***KairosCast*, Episode 9 - Karl Stolley**

[*KairosCast* theme music]

COURTNEY DANFORTH [over theme music]: Welcome to *KairosCast.*

HARLEY FERRIS [over theme music]: Welcome to *KairosCast.*

[*KairosCast* theme music continues]

HF: Well, hello there! This is Harley with *KairosCast*. Here we are in November, hovering dangerously close to the, the what--12-week mark in the semester? Amazing. Hard to believe how time surges forward. Case in point, I'm here today to bring you the long-awaited, much-anticipated, soon-to-be-celebrated interview with Karl Stolley about his well-traveled webtext, "The Lo-Fi Manifesto." The first iteration appeared in issue 12.3 of Kairos, from the summer of 2008. The reboot, a newly revised version of this webtext, leads off the 20th anniversary issue of *Kairos* from this past spring. I'll let Karl introduce himself, and then we'll get right to it.

KARL STOLLEY: So I'm Karl Stolley, I'm associate professor of Digital Writing and Rhetoric at Illinois Tech in Chicago, also known as Illinois Institute of Technology.

HF: Thank you so much, again, for joining us today. I want to start with just setting the scene for the original manifesto. So if you could, uh, talk about, sort of, where you were, what you were doing at the time, and if there was a particular catalyst to it.

KS: Well, I was getting ready to move... I was on the job market--although, I had signed a contract to come to Illinois Tech and get ready to move to Chicago. That was in, I think, January of that year. And then it was whenever, I think it was the 12.1 issue of Kairos came out, the call for the manifesto issue was actually in there. So I was like, well, I think I think could probably write something about that. 'Cause as I was getting ready to leave grad school, I had a really great experience just teaching at Purdue. One of the courses I got to teach there was English 419, which was Multimedia Writing. And the way that I started teaching that originally was like Flash--I'm just going to teach a class on Flash! And so I taught this Flash-based class, and then, you know, every semester I offered it, I had to keep updating all my course materials to reflect the status of the Flash software and its interface. And then I started working on the redesign of the Purdue OWL and started caring a lot more about open web standards, but I was still like, no, nobody needs to write code by hand--I'll just teach Dreamweaver. So then I just started teaching this 419 class in what I now think of as a toxic blend of Flash and Dreamweaver.

HF: [chuckles]

KS: And then, as I got deeper into the OWL project, I started learning about web standards, and it's like, oh my god, like, you know... And I'm looking at the code that Dreamweaver's putting out, I'm looking at the code that I know I should be writing to make things compliant with XHTML, and there's this huge disconnect there. But the thing that really changed was that I realized that, like, yeah, it's a pain in the neck for me every semester to, like, update my teaching materials because there's a new version of Dreamweaver, there's a new version of Flash. But what about the students who are my class, and we spent all this time learning that interface, and it's no longer valid? Like, the interface has changed, the software has changed... So really, where "The Lo-Fi Manifesto" came from was to try to run backwards against the grain and say, OK, instead of trying to keep up, what are some foundational stuff that are always going to remain more or less static and constant, and that's sort of where it came from.

HF: So, you cover some of this in your "Read Me"-- What pushed the update? Was it just the 20th anniversary and a time to reflect?

KS: [laughs]

HF: Or, and I'm also curious if these, uh, if these changes attend more toward technological or ideological issues or, and I'm sure probably "both" is actually the answer.

KS: Well, Cheryl and I were, um, I can't remember what conference we were at... We were someplace in Kentucky--this was two or three years ago, and she was already talking about the 20th anniversary issue. And one of the ideas she had had early on was to have some authors come back and revisit some webtext they had written. And I was like, I'll totally do that with "Lo-Fi," right? Ever the little self-promoter of my work. And then the call went out about the 20th anniversary issue and I didn't hear anything and I'm like, OK, I guess I won't have to write that. And then Cheryl's like, "Hey, we'd love for you to reboot "The Lo-Fi Manifesto," and I'm like, "I would love to do that!" But it turned out, as with most writing projects, I loved the idea of revisiting it way more than the actual act of revisiting it, because I think it's a piece that does have a certain standing in the field--people know of it.

I think in terms of whether this was primarily a technological or philosophical/ideological reboot of the thing, I think it's a combination of both. There's the one version that's part of the reboot that's actually got the embedded commentary, and I talk about how the original four lo-fi principles--lossless, open, flexible, and interactive, um, or interdependent, rather, sorry--uh, that those were really more-- They weren't really principles. They were sort of like technological features. So, the lossless thing, for example, like, I was really big into that. But now that I'm looking and seeing how, you know, more of the world goes online with mobile technologies, Internet access on mobile devices is feloniously expensive, and it's incredibly slow. So every byte that we send down counts. But that's paradoxical against the state of, um, visual displays where now if you want to provide an image that looks halfway decent on a high-density or retina display phone, you've got to push down twice as many pixels as you did before. So [chuckles] we've got this bandwidth-quality balance that we're always trying to work through, so it's

definitely important to preserve things in a lossless state, but it's also important to make sure that in that sort of rhetorical moment when you're reaching an audience, that you're actually being as parsimonious as possible about every single byte that you're sending down the pipe.

HF: And that "Lossless" switch to "Learning" seemed like, maybe, I don't know--you might weight in--it might have been the most significant change to those four, four points that I noticed. And in terms of, uh-- You're right that this is a well-known text in the field that probably more grad programs than you can count use it at some point in there with their students, uh, when they're talking with their students about technology. Um, but I'm sure you get a lot of pushback. And, you know, my--when it was introduced to me in my Writing for New Media course, um, the class instantly polarized. There were the half that grabbed onto this, you know, and uh, joined the manifesto, and the other half that said, no no no no no, this excludes people, or whatever. So maybe you could talk a little bit about that switch from "Lossless" to "Learning" and why learning became the most prominent one in there, and however you want to connect that with, um, eh, sort of the hurdles that people need to jump over in terms of how much code should they learn, how many tools should they use or try to avoid... That's a big question and you can unpack it however you want to.

KS: Yeah... Well, I think the most polarizing thing of the original one was really my tone, you know... I was like 27, 28 years old when I started drafting it, you know, almost finished with my PhD, firebrand about to go and set the world on fire, and really-- I, I think that that never really helped the case, and honestly, it was the opportunity to go back and even out the tone of what I was saying. I mean, if you look at them side-by-side, the first version really kind of opens up in a combative stance. It's me versus the Flashes and Dreamweavers and those who wield them of the world, whereas this new one, I really wanted to make it invitational. So I used words like "embrace" and really, you know, this is for everybody. And I even make the conciliatory remark in the new manifesto, like, even if you want to keep using Dreamweaver, or you want to use whatever tool du jour, you can still use these points as a heuristic to evaluate that you're on the right path with that. I'm cool with people, you know, not coding. If they don't want to do that, that's fine. Like, I get it. But, if they are interested in doing that kind of thing, I wanted to paint a particular way forward and a particular vision of the world that moves forward. In terms of the access, I think, you know, you don't need to invest, like... You know, I'm sitting here at my desk and I've got my little, um, $5 Raspberry Pi Zeroes here. You know, this to me is the benchmark. I want to be teaching classes and doing production myself that I can do on one of these things. Because then I can talk about access and welcoming everyone. It's when we have to start paying thousands of dollars for software that is doing the same thing we could by hand, except it's doing not as good of a job for it. That to me is the real point of concern. So I understand that, yes, in demanding certain kinds of knowledge, that becomes a sort of exclusionary move, but we-- [laughs] we're academics! Like, I don't understand the fence that gets drawn around technology where suddenly all bets are off. No one would say, "Well, you really don't need to know anything about rhetorical theory. I mean, you can just kinda go in and teach a writing class, right? I mean, we've all had writing classes. Just, you know, assign an essay and, you know, it'll be fine."

HF: [chuckles]

KS: Like, we would never accept that. But when it comes to technology stuff, then it's all like, "Well, I don't have time to learn this!" And "Oh, I don't think we should really have our students do that." Like, I just-- That's the part I don't get. Like, if you could come to a reasoned, like, "No--like, I have these particular pedagogical goals, and this is how I want to achieve them, and that's why we're not doing that stuff," that's fine. Go with it. The "I don't have time" or "I don't have to" arguments--those are the ones that I just don't buy. You know, nobody has time. I don't have time. And yet, I figure out a way-- You know, I learn a new programming language every year. I'm not good at them every year, but I always want to put myself back in that position of being a beginner. Which, of course, the more that you do stuff like learning programming languages, the harder it is to truly be a beginner when you pick up a new language, because you, you know, you accumulate a certain amount of overarching precepts and things like that. But it's still a healthy exercise. So... And that's something I'm always concerned with with my students, you know, as they come in the door of my classroom, they don't know any of this stuff. Like, I think we thought back in the aughts like, oh, eventually our students are going to come in our classrooms, and they're going to know more than we do. It's like, they know less! Like, there's certain things that they can do and certain apps that they're comfortable working with, but if you ask them to explain the difference between a PDF, a Word document, and an HTML page, you just kinda get dumb looks. And that's most people, because we don't have to think about that. And I do want to think about it, because it matters. It matters for things like how we communicate in an age where people have mobile devices that, for a large segment of the population--particularly the poorer section of the population--that's their primary if not only means of access. So, OK, rock on hurricane with your giant bloated Word document or your giant bloated PDF file. Think about how exclusionary that is. Versus something that's written in HTML by hand, where you're making every byte of that file count, right? Nothing goes in there unless there's a real good reason for it, because somebody is ultimately going to have to download that thing.

HF: You clarify right in the beginning that this is "sometimes free as in beer, sometimes free as in speech, and sometimes, if not chosen after careful research, free as in puppy." And that has a little bit, I think, of the politics that you're mentioning right here, um... Could you talk a little bit more about that?

KS: Yeah. I put it in there because I've seen people were like, "Oh, I tried, you know, Omeka, and open-source sucks! Like, I don't wanna do that at all." And, yeah, I agree with you. I have the battle wounds of having worked with Omeka once upon a time, and I'm not a big fan. I don't like it, I don't like the code base, I don't like the way the API is structured. So I think it's careful-- I mean, the original Richard-- I think it was, Richard Stallman was the one who said "free as in speech, not as in beer" when he was talking about free software. So the idea is you have free speech, right, meaning freedom or liberty to speak, versus free beer, which is somebody just giving you something for no cost. Um, so, the free speech part in free and open source software

is always the most important thing, but then there's this sort of quip that sort of floats around Twitter from time to time. It originated somewhere in the Linux community that talks about free as in puppy. Like, "Oh yeah, here you go! Cute little puppy!" Yeah, it's cute for five seconds, and then you have to clean up after it, and you have to train it, and it's all this work.

So, throughout the manifesto, the other thread that I think is really new in there, um, is research. Like, I mention research a ton of times, and always learning stuff. I mean, 'cause that's where we run into the most trouble is when we think, "Oh, I know HTML. Pssh. No big deal. Learned it in '94. I'm ready to rock!" And people go in and they teach HTML the way they learned it in 1994 or 2002 or whenever it was that they happened to learn it. And I know HTML really well, but the thing that is always on my desktop, always open, is the Mozilla Network Developer documentation, because the spec changes and things that were impossible before are now possible. And even stuff like accessibility-- Like, people fall all over themselves over alt attributes--which they incorrectly say "alt tags," right? "Alt"--not a tag, it's an attribute. And it was an early stab at trying to provide some text-based content to go with an image, right? And so if you were in an environment when, you know, you had somebody with low vision, they can't see the image, this would be read to them, or if the bandwidth was broken, maybe you'd see this text. Now we have these elements like "figure" and "figcaption," which allow us to put very robust captioning on things. But if you just assume that, like, well, we just need alt text, all hundred characters that were permitted under the sort of rough conventions of how browsers render that thing, you'll never get to that point. So I really want to emphasize the research angle of that.

So part of that is to not just say that free and open-source software is an unqualified good, but to really think about, like, what is this thing? Like, it-- Using the accessibility example: like, if you want to make an image more accessible and present alt text with it, or at least you used to, in Omeka-- Like, you have to go into the core libraries of Omeka and actually muck around in there. Which is a huge problem, because then you have a software package that, instead of doing what I want it to, which is just give me the URL to where my image lives. Please. I'll write the rest in my own template. It's like, "Oh, you want an image? Well, that must mean you want an image tag, and you must want these attributes and all this stuff..." So it's this really, you know, clunky, overdetermined kind of system. It's great if all you care about is showing an image that you uploaded, right? If that's as far as your concern goes, great. It does it, no question. The second you start worrying about, like, is this accessible? Is this presenting the image exactly the way I want to? Then you have a choice to make. Then you can go in and start mucking about with the source code, and then the next time there's an Omeka release, you're going to end up having to either rewrite your work, trace it over the top of any changes they've made, or hope for, you know, version control or something to make it a little easier on yourself, um, or you're going to do what most people do, and that is to never update. And then you're going to get a pharma hack or something like that. So it's really important to, regardless of what it is, like, how many pharma hacked Wordpress websites are there in the world? Like, there are so many of them. And actually Wordpress has changed its entire architecture in its most recent release, that they're trying to get away from people running their own MySQL databases, and they want to do

everything through an upstream Wordpress API. OK, fine, that's great. We've solved that particular security problem, now Wordpress can be in charge of that. But then you lose control of your content, and you're dependent on the continued existence of Wordpress. For me, like, and I know that this is really sort of luddite and old school, but if I can't kick the cord out of the wall that the machine is hooked up to, I don't really own it, right? Like, I've got server cloud storage space all over the place, and I keep stuff there, but man, I've still got a backup that I can pick up and run out of the house with when the house catches fire.

Like, I'm really-- Anything that we really care about deeply, there needs to be multiple copies of it, and you need to be in absolute, full physical control of it. I love cloud storage stuff, I love GitHub, I love my Dropbox account--don't trust any of it. Because Dropbox could fold tomorrow. They've already folded other parts of their services, like, I used to love their mail app--it was my favorite thing on iPhone. For quick, like, sorting all of my mail into, like, four or five big buckets that i could then in a fine-grain way go to. Well, last December, like, "Sorry, um, come February 1, no more mailbox app." OK, that was unfortunate, but my mail is backed up in multiple ways, and I can move to a different client, and so that's the other side of this, is to also think about not getting too attached to any technology. Like, which is why I like to work on Raspberry Pi sometimes. Like, if you can pick up and be that kind of nomadic person, it does wonders for how you think about production and-- So getting too invested in a particular piece of software, I don't care if it's open-source, closed-source, for pay, free--that's where we run into some real danger. So that idea of, like, being able to research this stuff and understand, sort of, all that you're buying into when you decide to use a particular piece of software, that was a really important point.

HF: Well, and speaking of huge, kind of, monolithic technologies that we become dependent on and don't think about, uh, how do these concepts transfer to thinking about, um, like, content delivery platforms, like YouTube and Tumblr or Pinterest. Are there ways that we can pull these heuristic in to analyze the platforms as well as our production software?

KS: Yeah, well, I'm-- comfortable so long as we can say things like YouTube or Vimeo or, you know, things in that category, like, under the broad category of "video hosting site." They do a wonderful job; I use YouTube-- I don't do a lot of public stuff, I've got a lot of unlisted videos just because I--believe or not--do have moments where I'm like, "Ahh, I don't know if I want this public." So like, you know, videos from when I teach or things like that, I'll sometimes post that stuff up to YouTube. It's a great and valuable thing. I still have my raw, off-the-camera video backed up on my own. So, yeah. Use the service, understand that there is absolutely no guarantee, um, that it's gonna be there. I mean, YouTube is a wonderful thing. It's incredibly convenient, it was, you know, a pioneer, in being able to deliver video to mobile devices, you know... I give huge kudos to the YouTube development team for getting rid of Flash and going to an HTML5-driven thing. I would love to seem them [chuckles] give us a native solution so you can just drop a video tag in your HTML, rather than have to have your goofy iframe and import all of that nonsense from it, but still, it's a good thing. I don't trust it. You know, I don't trust it to be there, I don't trust it the way I would trust a server I control more fully, and I certainly wouldn't

just, like, delete any of the videos that I've uploaded there, but you know... They're great things, and we have people in our field that do amazing stuff with YouTube. Um, I would never begrudge anyone for using YouTube or for creating videos for things like that. It just-- If the new "Lo-Fi" is to be a heuristic, you know, just be careful. Do your research. Know what's gonna happen. And video is a particular problem, particularly because we've got these high-density displays. Like, I'm sitting here in my house looking at my 5K iMac. Like, that's crazy that, you know, you could just put a 4K screen on my screen as, like, a window. I can't even wrap my head around that. But, you know, then we come upon, sort of like, all sorts of limits on processing power and compression and codecs and, um, just the raw pixel dimensions of these things, you know. That's why it's really important-- that's the lossless part, like, yeah, keep your sources files, because you just never know what other kind of, uh, output you're gonna need, you know, later on down the line, so...yeah.

HF: I, uh, I did chuckle out loud at your point number five: "If a hi-fi element seems necessary, keep researching until you conclude that it isn't." Which is a change from before, you know, with sort of the caveats of using it.

KS: Mm-hmm.

HF:So, a question for, I guess, toward praxis for people listening who are using technology in classes to teach students moves, um, with texts that are maybe meant to be ephemeral or non-sustainable--

KS: Mm-hmm.

HF: Um, are there things they should be thinking about in terms of, sort of, implicit arguments that this is good software, or different things like that for using real tight-bound hi-fi proprietary softwares. Where do you fall down on that regarding, uh, teaching moves versus teaching code or teaching lo-fi stuff?

KS: In the original manifesto, like, technically, technologically speaking, like, everything in that more or less held up. There wasn't anything in there that was really, wow, that's no longer true. There's just things that are truer. And the part that was the least true anymore was all of that stuff in the hi-fi section. All of that stuff has now been replaced with a good old lo-fi solution. And of course, those were some of the biggest production problems that haunted the web. The problem of rich typography: solved. CSS font-face, and now we've got some good free and for-pay font solutions. And by the way, like, I am all about paying for stuff. Like, you know, as much as I love free and open source, I give my however many bucks a month to GitHub. I give my money to Adobe TypeKit and fonts.com. I subscribe to this stuff and I use it in my projects because GitHub is so important to what I do that I want-- Even I don't really need that many private repositories, like, I wanna support the cause. And with Type Foundry, like, I don't want the world to just have to rely on what Google Fonts is able to produce. So it becomes a matter of--especially now... I don't think I could have written that line in 2006 or 2007 when I was

working on the manifesto. Like, you really, you know--if you wanted to use rich typography that was accessible, you had to go the Flash route. There wasn't an option. But the growth of standards, which is driven in large part by mobile access, you know. And deciding that, no, we're not gonna have Flash and we're not gonna have Java applets and we're not gonna have all this other, kind of, bloated stuff on our mobile devices. That's really helped to push standards in a particular direction.

HF: Thinking about low barrier of entry for maybe teachers, uh, who wanna expand some of their offerings... Are there codes or tools that you recommend teachers check out these days?

KS: Um, Git to me is the most important technology I've learned in the last 10 years, and I fully suspect that in another 10 years, I'll say it's the most important technology I've learned in 20 years. Whether or not you decide to use GitHub, learn Git. Version control is such an important thing--I mean, I reserved an entire manifesto point just to talk about version control. If we want to talk about digital writing and sort of what's the trouble with it, it's that there isn't a good mechanism to do revision. Like, our revisions are so surface-y. And we have these sort of bifurcated curricula--and particularly, like, in professional writing or something like that. Like, we-- Most professional or technical writing programs are going to offer some kind of usability testing class. And so we prepare our majors to go out and do these wonderful usability tests, uncover all of these problems that just need to be solved and the site'll work better, but then they can't actually implement any of them because, god forbid you touch anything with the project because you might break something you didn't intend, and so it's like, well, we know it's not usable; however, we're not actually gonna be able to fix it because, you know, who knows what would happen if we touch the, the source code. I feel as strongly about this not because I've been writing code since I was born, but because I came up through teaching through Flash and teaching through Dreamweaver and using Track Changes in my comp classes when I used to teach comp in grad school, you know. I-- I know exactly what that experience is like, and it's horribly cliche, but it really is never too late to sort of pick this stuff up.

But, at the same time, I think it's the kind of thing that you've gotta do it full bore. Like, if you only decide to use Git for one project that you're working on a couple days out of every three months, you're just never gonna learn Git. And Git was one of those things that, like, really learned myself. Like, there was no good example. There was Travis Swicegood's book that I've picked up kind of midstream, Pragmatic Version Control Using Git, but I really kinda had to figure it out myself, and it was hard-going, but the only way that I made it through it was to say, OK, every article, every syllabus, every website, every grocery list I make, I'm going to do it in Git, and I'm going to just keep working at it until the commands are second nature. There's such a, a mistakenly high value placed on user-friendliness. And if we can't get user-friendliness, then by god, at least give us a cheat sheet, right? And this has come up [chuckles] in, in, uh, the class I'm teaching on human-computer interaction. Like, I've got students in there, despite being in a very technical major, um, who have never used Git before. Who've never used the command line before. Which is shocking. But they're all like, "Anybody got a cheat sheet? Anybody got a video?" And so, you know, there's a great exchange of ideas in that class and people are

sharing that kind of stuff, but I asked them the other day, like, have you ever played a musical instrument?

[*Kairoscast* exit music fades in.]
You know, taught to play a musical instrument? Most of the students in there raised their hands that they had. I said, is there a cheat sheet for the violin? You know, is there a quick 20-minute YouTube tutorial on picking up the trumpet and sounding like Miles Davis? No! And so when you talk about, like, what tools would I recommend to people, I like the hard ones. That, you know, you can play a little ditty in your editor, you can write a little HTML or something, but that'll be the same interface through which you write your first lines of Ruby code. Or you'll write in Go, or you'll write in Rust, or you'll write in Perl, or PHP, or whatever it is. So I'm always looking for the, sort of, foundational tools that can serve a lot of different purposes and a lot of different uses.

[*Kairoscast* exit music rises.]

CD: *KairosCast* is produced by Courtney Danforth and Harley Ferris.

HF: It is distributed by *Kairos,* Doug Eyman, senior editor.

CD: Our editor is Cheryl Ball.

HF: If we had interns, their names would go here.

CD: *KairosCast* is made available under a Creative Commons license. For more information, please refer to our website.

HF: For more *Kairos,* see kairos.technorhetoric.net.

[Music fades out.]