyone in the class makes mistakes and needs help solving them. So they felt way more comfortable asking for help. They also felt more comfortable giving help. So there was a lot of really empowering stuff happening, and I think as they got into the next project, away from debugging the buggy project and into their next design, it really became apparent them how much more capable they were feeling and how much more expertise they had in this domain of debugging. In general, for the impact of frustration, this is something we talk about. There's this famous theory called growth mindset that talks about persevering through frustration. And it gets back to this idea of productive failure. That frustration is not a bad thing. When things are too easy for us, we honestly don't learn a ton from them.

I think the key is how big that frustration is and what comes before and after it. And then maybe how socially visible is it. So if you're frustrated with something, if something's not working for you and you feel isolated, you may think you're the only one who has these sorts of problems, and that might feed back into ideas you have about how capable you are. But if you know that other people are frustrated, that this is a common part of learning to become an expert in a domain, well gee, I'm just part of this group of people that's becoming experts. That's pretty cool. So as instructors, we want to design environments that allow kids, even encourage them to feel frustration, make it part of what it means to be in this class, is hey, we're all going to get frustrated every now and then. We're all going to fail now and then.

We're all going to run into except maybe we are not currently capable of solving. And that is a normal process of learning. At the same time, maybe we don't want to make some things too frustrating. So for instance, I try to make it so that the computer software works so that the kids are not spending now trying to work out technical bugs that honestly I could save them that time. So we want to maybe select the kinds of frustration that we allow. Maybe we jump in at certain times to support the students, but it's always better, for instance, instead of solving the problem for them to ask them questions, find out where they're at and give them just that little bit of hope that they need to make the next step.

So again, frustration is part of the process, frustration is social, and then as an instructor, designing a learning environment for the kinds of frustration that you hope kids will experience. And then trying to figure out when it's starting to get to be a little bit too

much and then maybe intervening in those things or preventing certain kind of things like technical issues or distractions from coming up.

Dr. Sara Grimes 27:30
Another great finding from this study that I want to circle back to is how mischievousness and fun emerged as these productive, enjoyable aspects of the experience for the kids. How did this finding fit within the broader goals of the project and within your work more generally?

Dr. Deborah Fields 27:48
That's funny, you really tapped into a key theme of my research, which values mischievousness. I love it when kids are playing with the rules, finding sneaky ways around things. I know that you know about this from some of my work and virtual worlds and chief and games. Mischievousness can be such a fun part of learning. It's an area where we as learners and I'll speak this with the kids or learners or the adults are learners, we're a little bit more in control. If you have the ability to be mischievous about something, you're in a little bit of power in this situation, and that's a good feeling.

And it's fun to break the rules. A lot of what we know and thrive in in society is knowing where the rules are and knowing just where you can break them in that really beautiful way, whether it's creating a piece of music or coming up with a new solution in programming or mischievousness in the way that you're participating in a video play or just mischievousness in finding a sneaky way around the rules that doesn't break them in ways that harm others, but breaks them in a way that you learn something. That is definitely a theme of my research and fun.

I mean, come on, learning is fun. If learning were not fun, we would never grow up. We would never learn new things. And I'm not talking just about learning in school. I'm talking about learning to grow that plant or learning to style your hair in a certain way or learning to make this project or making something for a family member for a holiday or a birthday gift or something like that. Learning is fun. Learning is addictive when you do it right. And we should be celebrating those moments.

Dr. Sara Grimes 29:31
In your research and in your teaching practice, you strive to quote, break down stereotypes regarding who can create with digital media and computing, end quote. What are those stereotypes and how do your projects work to disrupt them?

Dr. Deborah Fields 29:45

So I want to talk about two types of stereotypes. One is who can create with digital media and computing. And the second is what you can create with digital media and computing. So let's talk about the who first, because that's way more prominent in the news. Okay, we have common stereotypes about who can make with computing and code. Think about some of the famous movies that have come out recently highlighting for instance, black women who are the origins of the computational revolution in the early days, coders. And women were some of the first software coders and some of the most rigorous ones. And yet, those are not part of our public idea of software engineers or computer programmers. Those images are starting to change. You can see that in popular media and stuff, but a lot of people have these stereotypes about who can create what with computing and how sophisticated they can be.

So we really want to break those down because as I said earlier, I think anyone can code. Maybe not everyone can code at the most highest levels; that takes 10,000 of hours of expertise to develop and maybe we don't help pursue that, but it's certainly within everyone's capability. So how do we help people see that they're capable of it? That's one thing and we do try to break down those stereotypes. Another thing is what you can create. So for instance, myself as an example, before I was given this task of here's this syllable computer go make cool projects for kids to make with them. I had no idea that computers could be sewn, or that lights could be sewn. I didn't know that conductive thread existed. So the idea of computer fashion objects, maybe a computational quilt or a jacket that lights up, or just everyday of things, birthday cards with a light on them, that helps break down what you can do with computing.

You can code music, you can code art. Coding is important for so many social justice efforts around the world. So we need to break down ideas that computing is just for clever puzzles and whatever, solving this puzzle and there's a certain way to do it. That is part of what the domain of computing is. And those puzzles and those solutions might be helpful for certain problems, so they might be part of the skillset you develop. But what you can do with computing is tremendous and it can be off the screen. When we are engaging kid and making these physical things that don't involve a screen, maybe you code on the screen, but it uploads to this little device that we've sewn in on fabric. It blows their minds because they've never thought, quite often they've never thought that computing goes off the screen. Wait, wait, I can do computing stuff, and it just travels with me. And the code that I wrote is there and I can make it interactive and adaptive

based on different things, interactions with squeezing touching and the light, measuring temperature, those sorts of things. That's really powerful.

I think one other stereotype that occurs is that kids often think that learning code computation or even just learning in general, has to be isolated and it's all in your head. And that is so not true. And one of the things we see when kids are writing reflections afterwards, we actually have them create, again, create a portfolio reflecting on some of the challenges they face and the things that they've learned, is they start to name what computing is in ways that counter those stereotypes. So they start saying well, it's super collaborative. Like I got help from my friend or I gave help from a friend, or I got this idea from someone else. Or my teacher came and helped me. It doesn't change the ownership and the object that they've made, but they're defining computing as creative instead of this isolated thing. They're defining computing as non geeky, as artistic and aesthetic, as meaningful for society. So there's all these aspects of breaking down stereotypes in terms of who, what and how you're making it.

Dr. Sara Grimes 33:58
I'm going to shift gears a bit and ask you the question I'm asking all my guests this season. The United Nations recently adopted a new general comment, confirming and outlining how children's right apply in the digital environment. Do you think this will have any impact on the types of opportunities kids have to learn and engage in creative making and coding or on any of the other or relationships that you examine in your work?

Dr. Deborha Fields 34:23
Well, I hope so. That's a really awkward answer, but let me try to explore that a little bit. One, I think it's an amazing document and children's rights and a part of that is a responsive abilities and digital environments have been ignored for decades. There have been some really amazing progressive designers of sites who have focused on kids that have just created so much openness and such great communities and who have helped train kids about their rights and responsibilities of things like copyright and expression and being a part of a community where you might be hiding behind an anonymous username or something like that. That has really laid some groundwork. So I think this general comment is fantastic. And for the sites that are already doing this, maybe it helps them defend and promote their work a little bit more like oh, hey, this supports the work that we're doing. This is important.

Gosh, in the US, there's so much protective work about children, so much nervousness about designing things for children. In some ways we've had designers tell us that so much money has to go to lawyers, that the environment itself is not always the best design for children, which is really ironic. So I hope that this helps maybe free up some of those costs, and maybe it won't. I certainly would like the word to get out. And here's my favorite part about this whole process. At the very least, we have over 1,000 children who helped create the interpretive document, explaining this UN general comment by children for children. So that means from the perspective of constructionism and learning by me game, we have 1,000 children around the world who are very equipped to understand and teach about children's rights in digital environments.

In fact, I wish that every child around the world could be part of an effort to come together and make a document in their languages and with their personal expression that explains the rights of children for each other. That would be really, really powerful. If we can empower children to understand their rights and know them, then that's one step towards helping those rights be achieved, because then they can speak up and children's voices can be powerful when we allow places for them to be heard.

Dr. Sara Grimes 36:58
A big thanks to Professor Fields for joining us today. Please follow the links in the podcast description to find out more about Dr. Fields's work, the publications mentioned in today's episode, as well as information on where to send your questions or comments. This was the final episode of season two, but we'll be back in the fall of 2022 for season three, which will focus on game changing research on the politics of digital technology and life online.

The Critical Technology podcast is produced by me, Sara Grimes, with support from the KMDI. Audio mix and sound design by Mika Sustar, music by Nicholas Manalo, theme song by Taekun Park. Our logo was designed by JP King and the artwork for today's episode was created by Kenji Toyooka. Please subscribe to stay up to date on new episodes and posts as they become available. And thank you for listening.