Resolve To Get Tech Savvy

THOMAS WILSON: All right. Good morning, everyone, and thank you for joining us today for today's workshop, Resolve to Get Tech-Savvy, a collaboration between Washington University's Global Connections and Tech Diversified.

My name's Thomas Wilson, and I am one of the co-founders for the group Tech Diversified, along with our presenter today, Ryan Whitley. We're very happy to join you all this morning here in present and those that are online, as well.

Like I said, this is a collaboration between Washington University and Tech Diversified. You know Washington University, but you probably don't know anything about Tech Diversified. Tech Diversified is a nonprofit organization that my friend Ryan and I started to expose more people to tech literacy-- everything from coding, programming, career advising, mentorship, everything around tech that you can think of. And what we want to do is expose more people to tech, including people of color, women, and bring more people into the space.

And so today, we really wanted to touch on some of the basics of why it's important to increase your tech-savviness. As we all know, tech is really starting to pervade a lot of things in our world and a lot of the careers that are out there, as well. Those that have a little tech-savviness are going to see that they are going to be very successful and a very important part of the business landscape that is out there.

So today, I want to introduce my good friend Ryan and partner. He is a software engineer in the daytime. His day job, I guess, is with a company called Spatial Development. They take a lot of big data from companies such as Bill and Melinda Gates Foundation, NASA, World Bank, Red Cross, municipalities, different companies, businesses, private entities from around the world. They take a look of that big data and make it so that it's understandable to their clients and the people that they work with.

And so Ryan is a developer, so he works on websites, works on that data and heads up a lot of different teams that work on these types of things. And Ryan, my good friend, is really awesome in that he really wants to share everything that he knows with people that want to work hard and see if they can be a part of this new tech boom in space that we're all starting to see.

And so Ryan, I'd like to-- well, I'd like to introduce all of you to my friend, software engineer Ryan Whitley.

Thank you, Thomas. Welcome. Oh, thank you. Welcome, everybody. Thanks for coming. Thanks to you that are joining online. And I'm just going to get right into it. So I've got a presentation today. I'm planning on showing three examples, practical-- I think they're practical examples--

of how people who aren't necessarily planning on pursuing software development as their career, how you guys can get started and get exposed to three different technologies.

So I'm going to show those three things in a few minutes. But I have a few slides that I want to introduce first. So this is the WSU Global Connections talk, Resolve to Get Tech-Savvy.

So tech-savvy-- why should you care about being tech-savvy? I think there's two main reasons. Thomas touched on one of them. I think the first is, even if you're not planning on being a coder as your profession, I think it's important to know about technology because it can give you a competitive advantage-- if you're going to find a job, throughout your career.

Let's say you're applying for a position. And one person knows how to update a website, let's say, and the other one doesn't. I think the person who knows how to update the website's going to get the job, all other things being equal. So just from the standpoint of being more marketable in your careers, I think that's a major point.

The other one is that technology, as Thomas said, is everywhere. This webinar that's running. Somebody wrote code to make this all possible. Smartphones-- I'm sure everybody here probably has a cellphone or a smartphone. All the apps, all the cellular networks, everything that makes smartphones run and operate, it's all code. Somebody sat down and wrote code. Autopilot-- planes can fly themselves based on code that people wrote. Even the space shuttle-- space shuttle auto pilot can land by itself. All people wrote code to do that.

Things like traffic lights-- you might not think that that's controlled by code. But in some areas-- like Bellevue, I know-- traffic lights are controlled by code, timing and things like that. Different times of day they might change depending on traffic conditions. Artificial hearts-- all of the little electronics in there. Elevators, the internet-- it's obviously huge. Movies-- all the CGI that goes into animated movies like Toy Story. It's all computer graphics, computer coding. It's all technology.

Games, obviously. Self-driving cars are becoming more popular. We might see those hit the road in the next few years. ATMs. So basically anything you can think of is controlled nowadays through code or through some sort of technology.

So basically, it's pervasive. It's everywhere. And if you're not paying attention to it, unless you live off the grid like this guy, then you sort of need to be paying attention. I think it's definitely important to lean in and sort of pay attention to what's happening. Again, I'm not expecting you to be Bill Gates. You don't have to sit down and write code all day. But just paying attention-- pay a little bit of attention, figure out what's going on with code, just so you can participate in the larger conversation that's happening.

So this slide-- before I started coding, whenever I saw some code-- sometimes you'll see some on TV or in a movie-- it looks kind of like this. It looks like The Matrix. It looks like a bunch of incomprehensible letters and numbers. It's scary. It's overwhelming. So I'm here, hopefully,

today to show you that you shouldn't be scared. Don't be overwhelmed. If you take things slowly, you can figure out little bits and pieces and sort of learn incrementally.

So I think the best way to get started is to demystify this whole universe of coding and technology. I'm going to sort of talk more about coding because I'm a coder. So there's all different types of technology. But I'm going to sort of talk about coding today.

So I think the best way to start is, again, to demystify it. And so I would suggest maybe picking a project-- a small project, something that might be practical for you to do or fun, something that would blend, maybe, your own personal interest. So if somebody's a photographer, if you have a friend that's a photographer, maybe they want a website built. So maybe that could be a little pet project is building a website for your friend or for a club or church or something like that.

If you like gaming, maybe you could try making a game. That's another way to get into it. And these are the three samples that I'm going to show today-- simple HTML website. I'm going to demo a game engine, a free game engine that everybody can download and get started right away. And then the last one would be something like automating your job. And we'll get to that. Take longer lunches because you'll write code that'll do your job for you.

And I think the most important thing, and the bottom line for me if I'm talking to somebody who's trying to get into the field, is I think it's really important to just tinker. Again, you might start one of these things. You might start building a website. Maybe won't finish. But just through the process of tinkering, I think you'll learn quite a bit. And that's enough to be conversant and to be able to join in a conversation. If somebody starts talking about one of these things, you'll be able to participate.

So the first example I'm going to show is making a really, really basic web page. I put "in two minutes." We'll see. In order to do this, you'll need, obviously, a computer, text editor. You don't necessarily have to do this right now, but if you watch this broadcast presentation later, you can follow along with that. But basically, you just need a simple text editor-- so Sublime on a Mac. Or if you're on Windows, Notepad or TextPad, something basic like that. You could build an entire website just with Notepad.

So I'm actually going to walk through this. There's a few steps here. First is to open your text editor. Secondly, we're going to type in some HTML. And I have that here on the screen on the right-hand side. This is just about as basic as you can get with an HTML page. And I'm not trying to teach you guys HTML, obviously. I just wanted to show you that you can make a website quickly with the tools that you probably have on your computer already.

So I'm going to exit out of this presentation real quick. I'm going to hop over to Sublime. And I downloaded Sublime. It's free. I believe it's for Mac and PC. So you should be able to get it either way.

So I have a blank screen here. Now, I could do just about anything. You could build YouTube. You could build your own Facebook if you wanted to take the time to do that. But we're going to start simple. So I'm going to type just a few lines of code here. And we'll hopefully have a web page when we're done.

In HTML, which stands for HyperText Markup Language, everything has a beginning tag and an end tag. So I'm going to start the beginning . That's how all web pages start, . And I'll go ahead and close it. So everything's inside of this HTML container.

And inside of that, you'll typically see a body tag. That's where everything that you see on the web page lives, inside this body container. So I have an opening body tag. And I'll do a closing body tag. And then I'm just going to type some text inside of there. "This is the best website!"

And so all I'm going to do is save this into a folder. I'll call it bestsite.html. So if you save the file with a .html extension, then when we go to open it later, it will automatically open with a browser like Chrome or Firefox or Internet Explorer. And then that's it. That's a website.

So let's go ahead and save that. And then I'm going to go to the folder where I just saved it. And I'm going to double-click on it. What was it called? Bestsite? Here it is, bestsite. We're going to double click this, and there it is. So this is Chrome, where I'm viewing all the websites.

SPEAKER 1: Can you make it bigger?

RYAN WHITLEY: Sure. Sure, I can zoom in while I'm doing this for those of you having trouble seeing. So again, I didn't claim that this was going to be anything great. This is a website. Let's spice it up a little bit. Let's add an image.

So I'm going to go back to my text editor. And let's add an image. And I believe I have that Eddie Murphy picture. It's called eddie.jpg. Or I have another one, an animated GIF, eddie.gif.

So in my text editor, I'm going to point it at that file. I'm going to reference that file. And in HTML, you add an image by typing an image tag-- . And you say source is, and then you just give it the file name. And I want eddie.gif. So I'm going to put eddie.gif. I close the tag. I save it. Then I'm going to reopen the site. Hopefully we'll have a picture of Eddie Murphy. And there you go.

So with just a few small lines of code, I added an image, which is an animated GIF. I added some text. We can do things like change the background color. Let's make the background green real quick.

So I want the whole page to be green. So everything we say on the page is in the body tag. So I'm going to say style="background-color: green" equals And let me zoom in there so you guys can see it.

And background-color: green, with two E's, not three. And then I want the text color to be white because if it's black, it'd probably be hard to see. And so the way you change the text color is just color: white. And then I'll zoom out. And we'll save it.

And I'll open it again. OK. Now we've got a green page with white text, and the image is still there. So obviously, I just wanted to demonstrate that you could open Notepad, you could open a text editor on your laptop or your computer at home. You could start making a website.

Or better yet, let's say you get a job and nobody in the office knows how to update some website that somebody made five years ago. Nobody in the office knows how to do it. You come in. Well, now you know how to add an image or change an image. You go here, you change the image or you update the text. And suddenly you're the webmaster because nobody else knows how to do at the office.

So that's my first example. I'm gonna hop back to the presentation here. So some next steps-- so if you're interested in this sort of thing, maybe HTML, I think this is a very marketable skill. Even if you just know a little bit, you'll probably be ahead of the general population who's sort of intimidated by this.

So I would recommend a few resources. One is CodeAcademy.com, free online courses. CodeSchool.com, also free online courses. So you can go to these and, at your own pace, take online tutorials that will teach you, basically from scratch, how to get in there and modify some of these things. And this presentation will be available later or you could watch it again to get these links.

There's also in person bootcamp. So something like CodeFellows is in Seattle. That is a paid, intensive bootcamp. And I believe they last-- they have different offerings, but you might do a three-month program where you're doing 40 hours a week of just coding. And they really just teach you how to code. So that's another option, if you're sort of more interested in taking it further.

Again, as I said before, I think it's important to just tinker. So go slow. If somebody asks you to modify a page, don't go learning all of HTML. Figure out how to do one thing. Figure out how to change the background color. Figure out how to change the text color, the size. Just pick one thing, learn how to do it. Once you've mastered that, pick something else.

Google is a great resource. Even experienced coders use Google all the time to answer their questions. There's gonna be a lot of trial and error, but I think that's a great way to learn. So don't be discouraged. Stick with it. Again, just get in there, get an understanding of what's going on. You don't have to be an expert, but just get a good understanding.

So that's the first example. Are there any questions online or anybody here about anything I just said? No?

SPEAKER 2: Can you put the slide up for the steps?

RYAN WHITLEY: Sure.

SPEAKER 2: Like the open up your text editor, type in this text.

RYAN WHITLEY: Sure.

SPEAKER 2: I didn't get it all.

RYAN WHITLEY: Let me skip forward to that. So I'm going to just briefly show the slide again where it has the steps of what I did. So open your text editor, like Sublime or Notepad or TextPad. You're going to type in some basic HTML. So I started out by typing just an opening and closing html tag, an opening and closing body tag. And then in there, I started typing some text.

And then you save it as a .html file, the type. And then you should be able to then double-click on the file, like in Finder or Windows Explorer. And it should open in Chrome or whatever your default browser is. And that's it.

SPEAKER 3: How do you make changes to an already existing website? I understand how you do it. But let's say you're not the one that created the website.

RYAN WHITLEY: Sure. That happens all the time. So the question is, how do you make changes to an existing website if you didn't create it? So let's say you're in an office, and the manager ask you, please update this website. It says something about coming up in 2005. We need to change it to 2015.

So the first thing you would need to know is where those files are. Whoever made the website made a bunch of files. And they have them somewhere. So you have to find out where the files are.

Secondly, you would open-- you would find the file. Maybe it was a calendar page. You would find that, open it-- you could open it in Sublime or Notepad, like I just showed. And then you would just simply find the date and change it and save it, put the file back where it was, and that's it. So basically it's just identifying where the files are, modify them, and put the file back where you found it.

Yeah, we have another question.

SPEAKER 4: So when file, though, you said to open the page, you double-click that HTML. But if you just want to open it to read it and make changes, how do you do that?

RYAN WHITLEY: Sure. So the question is how do you open the file to edit it versus open the file to preview it? So I'll go back to the folder where I have the file. And so my file is called bestsite.html. And instead of double-clicking it to open it, I'm going to right click or bring up the options here. And I'm going to open it with Sublime.

So when I say Open With, you would choose Notepad or TextPad or Sublime. And then it just opens the file so you can edit the raw HTML.

So that is the first example. That's the HTML example. I'm going to move on to the second-- I think it's a practical-- example, which is making a game. And so for this, I'm going to use something called Unity3D. It's a free gaming engine. It works on Mac and Windows. This is a professional-grade gaming engine. So once I show you, you might see that name if you download an app or a game on your smartphone. I mean, this is something that game studios use to make real games that people make money from.

So there's a free version and a paid version. But just about anything you want, you can achieve with the free version. If you need the paid version, it would be after you've completed this huge game.

So there's a few steps here. I'm going to actually walk through these and show you actually how to get started. But basically, you would download Unity3D. You open it. We're going to make a new project, and then we're going to drag a few items out to the canvas. And I'll just show you how to do that.

So I'm going to exit the presentation. I'm going to-- I've already downloaded Unity3D. I actually have it open already. So I'm just going to bring up the Unity3D interface. And again, this is free. If you have a Mac or a PC, you should be able to download this and install it.

And so without doing any coding or digging into the guts of this thing, we're going to be able to have a world that we make. And we'll be able to control a person and walk around the world and go up hills and down hills in just a few minutes.

So when you start, you get this 3D view. And right now it's empty. And we're going to do three things. The first thing we're going to do is add terrain, which is like a ground to walk on. And to do that, there's a couple ways to do it. I'm to go to Game Object, Create Other, and then go down to Terrain.

So you should be able to see that it just added a big-- basically a big plane. And I can zoom out a little bit. So you can see here, I've got a big terrain. It's flat. It's boring. Let's add some mountains real quick. And to do that, I will click on the terrain. On the right-hand side, there's a few tools here.

There's a little icon of a mountain going up. So I'm going to add a hill. I'm going to click on it. Then out on the left-hand side, I'm just going to click. And you'll see some little mounds

forming. It's kind of like Sim City. If any of you have played Sim City, you can control the terrain. So a mountain, great. You can add trees. You can add water.

I'll just leave a mountain. And I want to paint it because I don't want it to be gray. So I choose the paintbrush. So I'll pause right there and say this seems a little daunting at first. And actually, I've only been tinkering with this for about six months. But it was helpful-- I watched a few 10-minute videos online. And I was able to sort of understand what's going on here in 10 or 20 minutes. So I'll have some links after this that show you where you can go see those videos if you're interested in this sort of thing.

So what I'm going to do is I'm going to paint this with a texture, which is just an image. We're just going to drop an image on top of it so it shows up as green or brown. So I'm going to select the texture. Let's do grass and rock. Add. It's hard to see here, for those of you live. But so now it's painted with sort of a brownish rock landscape.

And let me zoom in a little bit here so you guys can see. Sorry if it's a little small. So I've got my mountains. They're painted. So I'm going to do two more things. Now we've got somewhere to walk. The next thing we need to do is add a light, basically like a sun, so that we can see what's going on in our area here.

So I'm going to go to Create, Directional Light. And it just illuminated-- it just added a light to our scene. So now we can see all the objects in our scene. We can see what's going on.

And then lastly, I'm going to add something that comes out of the box with Unity3D. It's called a character controller. So over here in the Project column, you should see Assets, Standard Assets, and then Character Controllers. And I'm going to simply drag and drop this thing called First-Person Controller. I'm going to drag it out into my scene on the left. And I'll drop it out there.

Let me zoom out a little bit so I can see what's happening. I'm going to delete the main camera because there's a camera attached to our little person that I just dragged out there. I'm going to make sure the person is not halfway into the ground like you see now because if I start this, they would fall right through the ground. So I'm going to use a positioning tool and just drag this person up a little bit.

And the last thing I'm going to do is hit Play. And I didn't get him high enough up. So let me continue to drag it up. So this is just our person. So now you can see, I hit Play at the top, which starts the game as it is right now. And you can see I'm using my mouse to look around. I can look up at the mountain. And I can use my keys, WASD, to walk around. And I can use the space bar to jump. And it has physics built in.

So this is a really simple example of starting to play with this gaming engine. Now you can add cannons. You can add trees, water, wind, sound effects, all kinds of things. And so for those of you interested in gaming, this is a quick way to get started in that world, if it's something you're

interested in. And again Unity3D-- now that you've heard that, you might hear it come up on social media, or on the news, or you might see it when you start a game that you've downloaded on your phone.

So this is something-- again, if you're interested in this world, this is kind of a good way to get started in it quickly. You can do all kinds of crazy things. And you can write code for all your objects in your world to interact with each other. So you can go crazy with this. But again, take it incrementally, step by step. And just learn one thing at a time.

I'm going to jump back into the presentation. Any questions about any of that? OK. So some next steps with Unity. So OK, we didn't build Call of Duty or Angry Birds. We have some mountains with a character that walks around.

But some extra things you could do, for example, is you can load real terrain data. So I went to the NASA website, clicked on a Mars link, and you can download Mars terrain data. And I downloaded a big crater. And I was able to load it into Unity so that I was able to walk around a crater and just have fun.

I also downloaded West Seattle and Seattle and Ballard and made a little game that takes place in Seattle. So it's kind of fun to just tinker. Again, I haven't published any games. But just kind of learning what it takes to make some of these games, I think, is good. It's good to just understand what it takes. And so you'll be able to participate in a conversation.

So unity3d.com. They have some great tutorials, videos. You can just watch them, five- or 10-minute long videos. They'll show you how to get started with different aspects of Unity. There's unity3dstudent.com, which has great tutorials. And also, YouTube has tons of tutorials. So just go search for Unity3D.

I have the same paragraph on the bottom. Go slow, figure out one thing at a time. Use Google to help you. You're not trying to make a full-blown game right away. Just do one thing at a time. And you might learn a few things.

So I have a last example here, which is automating a task using something called Python. Python is a scripting language. It's very powerful, and it can do tons of things. But something that we use it for here, or that I use in my day job, is something that might be tedious or monotonous that you find yourself doing over and over and over. For example-- and this is a fictitious scenario, but I know plenty of people who do things like this. I've done things like this.

So let's say there's an automated system at your work. And every day, it dumps image files or text files into a folder. And your manager asks you to spend two hours a day renaming those files to remove the underscores and replace them with dashes. I mean, this sort of thing happens all the time. This is-- some people just do this. Or you add a timestamp.

They want you to manipulate files over and over. This is a perfect job for a Python script to do. And so a Python script, you write some code. And when you execute it, it will follow the instructions you gave it. In this case, we're going to give it the instructions to look-- I'm not going to explain what all this means.

But basically, it's going to look for all the files in a particular folder. And it's going to replace all of the underscores with dashes. And so if I put this script in a folder full of files and execute it, they'll almost instantaneously all be updated. So I'll show you that real quick.

So in order to do this, you can still use Sublime or a text editor to write this code. But you have to have something-- you have to install Python. And so you could Google Python and install it. It's free.

So this is a little bit more of a set-up. But once you install Python, you'll be able to write Python scripts and help automate some processes. So I'm going to show you-- so there's the code. I copied that code into Sublime text editor. I'm going to open a folder. I called it work files.

So in this folder, I just created a file called my_oh_my. And I put underscores in it. And then I copied it a bunch of times. So now there's a bunch of files called my_oh_my. They all have underscores.

Then I have my Python file here, which I wrote in Sublime. Let's see if I can open it with Sublime just so you can see. It's what we had on the slide. Again, I won't go into explaining what all of this means. But again, it's replacing underscores with dashes.

So in order to run this, I'm going to open up what's called the terminal. This exists on Windows and Mac. It might look a little scary. But what I'm going to do is I'm going to tell Python to start up. And I want Python to run my file, which is called example.py.

So I'm going to type example.py-- so python example.py. And soon as I hit Enter, hopefully, if we're lucky, you'll see all of these files magically switch. And they'll all have hyphens instead of underscores. So here we go. You got to watch quick because it goes fast. One, two, three.

All right. So I hit Enter, the script ran, it changed all the files. So that's it. So instead of taking-- what if there were thousands of files? Or some other people's job might be to copy these files to some other location. You can do all of those things with Python. So you could really automate a lot of this stuff. And you can chain things together. Maybe you rename and then copy.

So you can build some really complicated scripts to automate your work. And I think, again, this is definitely a competitive advantage. If you were to learn just a little bit of Python, if you were to go to probably most offices, they would have some sort of mundane task like this. And if you went to the manager and said, yeah, I could automate all that for you. So instead of paying

somebody for eight hours to do it, I'm your new IT person. And I'll have it done so it's running automatically every day.

So this is something that's very powerful. It's a little bit more hardcore because it is really just sort of hardcore coding. But again, there's plenty of tutorials online, ways to get started. Start with something small. You saw that was just one line of code that does a big job. We have a question?

SPEAKER 5: Yeah quick question-- so what would be the best way to learn Python?

RYAN WHITLEY: I'm glad you asked. Let me flip back over to-- I have a few links here for some--

SPEAKER 5: Should we learn things like [INAUDIBLE]? Or can you learn Python script without [INAUDIBLE]?

So the question is do you have to learn some other languages first before you learn Python? I think the answer is no. I think that HTML and Python are sort of two separate worlds and languages. It's possible to merge them. But if you're doing something like some small automation tasks, I think you could just learn those pieces for what you're trying to accomplish, whether it's moving files, renaming files.

So I don't think you have to learn any other language first if you're just planning on doing something small with Python. So I have a few links here. Again, Code Academy has some Python modules so you can learn. The second, leanpython.org. I believe they even have, right in their website, a little code editor. So you can try things out right there online, probably without having to install Python on your machine. So that's a good way to learn. And also YouTube. YouTube has a nice intro to Python.

So this is something you could use. Again, I think it would help differentiate you from another candidate if you were going after a job. It could help you get a promotion in your current job if you suddenly are automating all these tasks and making things more efficient.

So one more time, go slow. Do one thing at a time. Don't try to learn all of Python. Learn enough to do the job that you're trying to do, and you'll incrementally build up knowledge. So trial and error is good. Google is good. Googling answers are good.

And so those are my three examples, practical examples, of how you can get started in three different areas. I just had a couple more slides on lingo. So I thought of a few things that you sort of might hear in the media or you hear people talking about. And so I'm just going to sort of try to explain what those mean so next time you hear one of those, you might be able to participate in that conversation or at least know what people are talking about.

So I just thought of a few things. If anybody out there has any more questions or anything to add to this list, I can try to answer that after the presentation. Or if any of you have anything to

add here that you want to know what it means, please ask. So the first is GitHub. The second is API. Third, open source. Fourth is the cloud. Fifth is JavaScript. So here I'll just briefly give a little explanation of what these are.

GitHub-- GitHub is basically a website for coders. And the primary benefit is that it's backing up your code to the cloud, which I'll explain in a second. But if you drop your laptop in the bathtub for whatever reason, then you would have your code backed up to this GitHub website. So you would be able to hop on another computer and download it and keep working. So it's good. It's great to have backups.

The secondary benefit is that you can collaborate with other people. So you can go on GitHub. And let's say you wanted to-- that Python script I showed you. You could go on there and search. I'm looking for a Python script to rename files. You would find somebody else who had already done that. And you could basically copy their code. And then you could add to it. You could keep it for yourself. You could do whatever you'd like with it.

So it's a great way for people around the world to contribute, help each other out, share code, and back up your code. So that's GitHub.

Next is API. It stands for Application Programming Interface. You don't have to memorize that because nobody says that. Everybody says API. But what it is, it's basically a website that computers-- you wouldn't visit an API. But another program or another company's program would connect to that website and just bring down data. So it's a way for an organization-- let's say Facebook. Facebook has an API.

It's a way for a programmer to connect to Facebook data. And you make new apps using that data. So you could make your own Facebook application. You would allow it to share your contacts with your app that you're building. And so you could have lists of all your friends. And you could put them on a map or something based on where they live.

So it's a way to reach into Facebook and pull out data from Facebook. That's an API.

Open source-- so if you hear something about open source software, it just means that the company or the person that's writing that software, it's available for all to see. Basically, they're sharing that code with the world.

And that's good. It's typically good because it means, kind of like Wikipedia, everybody can chime in and make it better. And it just keeps getting better over time because everybody has the power to change it and make it better. So open source is a strong movement that's going on right now. You'll see a lot of open source projects on GitHub, number one.

And so anyway, it's a great way to sort of get started with a project. And then you build on top of it. And then somebody else will build on top of that. Pretty soon if this really great, powerful software. So that's open source.

The cloud-- I mentioned it earlier. The cloud is basically a bunch of computers that are always on. Usually, there's like a warehouse in the desert somewhere full of computers. And that's the cloud. It just means you can save your photos to those computers. And they're always on. And there's redundancies. And they have other warehouses that are backups of that warehouse.

So basically, it's a place where you can store your photos or run your applications from that's always on. That's basically the cloud.

And JavaScript-- JavaScript is a powerful coding language. And it's typically used to make web pages interactive. So if you remember 15 years ago or 20 years ago, web pages were pretty static. Nothing moved. They just sat there. With JavaScript, you can make things start to be interactive. So you have interactive menus that fly out or things flashing.

So it's a way to add some pizazz to a website and make it interactive. Any questions about any of those things or any other terms that somebody has heard that you want to know more about?

SPEAKER 5: I don't know if this is part of the question. But does it differ with mobile versus regular PC, a physical system?

RYAN WHITLEY: Yeah. So the question is, is there a difference between the mobile world and the PC world. When it comes to-- do you have anything like the cloud or--

SPEAKER 5: Probably more of the coding.

RYAN WHITLEY: Yeah. So in general, somebody who writes code for a mobile device is sort of in that world. And there's a whole specific language for that and a way of doing things for mobile devices that you wouldn't do for a website, let's say.

So yes. That's one thing to take into account if you want to learn a programming language, is whether or not you want to build mobile apps or games or something like that, or if you want to build websites or desktop apps. And there are some transferable skills that go between both. But you sort of would probably start by focusing on one over the other.

And so I believe, just to summarize-- so yeah, technology is everywhere. I demonstrated it's all around-- stop lights, phones, the internet. It's everywhere. And I think in order to participate in the conversation and to sort of keep up to speed with what's going on and not be left behind, it's probably a good idea to sort of at least scratch the surface a little bit, find out what's going on with something-- with mobile phones, with apps, with coding, with gaming, with scripting.

Pick something and just scratch the surface, take a peek in there, see what's happening. And I think it really will make you more marketable. I think it'll give you a competitive advantage. And I just think it's a good idea. Otherwise, you'll run the risk of being the guy who talks into the mouse thinking it's a microphone. And you don't want to do that.

So that's all I have. Thank you. Are there any questions? And then I have a few links here. One is to Tech Diversified, which Thomas mentioned at the beginning. The other is SpatialDev, which is my day job. Check those out. And you can email us if you have any questions.

Are there any questions in person or online? Yes.

SPEAKER 1: In regards to building and coding a website, I've used Squarespace and WordPress and things like that to update all of those things and to maintain a website. And of course, it always comes with a template so you don't have to build or code anything. Are there any advantages to learning code and doing that compared to sites like WordPress and Squarespace where it's already basically done for you?

RYAN WHITLEY: Sure. So the question has to do with a template site like WordPress or Squarespace where you might not have to write code, but you're just editing a template. And basically, what's the difference between that and is there an advantage to that over learning how to code?

And I would say that an analogy might be-- I'm trying to think of an analogy on my feet here. But it might be something like using a calculator versus knowing how to do the math on paper. So using something like WordPress is great for getting something up quickly. But as soon as somebody-- let's say your manager says, hey, actually I need this button to fly through and go to the other side.

Those sorts of things probably won't be built in. And at that point, you'll have to take manual control and go add your own code to make it do that. So--

SPEAKER 1: So if you hate math, then you should use WordPress and Squarespace.

RYAN WHITLEY: Yeah. So for easy-- yeah. So you'll have more leverage and more control if you know how to do some of the things with code than you would be-- you would be limited with just editing the template through Squarespace.

SPEAKER 1: Well, I'm glad you said that because the only advantage that I can think of is that with coding and just building it from scratch, it's original. And somebody else might have the templates from WordPress or Squarespace. But I didn't think about the aspect that you said, like if your employer wants something different on there. You won't know how to do it.

RYAN WHITLEY: For sure.

SPEAKER 1: OK.

RYAN WHITLEY: And so that's a good point is if you are using a template from one of these websites and you want your site to stand out, that might not be the best choice. Because sometimes if somebody's looking at your website, they'll say, oh, that's just one of those

WordPress templates. Not that there's anything necessarily wrong with that. But if you're trying to stand out and be unique and be creative and add your brand to it, then really having full control over that is a better option to make it a more unique experience.

Any questions online or in person?

THOMAS WILSON: I have a question.

RYAN WHITLEY: Yes.

THOMAS WILSON: So Ryan, you talked about how tech has really pervaded into all types of different industries. But if somebody was interested in going into sort of the field of being the developer or something along those lines, a coder or something like that, where would they kind of start?

And do you have to be a mathematical genius to really get going in that area? And what are some of the careers that are out there? And you talked about some of the salaries and things, as well, that you've from beginners?

RYAN WHITLEY: Sure. So you don't have to be a mathematical genius, I think. So there are applications that would require a lot of math. If you're doing some sort of modeling the universe in a supercomputer, of course, you would have to know the math and the physics in order to program that simulation. But if you're not doing that, there are math libraries that come built into all these languages to help take that pain away from you.

So I would say don't worry about the math part. It's not a prerequisite. But I will say, in order to get into development, I think you have to be inquisitive. In order to really thrive, I think it helps to be inquisitive because it takes persistence in order to dig into this. And you're the type of person who wants to know how things work. You dig under the covers to find out what's really going on. And I think that makes people successful at coding.

You can read a book. But that will only get you so far. You actually have to roll up your sleeves and get in there and try stuff out. Trial and error, I think, is a great way to learn.

In terms of positions and salaries and things like that, obviously software engineering and coding is in high demand. There are tons of jobs right now, and especially if you find a niche. And what I do in my day job is I work a lot with maps-- mapping data, spatial data. We do analysis and answer questions for, like you said, the Gates Foundation. And we take satellite imagery from NASA and we mess with it and answer questions with it.

So that sort of thing is in high demand. Salaries are high. I mean, I don't necessarily want to identify something. But I think a starting programmer, you might be in the $50,000, $60,000 range depending on what you're doing. Or if you come out of school with a degree in computer science-- and again, if you specialized in a niche, like maybe you are a mathematical coder or a

specific type of coder that solves a specific type of problem, the salaries can go way up from there.

And the types of jobs that you might get into-- I'm in the field of consulting. So what I do, day-to-day, is sort of I wear different hats. And one of those is coding. But I also am a consultant, which basically means I have to go-- when I work with a client or a company, I have to go to that office. And you sort of blend in to that company. So for three months or six months, you are part of the Gates Foundation or NASA.

So it's great because I'm exposed to a lot of different companies and company cultures. And you get to see what it's like to work in these different places. But I also have to have good people skills, communication skills. I have to be able to talk to somebody who's nontechnical and extract from them what is it you're trying to build? They come to us wanting to build something. I have to convert what they want into technical requirements so I can go build it. And then we're all happy at the end of the day.

So being able to communicate, to talk, that's very important, being able to do things like estimation. So I have to estimate how much a software project might cost. And that's the budget we're given, if we're lucky. So there are other soft skills that go along with what I do. So consulting is one avenue for a programmer.

There's another one, which would be to work at a product shop, like maybe something like Amazon or Microsoft, where you're building a software product that's like a commercial product. And it's a whole different world from what I do in my day job. You might be more focused on the coding and less focused on communication, although I think communication is important for probably any job that you're going to get.

But that's another avenue, the commercial space. There's research, scientific research that, like I mentioned, you're modeling the universe inside of a computer. And so that's sort of scientific research. There's a whole aspect to that.

So there's really a world, an entire universe of jobs and opportunities. There are things like I mentioned earlier-- elevators. You've got to write code for automatic pilots and for artificial hearts. I mean, all of that, somebody's writing all of that code. So yeah, just about anything you can think of, it's being done in coding. And like I said earlier, if you blend your interests with some sort of technology project, then I think that would make it more interesting to you and you might want to pursue that more.

Any other questions? Yes?

THOMAS WILSON: So what if you're a social sciences major or something like that, along those lines? Do you still think you could-- could you kind of self-teach yourself a lot of different programming languages and things with some of the tools that are out there and use that in your job or if you wanted to go into computer programming type of positions?

RYAN WHITLEY: For sure.

THOMAS WILSON: Or do you have to have a degree?

RYAN WHITLEY: No, no, no.

THOMAS WILSON: Do you find that companies also want a degree every time, all the time? Or has that sort of fallen by the wayside.

RYAN WHITLEY: No, I don't think that's true. I think if you did not have a degree and yet you spent your personal time learning how to do something and you are the foremost expert because you spend your own time doing it, a company would hire you in a flash because you know you're doing. And having a degree is probably less important than you being the expert at something.

Regarding the social sciences, there are all kinds of social sciences and research software packages. One of them's called R. It's a statistical package. I know it's popular in the research community. anything package is SPSS. I don't remember what that stands for at the moment.

But it does a lot of things, for example, in geography or if you're trying to figure out migration patterns or disease tracking, disease patterns. You can input some of your research data into it and model the outputs, sort of figure out how a disease spreads or what the outcome's going to be.

So yeah, there's tons of applications for social sciences. And I think it's definitely worth digging into a little bit. If that's what you're into-- if you're into social sciences and you're doing a research study, definitely figure out how you can use a software package and get in there and write some pieces of code that will answer a question that you're trying to ask.

Yeah. I think that's all I have, if there are no other questions. And thank you for joining and thank you for coming.

SPEAKER 2: Thank you.

[APPLAUSE]