>> It was an 8086.

At the time, 8086 was already out of date,

but I had one.

It wasn't until about senior year in high school when I realized what I could do with it.

I have a little brother, and so I
made it so that when he tried
to log into the computer,

it would just beep really loudly.

And then it would put up

this huge ASCII
warning error that

was like "Intruder, intruder."

>> "Intruder alert.
Intruder alert."

>> "This intrusion
has been logged."
It wasn't actually logged, but it looks scary.

Hi, everyone. Welcome to Behind The Tech.

I'm your host, Kevin Scott, Chief Technology Officer for Microsoft.

In this podcast, we're going to get behind the tech.

We'll talk with some of the people
who made our modern tech world possible and understand what motivated them to create what they did.
So, join me to maybe learn a little bit about the history of computing and get a few behind-the-scenes insights
into what's happening
today. Stick around.

Today, I'm joined by
my colleague, Christina Warren.

Christina is Senior
Cloud Developer Advocate

at Microsoft. Welcome, Christina.

>> Thank you so much. I'm
happy to be here and I'm

excited to learn more
about today's guest.
>> Yes. So, we're having

Alice Steinglass
on the show today.

Alice is the President
of Code.org, which is

an organization
doing stuff that's

super near and dear to my heart.

So they are trying to teach
every child how to program,
and they partner with teachers

in K through 12 across the country and

increasingly across the globe to try to help

make computer science a part of the K through 12 curriculum.

>> You have a lot of similarities with Alice because

you also have an organization that has a similar mission?
>> Yeah, I do. So, one of the things that I've been trying to do, and like this podcast is a little bit of a reflection of that,
is to show the truly diverse set of faces and tell the diverse set of stories that lead
people into computing and what their careers look like.

Because when I look around me and like I see all of the amazing people who are helping to build the technology that we all depend on, it's not this monolithic thing.
There are just so many different folks, genders, and ethnicities, and folks who came from like their parents were college professors to folks like me who no one in their family went to college, and it was an interesting quirk that they ever found their way into computing.
One of the things that we know both from my work, the Behind The Tech, and my family foundation is that the earlier that you set the spark of interest in a child and the more of the barriers you get out of their way.
to pursuing that is

an interest and maybe
ultimately as a career,

like the happier, more
successful they'll be.

>> Definitely. I
think a lot of people

have an orthodox path
into getting to tech.

I got into it because I had
that sheer force of will.
Behind the Tech with Kevin Scott

>> Yeah.

>> But I think about kids that I went to school with and if they'd had those opportunities that were accessible to them like the way that code.org is making things accessible now, how different things might be.
>> Yeah. Sometimes

your journey can

be sensitive, so to speak.

So, like one thing can
completely change your path.

Like with me, I was
lucky enough to get into

a science and technology
high school when I was a senior.

If I hadn't had that experience,
I don't know what my career would have looked like, whether or not I would have chosen computer science as a major when I went to college or maybe even whether I went to college at all. So I think what that tells me is let's do everything.
humanly possible to expose kids to as many of these opportunities as possible.

It's not that I think everybody should be a computer scientist,

but you should at least have the opportunity.

>> Definitely.

>> Thanks for
chatting, Christina.

We'll reconnect later at the end of the show.

Coming up next, Alice Steinglass.

Alice is the president of Code.org.

Her teams build curriculum tools and software to support introductory computer science classes for students from kindergarten.
They also partner with education and software companies across the industry to run the Hour of Code, a global movement reaching tens of millions of students in over 180 countries.

Alice, welcome to the show.
>> Thank you.

>> So, one of the things that I would love to start with is your journey.

So, how did you get into computing?

>> I'm so lucky to be here, but my journey was not the journey that a lot of people had.
I didn’t play with computers from the time I was little.

I didn’t take them apart for fun.

I actually got into computer science because my school taught it and-

>> This is your high school?

>> Yeah, my high school.

I didn’t really know what
I was signing up for.

I was into math, I was into other things.

I said, "Okay, I'll try this.

I hear you can make things with it."

I took a class and I loved it.

I had a final project, where I built a game called Snake,

which similar to Tron
what everybody built it back then.

But I finished it, it was fun.

I tested it, I tested it,

and then my teacher ended up staying up like all night

testing it and found out

that the high score could go even higher.
It broke if you had more than

like five digits in
the high score and I said,

"How did you find that?"

He said, "We were
playing it all night."

What other class do you get to

make something where your
teacher plays it all night?

>> Yeah. So, was it
the whole thing,
was it the technical challenge of writing the code,

was it the fact that you made something that someone was a little bit addicted to?

>> I think it's all of that.

I think for me it's like the best of Math and
Art and English, and all of that put together.

I always liked Math, but Math, most of the problems have an answer. There's no creativity.

Here's a challenge, can you figure out how to find the tip-top of this curve or something?

In computer science, it had
that same logical backbone,

but the problems were open-ended.

You're never done with a project, and even in real world.

When we're building software, we're never done with it.

So, we're always making it better,

you can always improve it,
where you can create something.

I loved art, I love creating,

and I think computer science is like creating both logic,

and then it gets to move at the end, which is cool.

>> Yeah. It's super cool.

So, when did you get your first computer?
>> When did I get my first computer?

I had a computer when I was younger. I was lucky.

My father's office was selling off cheap computers, older computers.

So they sold them to the employees for I think it was like $50. He got me
an old computer.

>> Wow.

>> It was an 8086.

At the time, 8086 was already out of date,

but I had one and it just sat in my room.

I didn't code it. I didn't program it.

I used it. I've wrote papers on it.
It wasn't until about senior year in high school when I realized what I could do with it.

Once I figured out computer science, I did go back and code it, but I'll have to tell you.

So, one of the first programs I wrote for it, I had a little brother
and I made it

so that when he tried to log into the computer,

it would just beep really loudly. And it would put up this huge ASCII warning error

that was like, "Intruder, intruder."

Then, of course, it named him because there's
no other possible intruder in

my house other than my brother.

So it would say, "Seth,

you were trying to break
into this computer.

This intrusion has been logged."

It wasn't actually logged,
but it looked scary.

>> Yeah. This is the thing that
really amazes and interests me about computing.

There's this notion I think in the minds of a lot of people that there is one stereotypical path that you're like a nerdy teenage white boy and you get your machine when you're 13 years old, and you start writing your first code.
This notion that you have to be a prodigy to get into computing.

But when I actually talk to people,

everybody’s story is so different.

Anders Hejlsberg, who we interviewed in a previous episode,
he didn't start coding
until he was in college.

So, some people early,

some people late, and

the motivations are
all over the map.

Some people just love
the creative aspect,

some people love the fact
that they can make
the machine do something.
My kids love that. It's like,

"Okay, I can tell
the machine what to do.

I can't tell mom
and dad what to do,

but the machine
will listen to me."

>> Yeah, absolutely.

I think it was a little
intimidating for a while
because there's this language that goes around computers,

and there's this barrier where you feel like if you don't speak the language then you probably can't learn computer science.

But the truth is you absolutely can learn it,
and the language is just a false barrier.

I went to college.

I heard all these guys talking about things like bulletin board systems in the '90s, and it was like a thing then. They were all on it, and I have never been on a BBS in my entire life.
You think, "Okay, BBS is some technical world, and I can't possibly code if I don't know what a BBS is."

It turns out that a BBS is just like Reddit, but in the '90s.

>> Yeah.

>> You absolutely don't need to use Reddit to do computer science.
I mean, I love computer science. I love the logic.

I love the challenges. I love building.

But to this day, I still have not done BBSs, and it's okay.

>> It's super okay.

>> Right, and it's this language thing.
It's this language barrier that just, it makes you feel like you can't but you absolutely can.

>> Yeah. So, from your senior year where you took your first computer science course, what was next?

>> So, I went to college and at that point, I was already into it.
Actually, that's not just me, that's really common.

What you see is that women who take AP Computer Science in high school are 10 times more likely to take it in college.

That's one of the reasons we're fighting so hard to get computer science
offered in high school is because it helps dispel these notions.

It helps make you feel like you can do it.

So, I went to college and I knew I wanted to take Computer Science.

I majored in Computer Science in college.
I did the typical startup on the side.

>> What was your startup?

It was dynamicfeedback.com.

Yeah. We partnered with a professor who is doing management consulting and worked on how do you help people take 360-degree surveys to
learn how to be better in the workplace.

It was interesting, it was fun.

Like everybody's first startup, we totally underestimated the amount of code that we need to get written to do what we thought we
would need it to do,
we worked all night.

285

00:10:20,580 --> 00:10:22,620

Part of it for me was
the experience of

286

00:10:22,620 --> 00:10:25,665

learning that a company
is more than just code.

287

00:10:25,665 --> 00:10:27,360

We had to figure out things like

customer support and lawyers,

288

00:10:27,360 --> 00:10:29,620

and I had to find
a space.

289

00:10:29,620 --> 00:10:32,980

>> Really unsexy stuff.
>> Yeah. Where we actually go to sit.

So, that was interesting.

I ended up coming out to Microsoft after that and I worked on.

>> How did you decide on Microsoft? What year was this?

>> This was 2001.
>> Okay.

>> I was working on the first version of Xbox.

>> So, super exciting.

>> It was super exciting, and then I got to work on the first version of Xbox Live.

What's weird is I'm not a hardcore gamer,
but it was still a really interesting set of problems.

I think, sometimes not being a hardcore gamer actually helped.

I was working on the high score system for Xbox.

I kept talking to people and everybody had a way we should do high scores.
They have to work
like this because

they work like this is
my favorite racing game.

They have to work like
this because they work

this way in my favorite
shooting game.

Coming in as a neutral
person I said, "No,

I'm going to look at
all the games and
understand how high scores work across everything."

I went and played 50 games and learned about how high scores worked in every game and talked to a lot of people, and then, designed a system to allow any game on Xbox to use the Xbox high-score system.
So, it was interesting.

>> Yeah.

>> Interesting work.

>> Did you have a course charted as you were going one thing to next?

The reason I ask is, I think,

everybody has such a different path
through their career in computing,

and they're all good and interesting.

>> I think in retrospect, I could probably tell you a story.

But the reality of it is that I think a lot of it is happenstance,
a lot of it is you don't know.

>> Yeah.

>> You try something and you find

out you like it or you don't.

The one thing that
I would recommend

to young people who are starting

try some different things.
I think you can get stuck in one thing pretty easily and not even have a plan that.

The easiest time to switch and try some new things is in your 20s, when you're not an expert yet in one particular field.
So, one of the things I did do was I tried different technologies.

So, I worked in Xbox,

I worked on Live,

I was in charge of all of the APIs for Xbox Live across the board,
I went from that to looking at the Toolchain that developers use and working on XNA before it was XNA.

Then I went from there, I said, 

"What's the opposite of everything I've ever done?"

Right. I've been working on more the APIs, 

I haven't touched enterprise software and
enterprise services and I just want to know what the other side looks like.

>> Yeah.

So, I went to Office, I went over to Microsoft Project partially because it was just a very different space.
I figured this was a good time to learn about a different space.

I had a lot of people who thought it was the most insane thing they'd ever heard.

Right. Why would anybody leave Xbox on purpose to go work on Project?

But I actually found it
really fascinating and interesting.

Understanding about how do companies make purchases, and what does it mean to sell and to enterprise sales,

and how do we make workplaces more efficient,

and what is business software look like.

I thought it was
a really fascinating space.

It sounds like one of the things that has driven a lot of your journey is just curiosity.

You've explored a bunch different things, startups.

>> Yeah.

>> Ton of different things at Microsoft.

Were you the kid that was
taking all your mom's stuff

00:13:43,860 --> 00:13:47,230
apart, or asking
five million questions?

00:13:47,230 --> 00:13:49,870
>> I mean, yes, but
I think we all are.

00:13:49,870 --> 00:13:50,840
>> Yeah, you think so?

00:13:50,840 --> 00:13:53,740
>> Yeah, I think kids
are naturally curious.

00:13:53,740 --> 00:13:55,320
I think we all want to learn.

00:13:55,320 --> 00:13:56,615
I think we all want to do that.
I think there are barriers that hold us back, and some of those barriers can feel more real than they are, especially in tech.

It's a booming space. There's a million jobs right now. Everybody's looking to hire.
like talking to young people in tech.

Sometimes they're afraid to make the choice,

to try something new or to change.

But, it's a false barrier they've put on themselves.

>> One of the things that really

strikes me about the industry over the past,
let's just say,
10 or 15 years is,

I think, in some ways
we've gotten more complex.

The number of
programming languages,

the number of frameworks,

the whole ecosystem
is just bigger.

But, in a very real sense
it's easier than it ever has
been to go make something with code or with technology.

When I was in college,

folks had this notion like, "Oh, my God.

Coding is so hard,

you have to go get this degree,

you have to practice."
To get really great at anything, all that's true,

but my kids can go make interesting things right now without a Computer Science degree because the tools that they have are so powerful.

Is that something that you're seeing helping students get into computing?
>> Absolutely. There's a level of relevance, right?

>> Yeah.

>> When I was a kid, I made a game from my calculator that was [inaudible].

I made a game and I also made it formula solver cheat sheet kind of thing.
Behind the Tech with Kevin Scott

00:15:27,815 --> 00:15:28,165

>> Right.

00:15:28,165 --> 00:15:30,225

>> But helped you with your physics formulas.

00:15:30,225 --> 00:15:32,925

This wasn't going to be the thing that took over America.

00:15:32,925 --> 00:15:33,395

>> Right.

00:15:33,395 --> 00:15:34,760

>> But it was popular,

00:15:34,760 --> 00:15:39,250

among all the students in my class. Right?

00:15:39,250 --> 00:15:41,090

I think there's
We see kids making games.

There are some of those things are just not that complicated, right?

>> Yeah.

>> So, students have the potential to make things that are definitely cool.
They're not as complex as an Xbox game, but they're cool.

But, you also see that there's a lot of space for things that are locally relevant.

Some of these kids' apps, there's one with their teacher's face, you could feed the teacher ice cream,
but the teacher got
a kick out of it,

and it's fun, and it's cute,

and it's relevant
in that classroom.

It's relevant in that school,

your friends are all
going to try it out.

I think it gives you a taste
of something without
having to be an amazing artist,

just like anything else, there will be steps.

>> Also, talk a little bit about what you do right now.

So, you're the President of Code.org.

So, tell us a little bit about what Code.org does.

>> So, we build curriculum, we do
professional development
for teachers,

we do advocacy work,

but our goal is that
every child should have

the opportunity to take a
computer science class in K12.

I was shocked, especially
from the tech industry.

I was shocked to hear that
most schools today don't teach computer science,

and it's not even that most kids don't take it,

it's their school doesn't teach it at all.

So, even if they want to take it, they can't.

This disproportionately affects students in high need schools.
It disproportionately affects underrepresented minorities and women who are discouraged from taking these classes. And the result is that because they never get this introduction in K12, it's really hard to start after that. It's really hard to start in college.
So they may never go into the field.

And even if they go into another field, they don't have that background in computer science.

So, our goal is that every school should offer this course, so that every child has an opportunity to take it.
At this point, we're the most popular computer science platform curriculum in K12 in the country.

About 25 percent of students actually have an account on Code.org.

So, we're reaching a lot of students
but there's a long way to go.

>> Yeah. So, how early should we be teaching kids computer science?

>> So, this is totally different from how I started, but our recommendation is actually to start in elementary school, and there's some good reasons for doing this.
Let me start by talking about how we teach about biology today, because I think it's a really good analogy for how I think about computer science education.

So, every child when they go to elementary school gets to learn that they have bones,
they have a digestive system,

just the basics of how

does my body work.

We don't do that because
they're all going to be

doctors or nurses or EMTs.

We do that because they're going
to live with that body for

the rest of their lives and
they should know how it works.

When they go to middle school maybe they learn more about it.

In high school, a kid can take Biology or AP Biology.

Even after they take all of those courses,
they're still not qualified.

I don't trust a high school student who's taken AP Bio to do anything to me.

So, there's still more work if they want to be a professional in the field,

whether it's a nurse or a technician or anything.
Computer science is the same way.

Every kid is going to be surrounded by technology their whole lives.

We have our phones in our pockets, who knows where they're going to be when they grow up.

The same way we get to know that we have
a digestive system,

they should understand,
what is the Internet?

What is the Cloud? What is data?

How does this phone work?

It's not a magic box
that does magic magic.

It's a computer, and what
is a computer, right?

These are just basics that should
be part of our education system.

>> Right.

So, I think of it in a very analogous way.

In K5, we get to teach the students,

what are these things? What is technology?

Then, when they get to middle school,
maybe they take more.

If they're interested, they can take an AP Computer Science class in high school, and at the end of that, they're still not a programmer.

They're going to go on and take a two-year degree. They could take a four-year degree.
They can become a lifelong computer scientist.

But, no matter what they do in life,

it's useful to know how computers work.

>> Yeah.

>> So, the same way we teach our kids how the body works,
teaching it in elementary school.

There's another reason to start so young, and that has to do with supporting diversity in computer science.

What we see is that women tend to become less interested in the STEM fields around the middle school, early high school.
In computer science, it’s between about 12 and 14 when they lose interest.

So, what we want to do is reach them before that year, so that while they’re still interested in learning these things, we can show them what it is,
so that if they're interested,

they can keep going.

So, there's a bunch of pieces here, part of it is encouraging them, thinking that they'll be good at it, getting that encouragement.

If they're very confident in their ability to do it,
they're four times more likely to go into computer science or take computer science classes than if they aren't.

Girls, right now, oftentimes, they don't get this opportunity in elementary school,

and so what happens is,

when they're thinking about taking it
in high school or middle school,

they do it just based on the zeitgeist of what people tell them that they're going to be good at.

>> Right.

>> Right? Unfortunately, what we
see is that they're often told they won't be good at computer science.

Teachers are two and a half times more likely to tell a boy that he'll be good at computer science than a girl. And it's not because they're against it.

These teachers are supportive, they care,
it's just these cultural norms are embedded in our society.

>> Well, and kids are also pretty good pattern matchers.

One of the things that I've noticed disturbingly with my own kids,

I've got an eight-year old and a 10-year old right now, and very,
very early when they were three, four years old,

they would look around at the world and start making these classification decisions.

It's okay, this is a boy thing and this is a girl thing,

and this is without anything in their household telling them that thing A and thing
B has a gender association with it.

It's just them sorting things out.

One of the things I love about what you all are doing is there's this bootstrapping problem that I think you have to solve where we just need more three and
00:21:40,700 --> 00:21:43,590
four-year-old seeing
seven and eight-year-olds

00:21:43,590 --> 00:21:46,750
being successful in a
computer science curriculum,

00:21:46,750 --> 00:21:47,860
so it helps them

00:21:47,860 --> 00:21:49,350
decide to do that when
they're just a few

00:21:49,350 --> 00:21:52,435
years older and up
the entire stack.

00:21:52,435 --> 00:21:53,690
>> That's absolutely true,
and you see it when you
go into the classroom.

So, you take a bunch
of second graders.

They don't have a stereotype

yet that computer
science is a boy thing.

>> Yeah.

>> Right? They're
too young to think
computer science is a boy thing.

>> Yeah. They probably don't even know what computer science is, right?

>> Right. They see like, "Hey we're going to make some stuff today,"

and they're so excited about it.
those elementary school classes,

they're half female,
the kids are all excited,

they're super into it.

We have a little tool at the end,

what we call our funnel meter.

They can give it a thumbs up,

thumbs down at the end of every activity,
and the girls actually
give it higher

funnel meter ratings
than the boys do.

The girls are into this
and they're into it young,

and so when we can get them

before they've got
those stereotypes,

they can make a huge difference
in terms of giving
them the momentum to keep going afterward.

I see the same thing you see with my own daughter.

But, she's also excited about computer science because she doesn't see it as a boy thing.

>> Yeah.
Even if you look back in history, computer science used to be a female thing.

>> Yes. It's just flipped, right?

>> It's about from the very beginning, the first programmer was a woman.

>> The first programmer was a woman,
Ada Lovelace about 100 years ago,

and then you look in the '50s,

in the '40s, computers were

women and computer science was a female,

the stereotype would have been women.

>> Yeah.
Then, it's men, and we can get back to a place where it's both.

We can get back to a place where we look at it and we say, "No, no, computer science, it's something that everybody does."

There's no reason it's one or the other."

But, it's not just teachers,
it's also parents, it's
social, it's friends.

Let's say there's
an after-school program,

you can just see this.

Mom says, "Oh, look,

some after-school classes.

Bobby, looks like there's
EP-02  Behind the Tech with Kevin Scott

a coding class after
school on Thursdays.

676
00:23:30,670 --> 00:23:31,860
Do you want me to sign you up?"

677
00:23:31,860 --> 00:23:33,280
Right? "Emily, it looks like

678
00:23:33,280 --> 00:23:34,840
there's a dance class
on Tuesdays,

679
00:23:34,840 --> 00:23:36,245
do you want me to sign you up?"

680
00:23:36,245 --> 00:23:38,460
It's so easy. They're
not thinking about it.

681
00:23:38,460 --> 00:23:40,790
They're just trying to find
activities for their kids.
So, when we do it after school,

what we see is that same skew

where boys are more likely to

get signed up after school

for computer science.

If we do it in school,

we don't see that.

So, that's why we want to
689
00:23:53,760 --> 00:23:56,010
>> Yeah, which I think is
awesome because

690
00:23:56,010 --> 00:23:58,270
sometimes when you're
focusing later,

691
00:23:58,270 --> 00:24:00,160
it's just really, really hard.

692
00:24:00,160 --> 00:24:02,135
I had this friend call me up.

693
00:24:02,135 --> 00:24:04,850
He was like, "I'm trying
to get my daughter to

694
00:24:04,850 --> 00:24:08,075
stay enrolled in her
AP Computer Science class."
She was a senior in high school then.

She just didn’t want to be in this class because she was the only girl in there.

>> That’s so hard.

>> And this isn’t Silicon Valley.

>> Yeah.
What wound up working was connecting her with a bunch of really successful women computer scientists, software engineers, who were having a really great time in their career. And she stayed in AP Computer Science class. She went off to university.
She majored in Computer Science,

dean's list student, is now in a professional,

so she's a software engineer at a tech company.

And that whole thing is hard to scale.

What you would want to do is do that for everyone.

But, it's so hard when you're starting
later, whereas starting earlier.

00:24:52,430 --> 00:24:54,240
you can maybe get to the point.

00:24:54,240 --> 00:24:57,090
where just naturally you're not having a class full of boys.

00:24:57,090 --> 00:25:02,900
in 12th grade in this AP Computer Science.

00:25:02,900 --> 00:25:04,150
>> Absolutely. We just hired

00:25:04,150 --> 00:25:07,080
a woman for our engineering team a
couple of months ago who's

studying computer science in college,

was one of the only woman in her class,

dropped out because she felt she didn't belong,

but liked computer science.

She liked it. She just didn't feel she

should be in it because
there weren't any other women in it,

and finished college still regretted it.

Still wanted to do computer science.

Ended up doing night classes and

side classes and learning it after work,

eventually did a boot camp, learned computer science,
moved into the career,

worked as a computer scientist,

and just recently joined our engineering team.

>> That's awesome.

>> But, you know that's the hard way.

>> Yeah. That's the hard way.
It would have been easier if she had just been able to stay in those classes in the first place.

>> Yeah.

Tell us a little bit about Hour of Code.

So, Hour of Code has just become a phenomenon.

It's exceeded our expectations.
If you're not in school right now, you may not have heard of it.

If you're in school, you probably have.

It's like Earth Day, but for computer science.
>> It's a national holiday.

I don't have the exact numbers or the number of which schools participate.

But, as far as I can tell, everybody I talked to, their school seems to be doing it.

>> I realized there was a bigger thing than I thought when Steph Curry
was posting on LinkedIn about him doing his Hour of Code.

>> Oh, yeah. Oh, hey, if you're into sports,

then Steph Curry did it.

If you're into other things,

Barack Obama's done it,

Justin Trudeau's done it, Dave Cameron,
that we've had about eight world leaders.

We've had musicians. We've had actors, actresses.

But, I think the most important thing is

the schools and the teachers are doing it.

>> So, tell folks what the Hour of Code actually is.
So, the idea is that I can tell you, until I'm blue in the face, that computer science is going to be fun, that you can do it.

There's nothing like actually trying it.

So, what we do is we get students and teachers to spend one hour trying computer science.
We've built scaffolded activities that make it easy for beginners.

In one hour, they can actually build something.

You could actually build a little, mini game, something you can share and be able to say, "Hey, I did that," and you actually learned some computer science.
I mean, you don't learn all of computer science, it's one hour, but you learn a concept or two. You might learn about if statements, you might learn about loops and how they work. So, the students get to try it, they get to try one hour. It's a great introduction.
We did a survey last year looking at thousands of students before and after they tried the Hour of Code, and what we found was that it does increase the amount that they say, "Hey, I like computer science or I'm interested in computer science."
But, was especially cool for me was that the group that was the most impacted by doing this was high school girls.

High school girls were probably coming into it thinking, "Hey, this is not something that I'm into."

They try it and then they're into it.
At this point, we've had 500 million hours of code around the world and it's been in 180 countries, it's in 50 languages.

It's a huge event every December.

We do it for CS Education Week, and basically it's just a way to introduce students around
That's incredible.

(by actually)

building something.

Yeah. It's really incredible.

Yeah. it's not just us,

done in partnership with
about 200 different companies and organizations that run it and do activities.

Microsoft has partnered with us on the Minecraft Hour of Code for the last few years which is our most popular Hour of Code activity,

and students and teachers love it.
It's an opportunity to use these characters they're familiar with from Minecraft, but to learn computer science with them. 

>> So, what's the dream for Code.org?

If you had a magic wand to wave over the world, and you can achieve
whatever success you wanted to achieve,

what does that look like?

>> I think it looks like every child has the opportunity to learn computer science and that the students who are learning it look like the world.
That the diversity matches,

so that when we look at
the workforce 20 years from now,

whether somebody is in education
or marketing or retail,

they're going to be
using computers.

It's going to be
a part of their lives

and everybody gets to understand
things like how the Internet

works and how computers work.

And that when we look
at the tech workforce,

that the students who are
prepared to join this,

that they look the population,

and I get to look around
and half my team is female.

I want to state that we're
working on one part
of the problem,

which is the K12 education.

That won't solve
the tech workforce by itself.

There are definitely
issues around hiring,

retention, workforce bias,

all of those other pieces
which also need to be solved.
But, I think if they we're working on one really important part of the problem.

>> Yeah.

>> We do need to bring more diversity into the tech workforce and I think education is critical.

>> Yeah, I think it really is.
The thing that keeps me up at night about our future is I just look at every year technology has a bigger and bigger impact on the world and the trajectory tells us that that's going to continue for the foreseeable future. And in a whole bunch of different ways
you want as many people and as representative a set of people as possible participating in the creation of this technology.

You want all perspectives, all backgrounds, all ethnicities, you want it to look like the world, which I think was beautiful way that you said it.
But, you also want society at large to be well informed because a lot of the funky stuff that's going on today we're going to have to make an increasingly large number of decisions, policy for instance,
in ethics and the laws that we pass and the regulations that are put into place to govern the intersection of society and technology.

You want people super well informed when we're making those decisions, and you want them represented--it's like everybody.

>> Absolutely. I mean,
it's just critical that in this world,

eybody has this opportunity.

>> Yeah.

>> At Code.org, what we do is we make it as easy as possible for schools to teach this.

We offer free curriculum,
Behind the Tech with Kevin Scott

we offer free professional development for these teachers,

we help teachers who don't have a computer science background.

>> Yeah.

>> Because the teachers don't.

I mean our schools don't teach it.
They didn't learn it when they went to school.

>> Yeah.

>> So, giving the teachers the opportunity to learn to teach computer science.

They're History teachers, English teachers, Math teachers.
>> Learning to teach
computer science,

as you pointed out earlier,

is different than even knowing computer science.

>> Right. It is different. That's funny.

We actually find that it's not the computer scientists make

the best teachers of computer science.
911
00:31:26,338 --> 00:31:28,863
It's teachers teach computer science

912
00:31:28,863 --> 00:31:30,308
the best because they're good teachers.

913
00:31:30,308 --> 00:31:34,658
What we've found is that experienced teachers with

914
00:31:34,658 --> 00:31:36,788
no background in computer science make

915
00:31:36,788 --> 00:31:39,378
excellent computer science teachers

916
00:31:39,378 --> 00:31:39,378
because they know how to teach.
Yeah.

If we give them the tools and the resources and the curriculum,

they're fantastic in the classroom,

and their students do really well.

So, that's what we're working on doing.
I mean, these schools teach computer science.

>> What are some ahas that you've seen over the past several years trying to teach computer science kids?

>> Oh, there are so many.

I'll give you a personal one to start out with.

So, I came into
this thinking I was
a good computer science teacher,

and it turned out surprise,

surprise, I was not.

I love teaching. I think
a lot of people like me,

they enjoy it. It's fun.

I taught in college,
I started a program
to bring students

into local schools to
teach computer science.

I was TA, I was

a teacher, and I always
got good reviews.

I always got
high scores on the

which TAs are the best,

which teachers are the best.
So, I had this misimpression that I was good at teaching.

It's been fascinating getting to work with a bunch of pedagogy experts on how do you actually teach because what it turned out was that I was entertaining in front of a room, which is different from...
behind a good teacher.

00:32:43,738 --> 00:32:44,088
>> Yeah.

00:32:44,088 --> 00:32:45,583
>> So, when we teach networking,

00:32:45,583 --> 00:32:48,393
we have a thing called ABC CBV,

00:32:48,393 --> 00:32:51,338
which is you do the activity
before the concept.

00:32:51,338 --> 00:32:51,548
>> Yeah.

00:32:51,548 --> 00:32:54,023
>> You do the concept
before the vocabulary.
It's not about a teacher standing in front of a room lecturing.

It's about letting kids discover it on their own.

The art of teaching is stepping back. It's doing less.

It's not being entertaining.

It's not being this person who's like super energetic,

exciting person to watch.
It's about crafting experiences where the student is going to get to figure it out without you being involved. Because if they figure it out themselves, they're going to remember it.
We pair them up and we say, "Hey,

you guys got to figure out how to send some messages back and forth."

We have this little software that lets them send these little packets of messages back and forth.

But, our software is going to
EP-02  Behind the Tech with Kevin Scott

drop some of those packets

on the ground.

975
00:33:33,128 --+ 00:33:34,393
We're just going to lose them.

976
00:33:34,393 --+ 00:33:36,188
We're also going to send
some of them out of

977
00:33:36,188 --+ 00:33:38,243
order because that's
how the Internet works,

978
00:33:38,243 --+ 00:33:39,438
and they've got to figure out,

979
00:33:39,438 --+ 00:33:40,838
"Okay, I'm sending you messages,

980
00:33:40,838 --+ 00:33:41,948
some of them come on out of
order and some of
them get dropped.

How am I going to
deal with this?"

I don't care how
they deal with it.

Some of them will send

five copies of the packet

because there is
going to be like,
"Okay let's just keep sending them because they're going to keep dropping them."

Some of them will number them, some of them will send back [inaudible] to say,

"Yeah, I received or didn't receive your packet."

It doesn't matter what method they come up with.
The important part was that they really understood the problem because they tried to solve it.

Then, after they've done that we say, "Okay, that thing that you just did, that's called a protocol."

>> Yeah.

>> The protocol the Internet uses is called TCPIP.
Now, what did the teacher do in that whole lesson?

They facilitated the communication with the students.

They got the students paired up,

they helped a student who was blocked get to that next step.

But nowhere in that lesson that the teacher stand up in front of the room and
draw a picture of TCPIP.

>> Yeah. I've had similar sorts of problems with my kids and it was the same thing for me at my goal in life was to be a computer science professor from age 16 to 31 when I left academia.

I taught undergrads for years,
I taught grad students,

and now I'm trying
to teach a couple of

really young children about

these computer science concepts.

And so I'm sitting down at

a restaurant and teaching
them about binary search,
I think they got it right away because I made it into a guessing game.

I'm going to teach you a trick for how you can get someone to play this guessing game with you where you can find the number that they guess between zero and 128 in seven steps or less.
You know they're like, "This is great."

But, then I wanted to teach them how to do search, and there are like these little things about teaching search that sort of hard.

One of the things is, if you just take
a bunch of numbers and write them down and say,

"How would you sort these?"

One of the things that's interesting is human beings can see all of the numbers at one time.

So, they're cheating in a sense when they're imagining how they're sorting.
And so I devised this thing
where I could give them
a bunch of blocks
where the numbers on the blocks
were covered up and,
so they could go examine
the number on the block
one at a time,
which is how the computer
goes and does things.
I just really realize that I was all kinds of wrong about how good I was going to be at teaching little children these computing concepts. Actually, the way you ended up doing it is very similar to how we do it in our class. So, what we do is we give
the kids decks of cards.

They're only allowed to lift two at a time to compare them because that's how a computer would do it.

>> Yeah.

>> They can't look at the cards when they flip on.

They show him to the other student and the student says
which one's bigger.

00:36:13,913 --> 00:36:14,218
>> Yeah.

00:36:14,218 --> 00:36:15,468
>> So, they get to pick two at

00:36:15,468 --> 00:36:17,528
a time and see,
and then actually,

00:36:17,528 --> 00:36:19,573
one of the things
that's cool about that

00:36:19,573 --> 00:36:21,858
and a lot of our lessons is
they're not on a computer.

00:36:21,858 --> 00:36:24,018
They're actually using
physical cards in the classroom.
>> Yeah, which I think
it's actually great.

>> It's great. Yeah. Because you
know when you say
computer science,

I think, sometimes
people think, "Oh,

it's all on a computer," and

really about half of our lessons
are off the computer,
and it's about interacting with other students.

It's about internalizing the concepts by working with the actual concepts and the logic outside of the context of the computer.

>> Thank you so much for doing this work.

I couldn't be a bigger fan and I think you guys
are having an enormous and amazing impact on the world.

Thank you for taking time to be on the show today.

>> Oh, no, thank you,

and thank you for Microsoft's support.

>> Well, thanks for joining us on Behind the Tech.

I'm back with my colleague, Christina Warren.
Some of Alice's insights were pretty awesome. What stood out for you?

>> So, one of the interesting things I thought about your conversation with Alice, and we talked about this a little bit before,

was hearing her story and hearing about
the atypical journey and how she got involved with technology.

>> Yeah, I think there's an incredibly diverse set of folks in tech,

just sort of based on the path that they took to get into the industry.
I've had the great pleasure of being a computer science teacher and being an engineer and engineering leader for a really long time now, and have just come into contact with tons and tons and tons of engineers.

Each one of their stories
is a little bit different and some are sort of stereotypical image.

But there are all sorts of other folks like Alice, who discovered computer science in their senior year of high school.

There are some folks who discover it in college.
who actually go off and have a career in some completely different thing and decide that they want to get into computing later in their life or later in their career.

The thing that I'm seeing now is that, it's increasingly easier to make those transitions because
the tools and capabilities and

sort of richness of

our programming environments and the way that we build

software just sort of allows

more and more people to get bootstrapped more and more quickly.

Part of that's a byproduct of the open source wave of
software that we've been witnessing over the past three decades.

>> Yeah, definitely. One of the things I love about code.org is that, even if the kids who are going through this programs, even if, say,
they don’t choose to study computer science in college,

they still have that foundation.

>> I think it’s a really important thing

that everyone in society understands a little bit about computing because computing and technology is having a bigger and bigger impact

on all of our lives
all of the time.

So, being informed about

some of that stuff

and having an idea

in your head about how things

work is going to help

you be a better citizen.

>> I feel like that's

the only way that

our products get

better is by having
more diverse viewpoints

and different types of people coming into doing things,

because you never know what someone's perspective is going to bring.

I love what code.org is doing in bringing more and more people
EP-02  Behind the Tech with Kevin Scott

into the fold and
letting them know,

1143
00:39:33,268 --> 00:39:35,643
"Hey, you can do
this and it's fun."

1144
00:39:35,643 --> 00:39:37,443
>> Yeah, tons of fun actually.

1145
00:39:37,443 --> 00:39:40,818
But I have a biased
opinion there.

1146
00:39:40,818 --> 00:39:43,378
I think that whole
pedagogical framework for

1147
00:39:43,378 --> 00:39:45,898
teaching computer science
to kids is really great.

1148
00:39:45,898 --> 00:39:48,078
I think it's actually going to prove to be great not just for kids but for adults.

When I was a lecturer at the University of Göttingen in Germany, I was teaching a class on programming languages and the theory of computation,

and some of that is
That certainly challenged my ability as a teacher especially because I was lecturing in English to a class full of non-native speaker. >> Yeah, I was going to say, so you're doing this in Germany, teaching English and then
there are non-native speakers,

although I guarantee that they understand English far better than I

understand German, but still.

>> That was always embarrassingly true for me.

Their English was way better than my German.

In some ways, it's a different challenge to
really bring someone up from the ground to how

do you get over this beginning set of conceptual hurdles

so that you can then get

into the computer science curriculum?

By the time I got them,

they knew sorting algorithms,
they knew if-then-else statements and while loops

and all of the basic things of how you construct a program.

I think at least until I had kids of my own,

I took for granted how difficult it is to teach the "quote unquote" simpler stuff.
I think the lesson for me is appreciate my teachers even more than I already did.

We should all appreciate those teachers who are out there loading knowledge into the heads of our future fellow citizens.

>> Absolutely.

>> Well, thank you so much, Christina.
This has been a great conversation, and I look forward to being back with you again in the next episode.

>> Me, too. Thanks so much.

>> Next time on Behind the Tech, we'll talk with Andrew Ng,
the co-founder of the Google Brain project, Coursera,

and most recently, deeplearning.ai and Landing.ai.

Andrew is one of the most influential leaders in AI and Deep Learning.

Be sure to tell your friends about our new podcast, Behind the Tech, and to subscribe. See you next time.