CREATE A CONTINUUM OF SKILL

So far, we've focused on making the skills of a full-blown developer as accessible as possible. But is that level of skill really necessary for every job?

This report argues that if we want to increase the odds of truly democratizing emerging tech, in every field of emerging tech we need to begin to ask two questions:

* Is it possible to create a continuum of skill in this field -- e.g., from beginners to power users to blue-collar coders to highly skilled developers?
* Can we use coding UX to reduce the work required to "level up" along this continuum?

# From Power Users to Blue-Collar Coders

Power Users

In most organizations, IT staff and their consultants spend a lot of time building systems that let users produce powerful analyses with a click of a button. And yet it's not uncommon to find plenty of users in these organizations who do most of their analysis with Microsoft Excel.

One-click solutions are great so long as users stick to the well-marked trails the solution supports. But what if your needs take you off that trail? For example, what if you need to make changes to the report's format that the solution doesn't support? You're hopelessly stuck without the help of a developer.

But with Excel, if you're a little adventurous it's remarkable what you can do as a "power user." You can start by learning a few tricks, then gradually add more tools to your toolbox as you need them. The results aren't always pretty; sometimes it feels like Excel is duct tape for data. But like duct tape, you don't need to be highly skilled to use it to solve a wide range of problems.

If we're going to create an economy where many people in communities from Harlem to Harlan County can make a living from emerging tech, we need to start designing emerging tech tools so they can do what Excel does so well: support a culture of power users. Power users don't need a computer science college degree -- or for that matter any college degree -- to possess valuable skills that pay well. In short, by designing tools that support power users, the tech industry can create a wider range of opportunities for paid tech work.

Blue-Collar Coders

On the other end of the spectrum, Anil Dash and others argue that not all programming jobs should require a full-blown background in computer science. Instead, we need to foster the development of "blue-collar coders."

Vocational-technical schools (vo-tech) provide trained workers in important fields such as healthcare, construction trades, and core business functions like accounting. For a significant number of my high school peers, vo-tech was the best path to a professional job that would pay well over the duration of an entire career. Now it’s time that vo-tech programs broadly add internet and web technologies to the mix. We need web dev vo-tech...

Put another way, our industry can grow in a very meaningful way by giving lots of young people at a high school level the knowledge they need to learn [web development frameworks] straight out of high school, or teaching maintenance on a MySQL database at a trade school without having to get a graduate degree in computer science.[[1]](#endnote-1)

Exactly what blue-collar coding will look like will differ from field to field. But the more we can create jobs that fall along a continuum of skill, the more opportunities we have to open up emerging tech development to communities our society has left behind.

# Smooth the Learning Curve along the Continuum

Recently, several tech companies have been exploring the space along the continuum of skills. They've created "low code" or "no code" tools aimed at what some have called "citizen developers."

These tools are often quite powerful. But most of them suffer from the same problem: if your needs outstrip what the tool was designed to do, the learning curve to level up your skills is too steep.

If we aren't careful, there's a real danger that these well-meaning efforts to empower people will lead to the creation of a new class of dead-end jobs. That's why any effort to create a continuum of skill needs to make smoothing the learning curve along that continuum a priority.

# Create a Continuum for Artists and Designers As Well As Coders

Emerging tech won't just require coders. The tech industry will also need designers and artists for AR/VR, robots, 3D printing/digital fabrication, and other forms of emerging tech. It's worth exploring if we can also create a continuum of skill for this work -- especially since, as is already the case with web design, some of this work may require the ability to do a little coding.

Opportunities to make a living from designer and artist work could play a particularly important role in marginalized communities. In the past, these communities have often been founts of new forms of art and culture. Both they and our society overall will greatly benefit if they are able to fully participate in shaping the world of emerging tech.

# With Automation, a Continuum of Skill Is Even More Critical

When people talk about the threat of robots/AI, they often sound like they assume automation is a one-time event. It won't be. We can expect wave after wave of automation that either eliminates or deskills jobs.

And we can't assume that the pace of automation will slow down. If anything, it's at least as likely that over time the pace of automation will speed up.

In the past, there was often a significant gap between rounds of mechanization because the tech needed to mechanize one skill didn't easily translate to another. Wheat harvesting, for example, was mechanized in the 1930s and 40s, but it took almost two decades before the mechanization of tomato harvesting became commonplace.

In contrast, it took just a few years for the machine learning libraries used to categorize pictures of cats on the Internet to begin to be used in a wide range of fields, from drones to medical systems that can analyze some types of x-rays better than most doctors can. Thanks to the Internet and Open Source, we can expect even more cross-pollination in the future.

As AI/robotics allow people to do more and more with less skill, we can also expect it will lead to the rise of new, highly demanding developer skills. That's why creating a software tool development process that iteratively creates and re-creates a continuum of work is so critical.

To do so, we will need to continually ask two questions:

* As automation allows us to do more and more with less skill, how do we create more opportunities for power users?
* As developers continue to build ever more powerful new languages, frameworks, APIs, etc., how can they be designed so gaining more skill and more sophistication in using them is easier to do?

In short, as automation continues to transform the skill landscape, we will need continual rounds of user experience design to smooth the learning curve so "leveling up" is as easy and engaging as possible.

# Hip-Hop Wasn't Created By Turntable Engineers

Some advocates for power users or blue-collar coders sound as if as if they think these types of workers won't be capable of the kind of creative work that will drive emerging tech forward. While it's certainly true that some cutting-edge emerging tech work will require a PhD, the history of hip-hop suggests you don't need to be a rocket scientist to power innovation.

Hip Hop came out of neighborhoods that had lost hundreds of thousands of jobs to outsourcing and had been devastated by urban renewal. And yet the people who lived in these neighborhoods built some of the most amazing artistic creations the world has ever seen, transforming both music and culture around the globe.

A key part of Hip Hop’s rise was a brilliant technical innovation: morphing turntables from tools for playing music into tools for making music. This innovation didn't come from the engineers who created turntables, it came from people like Grandmaster Flash. Grandmaster Flash, one of the “holy Trinity” who created hip-hop, had the most technical training of the three, and he’d only attended a vocational high school. But while Grandmaster Flash wasn’t a PhD scientist, he was a mad scientist -- a genius with an obsessive drive to experiment and create something new.

As important as it is to spread the most advanced technical skills to every community, what the birth of hip-hop shows us is that we must also empower people with a wide range of technical skills if we want to unlock the full creative and economic potential of emerging tech.

EVANGELIZE BREAKTHROUGH RESEARCH ON CODING UX

The following are some examples of what researchers in University of Washington's Code & Cognition Lab have cooked up to make coding more accessible to all: [[2]](#endnote-2)

* **Learning by Solving Debugging Puzzles**.Mike Lee created Gidget, an interactive tutor for learning the programming language Python by solving debugging puzzles. In a study of 1,000 adult learners, he found that
* People who used Gidget picked up Python twice as fast as a well-regarded online Python tutorial and learned twice as much Python as participants who learned by doing.
* Gidget "changed attitudes about the difficulty of learning to code from negative to positive in 20 minutes" -- an incredibly important finding for efforts to make coding accessible.
* **Learning by Tracing on Paper**. Benji Xie believed learners often have a "brittle" knowledge of a programming language's semantics, so they just guess what will happen when their program runs -- a surefire strategy for producing frustration and failure. He developed an innovative strategy that involves students figuring out on paper how their program will execute. His study showed that with less than 15 minutes of training in this technique, students did 15% better on computer class lab problems and 7% better on midterms -- and no one who used the strategy failed the midterm, in contrast to 25% of the control group.
* **Speed Up Learning JavaScript.** Greg Nelson created PLTutor, an online interactive textbook designed to cover all of the basics of standard JavaScript *in 3 hours*. For each core idea in JavaScript, PLTutor explains why & when you'd want to use that idea, then lets you step through code using the idea so you can see how the code executes and what impact it has. His study demonstrated that compared to a well-regarded online class, participants who used PLTutor had 60% higher learning gains.

The Code and Cognition Lab has even more dazzling work in the pipeline. In a 2018 talk, Professor Andrew Ko says they are now working on a project to create a scalable online tutor that “provides infinite personalized practice by applying program synthesis and our theories of programming knowledge." Their goal: students would be able to master the basics of a 10 week intro computer science course in just 10 hours.

There are many more research shops around the globe, many of which have produced equally impressive work.

And if you're a developer or a coding trainer, you've probably never heard of any of them.

In fact, it's possible to spend an entire career as a coding professional, reading articles and blog posts in your area of specialization, going to tech conferences, and talking with your colleagues, without ever bumping the world of research on making coding easier to learn.

It's not that some researchers aren't working with people out in the field. For example, Gidget influenced some of code.org's work. But the broader connections between academia and practitioners rarely exist. Too often this research is like a tree that falls in a forest and nobody but researchers hear it. If we're going to truly democratize emerging tech, that's got to change.

The following are some strategies researchers might try if they want their research to make some noise.

WARNING: this chapter gets into the (geeky) weeds, so if you aren't a techie and don't work in tech, you may want to skip to Chapter 2.4.

# 1) Embrace Tech Groups

More researchers need to start working closely with groups within the tech world. For example:

* **Pythonistas**. The Python programming language is often used for teaching beginners because it is one of the easier languages to learn, and there are many people in the community of "Pythonistas" who are passionate about spreading Python. If researchers could build a bridge to these people, they might find a very receptive audience for the results of the research. They also might find people who would be interested in helping them with their research.
* **JavaScript Evangelists**. The world of JavaScript has undergone several transformations in the past decade or so. If researchers were an active part of that subcommunity, they might find strategic moments where they could help shape its direction. And there plenty of websites for web designers and developers, such as Smashing Magazine and CodePen, where researchers could find influential voices in the JavaScript community who are receptive to research on making coding easier -- after all, they're already strong advocates for website UX.
* **Tech Organizations for Women and People of Color**. The past decade has also seen the rise of a wide range of tech groups aimed at making coding and other tech more accessible to women and people of color. Researchers who shared the same passions could undoubtedly find people receptive to their work, some of whom would also be interested in collaborating.

# 2) Focus More Research on Libraries/Frameworks

One of the most striking features of academic coding UX research is that it is primarily focused on languages -- either on modifying existing languages or on creating new ones. But these days, libraries, APIs, and frameworks that are built on top of programming languages are as or even more important to accessibility than the underlying language.[[3]](#footnote-1) For example:

* If someone says they are using Python for machine learning, odds are they aren't spending much time directly working with plain-vanilla Python. The bulk of their work is using Python libraries with names like Tensorflow or Pytorch, and they may also be using a Python library called pandas to "clean up" their data so it's ready for analysis.
* If someone is developing VR/AR, they probably aren't spending much time with plain-vanilla C# or JavaScript, they're mostly using a C# framework called Unity or a JavaScript library called A-Frame.

If researchers focused more of their time on influencing libraries and frameworks, they might have a bigger impact.

**A) It's Easier to Change Frameworks/Libraries**

* Getting Python coders to agree to a fundamental change to their language or to switch to a new language that's designed to be easier to learn? That's a pretty tall order. Changes in a language, for example, can end up breaking a staggering number of scripts and software products. Convincing them to switch from one Python data visualization library or machine learning library to the library researchers have helped make much more accessible? That's a much easier sell.
* It's also common for developers to switch libraries/framework as better ones become available, so there are more opportunities for researchers to get their ideas out. And if their UX ideas help propel the use of an up-and-coming library, other library/framework designers are more likely to adopt these ideas.

**B) It's Easier to Build an Easy-to-Use Library on Top of another Library**

* If users are already using an emerging tech library, it's often pretty straightforward to build another library on top. For example, fast.ai's library that underlies their terrific course on deep learning is built on top of the Python library Pytorch.[[4]](#endnote-3) This strategy can be a great way to either eliminate the hard parts of the underlying library or shield users from its weirder bits when they are first getting started.
* This strategy lends itself to an iterative, organic approach, trying out a researcher's ideas and getting quick feedback -- or even working with members of the community to critique libraries as they are learning them and to provide suggestions for improving them.
* Researchers who build an easier to use library on top of an existing library also have an almost built in audience for their work, which improves the odds they'll have an impact.

# 3) Make a Communications Plan

When medical researchers publish their findings in a journal, either their team or their institution has at least an implicit communications plan. Sometimes it involves a sophisticated press strategy, including having PR staff attempting to get the researchers booked on TV and radio. At the very least they put out a press release.

But in the field of UX coding research, this appears to happen only rarely. Not surprisingly, the results of most UX coding research never reaches the audiences who need to know about it if it is going to have a real impact.

UX coding researchers don't need communications strategies as sophisticated as medical researchers; they aren't trying to reach a general audience. But even a little communications work could go a long way -- especially if researchers build relationships with tech groups.

# 4) Learn from Extension Services

Most UX coding researchers work in universities and colleges. And in many of those universities and colleges, there are staff who have a solid track record of making information more accessible: Extension Services. Coding and agriculture face different audiences and challenges. But there are surely valuable lessons UX coding researchers could learn from the decades of experience of their agricultural counterparts.

# The Tech World Must Play Its Part

If researchers' work is going to have a significant impact, ultimately it's up to them to make it happen. But those of us who aren't researchers can also play a part by seeking out their research. Large tech companies can play a particularly important role, as they have the resources to find interesting research and figure out how to implement it.

And if university and college researchers begin to build a more vibrant relationship with the world outside academia, demonstrating the practical worth of their research to a wider audience, large tech companies could help fund research that might have a strategic impact.

For example, it's relatively difficult to get research funding on making machine learning easier to learn.[[5]](#endnote-4) Given that most big tech companies are pouring huge amounts of money into machine learning/AI, it's in their self-interest to either fund this research themselves or lobby to get it funded.

1.4 CREATE INSTITUTIONAL SUPPORT

FOR CODING UX

Say you work at a small startup or a big tech company and you're frantically racing to build the first version of a new emerging tech library. You want to democratize your tech and you care about the community. But you can barely keep your head above water. How would you find the bandwidth to take on community-oriented coding UX or creating a continuum of skill? Where would you even start?

Or perhaps, as many popular tech libraries first began, your project is something you work on after hours as a labor of love. If you eventually succeed, you'll probably have a small army of Open Source volunteers helping you out. But for now, you don't have the resources to take on community-oriented coding UX.

If we are going to truly democratize emerging tech, we need to bring about a cultural change in the tech industry so techniques for smoothing the learning curve are as common for emerging tech development as UX is for designing websites. Getting there will require institutional support. The following are some ideas about what that support might entail.

Large Tech Companies

For Google, Microsoft, and other large tech companies, there are two keys to making community-oriented coding UX and other techniques for smoothing the learning curve an inherent part of the process:

* **Resources**. For any emerging tech project that is creating languages, libraries, frameworks, etc., the company should ensure they have the resources they need to fully embrace community-oriented coding UX and create a continuum of skills -- e.g., UX staff who work on the project and/or a liaison with a community group that's providing the UX work.
* **Rewards**. Companies should ensure staff are rewarded for making community-oriented coding UX a priority.

Venture Capitalists

While we certainly wouldn't expect all venture capitalist funds to embrace techniques for smoothing the learning curve, if a significant number of them began expecting that their emerging tech startups would use this approach, they could radically transform the tech community.

Foundations

* **Fund UX Services for Small Fry**. Foundations could provide grants to community groups to provide free coding UX services. These services could be targeted at unpaid labors of love, very early startups, and other individuals or organizations that have a significant audience for their open source coding tool but don't have institutional support. Foundations might also facilitate the creation of a pool of these resources that's funded by foundations, big tech companies, and other actors.
* **Shape Software Project Grants**. In some cases, foundations directly fund software projects -- e.g., Jupyter/IPython notebook, which has become a major development environment for data science and machine learning. They could begin to require that grantees follow a community-oriented coding UX approach and, if relevant, a strategy for fostering the creation of a continuum of skills.
* **Leverage Nonprofit Software Purchases**. Many nonprofits purchase software using foundation grants. If foundations banded together, they could have a significant impact on the ecosystem of tools used by their grantees.

Colleges and Universities

* **Shape Student Startup Culture**. For colleges and universities whose computer science departments teach their students how to create startups, if they encouraged community-oriented coding UX and other strategies for smoothing the learning curve, it could substantially speed up the adoption of these techniques.
* **Leverage University Resources**. Colleges and universities are especially well situated to have an impact, because not only do they train many of the students who will create startups and go to work at tech companies, but they also create research on UX techniques and coding UX. Many colleges have agriculture extension services as well as researchers and centers who have extensive experience collaborating with the community. If they leverage all of these resources, they can provide a much richer experience for their students, who will go on to help shape tech industry culture.

1. Anil Dash , "The Blue Collar Coder," http://anildash.com/2012/10/the-blue-collar-coder.html [↑](#endnote-ref-1)
2. Andrew Ko, "Stanford Seminar - Learning to Code: Why We Fail, How We Flourish," https://www.youtube.com/watch?v=mkzHIhKaUX4 [↑](#endnote-ref-2)
3. Although there are important differences between libraries, APIs, and frameworks, for the purposes of this report you can treat them as interchangeable. The distinction that matters here is between a programming language, such as JavaScript or Python, and tools such as libraries, etc. that are built on top of a programming language in order to extend its capabilities. [↑](#footnote-ref-1)
4. For a great overview of why fast.ai decided to build a library on top of Pytorch, see Jeremy Howard, " Introducing Pytorch for fast.ai," fast.ai blog, September 8, 2017, http://www.fast.ai/2017/09/08/introducing-pytorch-for-fastai/ [↑](#endnote-ref-3)
5. Andy J Ko, "We need to learn how to teach machine learning," https://medium.com/bits-and-behavior/we-need-to-learn-how-to-teach-machine-learning-acc78bac3ff8 [↑](#endnote-ref-4)