Learning Technology in the Information Age

by Dan Appleman

Start Course

Bookmark Add to Channel Download Course

Table of contents Description Transcript Exercise files Discussion Learnin

The World Has Changed - and That's No Cliche

Introduction

Learning Technology in the Information Age. In this course, I intend to change the way you look at learning technology and education in general. I expect many of you will discover that the way you are currently learning technology is shockingly inefficient and that many of your assumptions about how to best learn technology are completely wrong. Now these are strong statements and you may be ready to dismiss them out of hand. After all, you've been learning technology all along, you've been working in the field, you've been getting good grades in your classes, you have certification. Take a moment and think back at the last technology you learned. Why did you bother? Did you think strategically and evaluate whether the benefits of learning that technology justified the effort? When you started, did you take the time to plan out your course of study or did you just grab whatever resources you happened to find, careening from website to book to course like a ball on a tilted pinball machine. How far did you get? Beginner, intermediate, expert? And did you choose the level that you wanted to reach ahead of time or just kept studying out of habit? That last one is a trick question. As it turns out, thinking of learning a technology as a spectrum from beginner to expert is hopelessly obsolete. I'm not going to waste your time here with clichés or obvious approaches like taking a class or using online search-defined information. That's common knowledge and the real problem is that common knowledge itself is obsolete. In this course, we're going to demolish some well-loved clichés and assumptions and explore what it really means to learn technology in the Information Age.

The Information Revolution

Now here I was promising to demolish well-loved clichés and here I am starting with one. We live in the Information Age, the world has changed, it's the information revolution. What a cliché! And what does it bring to mind? Flat screen TVs, smart phones, and endless gadgets. So let's get one thing clear. All of those gadgets and technology are superficial and irrelevant. I love gadgets as much as anyone, but those are not what makes this the Information Age or makes this era the Information Revolution, because you see, the term Information Revolution is not a cliché. It is a reality. One-hundred years from now they will refer to this period in the same way that we look back on the Industrial Revolution, the Renaissance, and the Agricultural Revolution and this is not marketing hype or general hyperbole. There is something new under the sun and its impact on human society is, and will continue to be, as significant as the discovery of agriculture, the printing press, and the steam engine. In fact, there are two new things under the sun. First, for the first time in human history it is possible for any individual to communicate directly with any number of other individuals and to engage in commerce with any number of other individuals at effectively 0 cost and with 0 delay. In the past, it has been possible for a few people to communicate with many people. The printing press and later broadcast media saw to that, but for anyone to do so, that is new. Second, for the first time in human history, every individual can have access to virtually all human knowledge at effectively 0 cost. A few of you may object to this point, noting that it's not true for every individual on the planet and that is true, but it's also being picky. The technology to support what I describe exists and the trend in that direction is clear. We are well on the way. These changes are having a huge impact on every aspect of society. Many of the consequences are obvious, though we don't always see them in the context of these two facts. The growth of e-commerce and online retailers, the increasing mismatch between available jobs and people with the skills needed to fill them, and the wide-spread adoption of social networks are clearly results of these two revolutionary developments. Despite its best efforts to resist change, education is being impacted as well. I'm not referring to the obvious changes, the presence of computers in classrooms, computer-aided training, testing, and even online courses like this one. Like gadgets, those are just symptoms, today's expressions of the massive underlying changes that are occurring and are but a hint of things to come. The change that really matters can be found in the death of a cherished phrase, one that is not just obsolete, but is now a blatant lie.

Knowledge is King - The King is Dead

You've heard the phrase, Knowledge is King, Knowledge is power, knowledge is the most valuable commodity. Well if knowledge is king, the king is dead. Once, these phrases were true. One would study to gain knowledge and your knowledge could open the door to opportunity and respect. You could be the one people would go to for answers and would be paid nicely in the process. One of the consequences of the information revolution has been a sudden and extreme devaluation of knowledge. Knowledge is now cheap, indexed by Google, and delivered directly to your computer or phone instantly. Wait, you may argue, that's not knowledge, that's information, they aren't the same thing and while that's true, it doesn't really matter. Both have experienced the same devaluation. Ask a question or ask for help, chances are good you'll find the question asked and answered already, your problem solved, and the information in a correct context. What? You think that doesn't count? Well, if you have a problem to solve and you can solve it by bringing somebody else's knowledge to the task at virtually no cost, how important is it that you yourself have the knowledge to solve the problem? I know that's almost blasphemy, and I bet you can come up with many reasons why it is important that you do gain the knowledge yourself, but from an economic perspective, if the knowledge needed to solve the problem is readily available for free, the value of you having that knowledge yourself is effectively 0. In the realm of technology, it's not just the knowledge that has been devalued, so has the technology itself. Gone are the days when using or building technology required a small fortune in hardware and software. The cost of hardware of all types has crashed. It's been years since you needed access to university or corporate laps to have hands-on experience with most technologies. Today, individuals can afford all the technology they need to learn whatever they want and the tools will probably be better than what you'll find in school or even at work and when it comes to software development, almost everything you need is free. Open-source alternatives to paid software are widely available and sometimes just as good or better than the commercial equivalents. If knowledge and tools are cheap, what is left? The answer is skills. We are no longer a knowledge-based society, we are a skills-based society. Most people just haven't' realized it yet, though the signs are all around us. Once, almost any college degree guaranteed a good job. Now it doesn't. Nobody cares what you know. They care what you can do. Having knowledge is not enough. Knowing how to use it, whether it is your own knowledge or someone else's, is everything. So in this course, the emphasis will not be on learning, at least not in the traditional sense of acquiring knowledge. Our emphasis will be on gaining skills, learning how to put to use knowledge that you have and knowledge that you acquire as needed. Qualifications matter. Even when I graduated from school, there was a common belief that when it came to technology, nobody cared where you went to school as much as they cared about what you did on your last job. It was true then and it's more true now. It's not enough to be able to do something, you have to prove it. You need evidence. Formal degrees matter more and less than before. More, because they do represent a form of certification that you know something. Less, because knowing things is less important than it was and because there are more and more ways to prove your skills and knowledge. As we continue, we're going to look more deeply at the consequences of these changes, how they demand that we look differently at learning technology, and on the strategies that you can use to take full advantage of the information revolution going forward.

Learning Paths

Learning Paths

When we think of learning anything, there is a tendency to think of it as a journey from beginner to expert. We start out at the beginning of a path, knowing nothing about a subject, then travel along the road picking up bits of knowledge along the way until we reach a finish line that declares us an expert. Except that, in technology at least, as soon as you reach a finish line the officials congratulate you and mention that, by the way, we've moved the finish line out beyond that distant hill, so you really should start running again. Even if we do question this view of learning as a journey, it's usually to observe that the path is never straight. It has twists and turns, road blocks and potholes, and occasional dead end spurs where we discover that the things we learned turned out to be completely wrong. This model of learning as a journey was occasionally a useful tool a decade ago, but it's completely flawed for the Information Age. instead, look at learning as four distinct paths or tracks. I'll call them fundamentals, information, skills, and innovation. Each one of these represents a distinct journey. Dividing learning into four tracks this way gives us a mental model to better understand the learning process and the characteristics of different types of knowledge.

The Fundamentals Track

The fundamentals track is where you learn fundamental concepts relating to the technology you are interested in. It's where you learn how to speak the language, call it the basic literacy of the technology. If you want to study network configuration, this is where you might learn about internet protocols and the concepts behind acronyms like TCP, UDP, and DNS. If you want to be a software engineer, you would pick up the concepts behind common language elements such as data types, arrays, and principles of object-oriented programming. I remember going to the West Coast Computer Fair when I was in college and hearing everyone talking about the different types of computer buses. I found this very confusing as I had absolutely no idea what a bus was doing in a computer. I did guess that it wasn't a real bus, of course, but I had no clue what the word meant in this context. At that time, I had no real idea of how a computer worked at all. It wasn't long before I figured it out and, being an electrical engineering major, as well as a computer science major, I not only learned what a bus was conceptually, but ultimately learned aspects of computer buses of which most computer scientists remain blissfully ignorant, topics like line termination, use of ground planes, electromagnetic interference, capacitive effects, current handling, and high frequency characteristics. In other words, on the topic of computer buses I went from beginner to a high degree of competence, though I never approached real expertise. Fundamental concepts are a type of knowledge. They are widely applicable. Your knowledge of internet protocols applies universally to any piece of network hardware. Object-oriented programming concepts apply to numerous languages for all that the syntax and implementation may differ. Virtually every computer has a bus and the concepts I learned in college are just as applicable today as they were then, which brings us to the other characteristic of fundamental concepts. Though new concepts do show up and existing concepts sometimes are improved on, they rarely go completely obsolete. They represent the foundation that allows you to understand additional information and knowledge that is more specific to particular technologies. It is wrong to look at fundamental knowledge as the exclusive domain of beginners. I assure you there was nothing beginner level to the analysis we had to apply when calculating the effects of line termination on a bus. Fundamental concepts can be complex and difficult to learn. While I'm sure there are people who can pick up any concept quickly, for myself I find that it usually takes three attempts, learning a concept three different ways before it really sticks, and along the way there is a lot of banging my head against the wall in frustration. I remember when I was first introduced to object-oriented programming. there was a week or two where I just didn't get it. I talked to the teaching assistants and asked numerous questions, but it didn't make sense. Then one day in class something clicked, lightning struck, and I got it and while there have been advances to object-oriented programming since then, that basic understanding I gained so long ago is as valuable and applicable today as it was then, maybe more so.

The Information Track

To work in a technology, you must inevitably acquire knowledge and information that is specific to that particular product or technology. We'll call it the information track for convenience. Where your fundamental knowledge might cover what a router is and what it does, the information track might cover the features and programming of specific Cisco routers. Where fundamental knowledge of computer languages teaches you how an integer is stored and manipulated by languages in general, the information track teaches you how to declare and design one or more integers in C#, and later, as you gain expertise, how variables are boxed and unboxed or how you can use reflection to dynamically access integer variables by name. logic suggests that one must learn the fundamentals of a technology before being able to use or understand information specific to implementations of the technology, but this is one case where the logic is simply wrong. If you've been involved in technology for any length of time, you've probably met someone who has on hand a vast amount of information on a topic with no real understanding of what they are talking about. I remember working with one programmer who was struggling to debug an application. He had assigned an untyped variable with the string representation of a number instead of the numeric value, "4" instead of 4. Now this, in and of itself, wasn't the problem. We've all done that on occasion. The problem was that on discussing the issue with him I came to realize that the developer didn't even understand the difference between a number and the text representation of that number, which means he didn't really understand data typing, conversions or charactering coding, yet he was working as a software developer. A solid understanding of the fundamental concepts of a technology gives you a huge advantage when it comes to learning and understanding specific implementations and applications of a technology. People who go directly to the information track can be effective, but there are almost always holes in their knowledge. Unfortunately, much of our technology education is focused on learning specific information, learning a particular language or platform. Beginners who take these courses often miss out on fundamental concepts and have no idea what they are missing. One of the main characteristics of knowledge on the information track is that it goes obsolete very quickly. Your knowledge of best practices for configuring a domain controller or a mail server become obsolete with each version update, platforms evolve rapidly, even languages that are based on standards change. JavaScript echo script may seem relatively stable, but in most cases, its use is tightly bound to one of many JavaScript libraries, which evolve rapidly. Information is also acquired quickly, especially if you have a good knowledge of the fundamentals of a technology. I discussed this in my course, Career and Survival Strategies for Software Developers, in the module titled, The Just in Time Programmer, pointing out that many software developers today really do learn technology as it is needed, rather than studying broadly with the hope that something will prove useful someday.

The Skills Track

Have you ever met someone who seems to know everything, but can't actually do anything? Learning to actually do things is what the skills track is all about. I started working part-time as an engineer after the summer of my sophomore year in college and it became quickly clear to me that I was learning as much at work as I was at school. Not only was I picking up the specific information that related to my company's products, I was learning how to do things. I had to design circuits using real parts, not the theoretical devices from classroom examples. I had to consider issues like price, availability, and lead time, that almost never came up in class. I had to document my work, not for a graded report, but according to the company standards, and I had to build it and debug it and it had to work, not because I needed to get a grade, but because it was my job to make it work. It became clear to me that my school was doing a great job teaching fundamentals and even some information, but real skills you gained on the job and, in fact, it was common knowledge that new grads didn't really know anything useful, that it was up to companies to give them the skills they needed to get the job done. I once had the experience of working with an engineer with a graduate degree. At first, I was somewhat intimidated having just a couple of bachelor degrees and being younger and less experienced. His project ran into trouble and I was asked to take a look and see if I could help. I started digging into the design and was horrified at what I found. The guy had designed a project that in theory might have worked, but could not work in the real world using real parts. It was fundamentally flawed at the architectural level and needed to be completely redesigned. He had plenty of knowledge and information, but did not have the skills to translate them into reality. In a world where knowledge and information are both inexpensive and widely accessible, the value of skills has grown. The longevity of skills varies depending on the skill and the knowledge on which it is based. Knowing how to wire up a network cable or interpret the contents of a TCP packet is based on fundamental knowledge, so those skills can last a long time. Being an expert on Visual Studio 2010 was a skill that lasted about two years. Then, those who wanted to maintain their skills had to study the new features of Visual Studio 2012, then Visual Studio 2013, and so on. The key thing to remember from a career perspective is this: fundamental knowledge is the foundation that helps you understand and digest information. Information is the specific knowledge that you need to solve problems. Skills represent the ability to implement solutions using your knowledge. As you increase your level of expertise on the skills track, you can come up with better solutions and implement them more quickly with less chance of mistakes along the way. In most cases, it is the skills that generate income. People pay you to do things, not to know things.

The Innovation Track

I was originally planning to call the innovator track the experts track, but I came to realize that the term, expert, is itself confusing and misleading. I mean, what is an expert really? I, myself, have often described an expert as someone who has a really good understanding of fundamentals, but that's really a reaction to the fact that so many so-called experts don't. So let's take a quick look at what the progression from beginner to expert looks like on each of the tracks I've described so far. Everyone starts out as a beginner knowing nothing. If you're lucky, you have some understanding of another field that is similar enough so that some of the concepts carry over, but it's always a struggle to start out because beginners don't even speak the language of the domain. TCP, OOP, GPU are cryptic acronyms and because the explanations of a technology sometimes don't explain the terminology, it's twice as hard to make progress because you don't even understand the explanations. On the skills track, you may even lack the most basic knowledge. How do you find that specific configuration page the documentation says you need to use? What is a variable? Where is the on/off switch? As you progress, you start to speak the language, so you can understand the documentation and follow tutorials. You can read and understand sample code. You may even be able to follow specifications, references, and product manuals, and that's good because at this level you have to refer to them constantly. You can solve problems, but it is very slow and you make many mistakes along the way. You get stuck often. Next, you reach a level of competence. You understand the basic concepts and are constantly refining your knowledge of fundamentals. You can effectively use search to gather additional information as needed and you consult most problems in a timely manner. You rarely get stuck. This is where most good technologists spend most of their time. So what comes after competence? Let's call that expertise. Experts can solve virtually any problem that is solvable. They have access to the fundamental knowledge, information, and skills needed to accomplish that, but let's take a closer look at that statement. Experts can solve virtually any problem that is solvable. That also means that part of being an expert is identifying problems that cannot be solved and understanding why they cannot be solved. Experts have access to the fundamental knowledge, information, and skills. I didn't say they themselves have the necessary knowledge and skills, or even that they can find it. Access is much broader than that. If an expert can't find what they need, they create it. Isaac Asimov wrote a great short story once called Profession about a world where everyone learns both information and skills through a form of hypnosis. The knowledge is imprinted directly on their brains. But the story asks an interesting question. Where does the knowledge come from? Who makes the knowledge tapes? True experts are innovators. They are the ones who discover or invent the fundamental concepts, who write the blogs and the books and the articles, who figure out the best way to do things. They create knowledge. They invent solutions. They define best practices. The innovator track is really part of each of the other tracks, representing the journey beyond competence.

Maximizing Value

Expertise is almost always narrow and is often fleeting. I was once an expert in Visual Basic 6. That is no longer true and it wouldn't matter if I was, because few people care anymore about that technology. I was once an expert on. NET development and while I remain competent in many aspects of that technology, I am no longer an expert. I am currently an expert in software development using the Apex language on a platform called Force. com, but I'm competent in a wide range of technologies and routinely use several other languages and platforms on a daily basis. There is a natural tendency to want to become an expert, but it turns out that this is rarely the best learning strategy. Learning is fun, or should be, and if you want to learn something because you are excited about it and want to learn everything you can about it, please don't let anything I say discourage you. Right now, I want to talk about learning from a career and value perspective or put another way, maximizing the financial returns from the effort you put into learning technology. Let's start at the beginning. Being a beginner rarely pays well, regardless of what track you are looking at. It is a necessary first step towards competence, but beginners are slow and make mistakes and the supply of beginners is generally large. The law of supply and demand comes into play. Low quality with high supply results in low wages unless demand is truly extraordinary. You can get jobs as a beginner, in that companies often recognize that they will need to train people, so they may hire someone without the skills they need, but who have other characteristics that they care about. This is especially true when there are not enough competent people to go around. Companies hire new graduates knowing full well that they probably don't have the skills needed for the job, based on a belief that they will learn quickly, especially if they have strong knowledge of fundamentals. That's why companies like Microsoft, Google, and Facebook hire heavily out of top computer science schools. They know that while those students may lack skills, they will have significant competence on fundamentals track and may be strong on the information track as well. Competence is where the professionals play. If you have a decent knowledge of the fundamentals, the ability to seek out and understand information quickly as needed, and a broad skill set, you can efficiently solve problems using a wide variety of tools and technologies. Having competence in a variety of technologies is particularly valuable today, where much of software development involves integration of different technologies. It also offers a high degree of job security, in that your skills can qualify you for a wide variety of possible jobs. In other words, competence pays. Reaching a level of competence on the fundamentals, information, and skills track for a particular technology is the most efficient way to maximize your income from solving problems using that technology. But what about expertise? Believe it or not, gaining expertise in an area is rarely the best use of your time from a financial perspective. That may seem counterintuitive, but hear me out. First, think of it from a company's perspective. If a beginner can do a particular job, would they spend more money to hire someone more experienced? Maybe, maybe not. Companies hire competent individuals because they can get the job done where others can't. Now, if competent individuals can solve 95% of all technology problems, which may be an understatement, it doesn't make any sense for a company to hire experts at a greater price. Sometimes it does, say, to work on an area that is core to the company's success and sometimes it's good to have an expert around just in case, but you rarely need a whole team of experts even if you could find them. Thus, experts aren't necessarily paid that much better than those who are just competent. So why become an expert? You may not get paid significantly more, and once you become an expert you have to work really hard to stay an expert. Being an expert is hard work. Understanding information that exists is much easier than the kind of digging and experimentation that it takes to create new information, and the time it takes to gain and maintain expertise on a subject is time you don't have to become competent in other technologies. Now there are particular career strategies where is does make financial sense to gain expertise. If you want to be an author or speak at conferences, becoming an expert is an effective way to start. Independent consultants can often benefit from having areas of expertise. Sometimes companies really do need an expert to solve a problem and will pay a lot to borrow one for a while, but the truth is that many experts become so, not out of strategic choice, but by accident as they work on a technology that they are passionate about, and it is not my intent to discourage that. It's exactly what happened to me with Force. com. My real point here is to convey that you should not feel the need to go beyond competence unless there is good reason, that being competent in a technology is good enough and once you reach that point it is often just as good or better to switch over and learn a new technology than to continue on the path to expertise and innovation.

Choosing Your Domain

Learning always has a cost and that cost is one of the key factors to consider when choosing the domain you want to learn. I'll talk more about costs of learning in the next module, but for now I want you to keep one thing in mind. The real costs of learning are not measured in course tuition and expenses for books and tools. The real cost of learning is in time and learning always takes time. Because you don't have time to learn everything, it is important to think strategically about what you are learning. Think about it, when you make a major purchase, you take the time to compare brands and products. If you're buying a car, you research safety, gas mileage, features, and comfort. You want to be confident that you'll get a good value for the money you spend. Learning a new technology is like making a purchase where the cost is your time. It's worth the effort to compare technologies and choose the ones you learn based on the value you will get in return. In many cases, the value is monetary. Will learning the technology help you make more money or at least preserve your current income and will the financial benefits justify the cost? But in some cases, the value has nothing to do with money. Learning the technology may improve your reputation among your peers or it may just be a technology that fascinates you. The cost to learn a technology depends both on the nature of the technology and how far up the knowledge tracks for that technology you wish to go. Let's take C# for example. Learning the C# language isn't hard. If you already have a good understanding of the fundamentals of computer languages and experience in other object-oriented languages, you can become a competent C# programmer in days, at least when it comes to the language. However, the real cost of learning C# is not in the language at all, it's in the platform. To develop C# on. NET, you need to learn about the. NET framework, one or more. NET technologies such as ASP. NET or WPF, and the Visual Studio development environment. The cost to become competent in. NET development is typically measured in months, even for an experienced developer. Learning a platform is always more costly than learning a particular language, so choosing the platform is the more critical decision. If you're going to be a mobile developer, do you develop for the iPhone, for Android, for Windows Phone or do you stick with HTML5 or try to find a tool or environment that lets you target multiple devices even though it compromises on your ability to take full advantage of each platform. Because the real cost in learning is always in time, choosing one technology always implies limiting or sacrificing others. For example, I am very competent with the core Windows operating system, though not with Windows Phone or RT. I'm competent on multiple levels, ranging from low-level internals to high level configuration, but, and I'm almost embarrassed to confess this, I've never used a Mac. I mean, never. If you sat me down in front of a Mac, I'd be an absolute beginner, not even knowing how to use the most basic UI functions. I don't doubt that I could learn them if I wanted to, but I've never had a particular reason to do so. If I did for some reason need to become a Mac developer, I could do that as well and I'd be able to draw on fundamental knowledge from other technologies to speed the process, but honestly, I can't see any way to justify the effort, so I will probably remain a Mac neophyte for the foreseeable future. How do you determine the potential value of learning a technology? The easiest way is to look at job opportunities. Hot technologies are in high demand. If you see lots of openings in a field, that's a good sign. Look at salary ranges offered to see if skills in a particular technology are commanding unusually high salaries. Don't just look at languages, remember languages are easy, look at platforms. Remember that platforms are not just operating systems. Major applications are platforms as well. Oracle, SAP, Facebook, Eloqua, and Salesforce are platforms, and developers who have confidence building applications on those platforms can often command much higher salaries than those with generic operating system experience. I mentioned earlier that I've become a Force. com expert where we use the language, Apex, that most of you have probably never heard of. Trust me, I wouldn't have invested the effort to learn this particular platform if it wasn't very worthwhile. Don't just look at platforms, look at architectures. Do you know how to configure and manage a Linux server? That's nice. Do you know how to do so on Amazon Web Services? That's interesting. Do you know how to do so on Amazon Web Services in a way that can handle fallbacks to different data centers and quickly scale according to demand? You're hired.

The Time Value of Learning

When choosing where to invest your time learning, it's also essential to consider how long you expect the knowledge to have value. After all, the sooner knowledge becomes obsolete the less time you have to capitalize on it. Let's take a second look at the time value of knowledge on all four tracks. Generally speaking, knowledge gained on the fundamentals track is very long-lasting. When it does become obsolete, it's more likely to be transferable to something else. For example, a few decades ago, having a strong fundamental knowledge of serial and modem ports had great value. Serial I/O ports running at speeds ranging from 110-32K bits per second were used to connect almost any peripheral from printers to mice to modems. The specific information on how to configure device ports or control your serial ports on the operating system or individual serial modem commend sets would vary and change, but the serial protocol remained the same. The skills you needed to attach and sometimes wire different serial connectors and cables or craft complex modem commands would also vary, but again, the serial protocol mained the same. Today it's virtually impossible to find a computer with a serial port. Hardly anyone uses the old Hayes AT command set, except for those few people who still have to write or maintain fax modem drivers. I haven't wired up a serial cable in years. The fundamental concepts relating to serial ports are largely obsolete. However, if you did understand those concepts of how it was possible to transfer data quickly and accurately on a single wire, if the time came where you needed to learn how USB or serial ATA connections work, you would have found the transition remarkably easy. Some of your fundamental knowledge would transfer to the new domain. This is also why it is still worth investing even large amounts of money to get an engineering or computer science degree. Those degrees tend to focus on the fundamentals track, so even though the education is expensive, the payback lasts for many years. The information track is the exact opposite. Knowledge that is specific to a product or platform or anything that changes and versions frequently, tends to go obsolete very quickly. Just a decade ago, many products would typically version every 2 or 3 years, but in these days of continuous updates and releases, it's not uncommon for a platform or class library to update every few months or even more frequently. In these cases, learning a technology is not a one-time investment, it is an ongoing commitment. For example, the Force. com platform that I work on these days versions every four months. That means that choosing to learn Force. com involves not only an upfront investment in time to gain competence in its current state, it's a commitment to spend additional time every four months to stay up to date. Salesforce even requires you to take a certification exam every four months if you wish to remain certified. This rapid obsolescence of technology is one of the greatest challenges that those of us working in technology face. I discuss this point at some length in my Pluralsight course, Career and Survival Strategies for Software Developers. The skills track shares characteristics with both the fundamentals and information track. Because skills represent the ability to apply knowledge, those skills that are based on fundamental knowledge tend to last a long time. Those based on specific information tend be short-lived. For example, when studying computer science, you're likely to learn many ways to sort an array of data, different algorithms for doing so, and how to evaluate a sorting algorithm. The skill of being able to craft a sort routine is long-lasting. However, when facing a particular task that involves sorting, there are a number of things you need to do before you can use that skill. You would first check if any existing sort implementations that are in the language or platform are suitable for your needs. You might evaluate the systems on which the software will be running. If they are more likely to be limited by memory size, disk performance or CPU speed, and what size and type of data sets need to be sorted. All of those factors represent knowledge that not only may vary based on your situation, but can quickly change over time as the tools or software evolve. Your fundamental knowledge-based skill in understanding how to evaluate a sorting strategy will definitely come into play, but once you determine that the best approach is to implement your own sort routine, the skills required to implement it on a specific platform may be completely inapplicable by the time the problem comes up again. The innovation track is similar in this regard. If you are an expert on the fundamentals track building and discovering new principles and concepts, you can count on holding your expertise for a long time. College professors count on this. Being an expert in a particular product or platform requires constant effort and it's much harder and more time consuming than just staying competent. Understanding the time value of knowledge is one of the main reasons that I took the approach of dividing learning into four tracks. Clearly the lines between those tracks are vague. That's not the point. Thinking of learning as four distinct tracks provides a tool, a mental model that helps evaluate what you are learning and what are the likely benefits of doing so. In other words, the idea isn't to look at a set of knowledge and say, oh, this is just information and will go obsolete quickly. The idea is to look at what you're learning and ask yourself, how long do I think what I am learning will have value. If the answer is that it will become obsolete quickly, it's probably just information and you might want to think twice if it's worth the effort to learn it. If the answer is that what you are learning will be long-lasting, you know that it is probably fundamental knowledge and might be worth an even greater investment because of the longer time that you will have to benefit from it.

The Three Branches of Knowledge for a Career in Technology

Looking at learning technology as four tracks is useful, but it's not enough. Because you see, there are really are twelve tracks. In fact, there are three distinct branches, each of which has all four of the previously described tracks. So far, I've only discussed the technology branch. That is the one that this course will mostly focus on. However, relying on technology alone can significantly limit your career opportunities. Gaining knowledge in the other two branches can help you maximize the returns from your knowledge of technology. The first of these is the business and finance branch. A career in technology is a career. That means you're making use of your technological knowledge and skill to make money. It also means that you are probably working for a company that is paying you in order to make money for the company. Understanding the economics of software development and the economic trade-offs behind technological choices will make you more valuable. It will allow you to come up with more efficient solutions to problems. It will help you avoid mistakes. For example, if you are tasked to choose a voice-over IP system for your company, your technological skill will help you choose the one with the best features, but your business and finance skills will help you to evaluate the stability of the provider, so that you don't deploy a new system a few months before the vendor goes out of business. This also applies to software developers choosing a tool, product or class library. Companies go out of business all the time. You don't want them to take your company down with them or even make you look bad for choosing them. The next branch is the leadership and management branch. The best technical skills in the world are useless if you can't convince people to let you use them, or worse, use them only to have others get the credit and salary increases that go with them. Having great ideas can be frustrating if you can't get others to listen to them and follow. You can't be a consultant if you don't know how to work with and manage clients. Leadership and management skills are multipliers of your technical skills. I have three other courses that address these two branches, so I won't be covering them here, other than to say that the principles of learning technology are very applicable to them as well. The courses are: Career and Survival Strategies for Software Developers, which focuses on the challenges and strategies of managing a career in technology, So You Want to be an Entrepreneur, which is particularly focused on the business and finance branch, Introduction to Leadership and Management For Developers, which focuses on the leadership and management branch. Despite the titles that reference developers, all three courses should prove equally valuable to anyone working in IT or engineering. Skills in the business and finance and leadership and management branches are often referred to by the term soft skills and are often dismissed as unimportant or of secondary importance by those of us in technical careers. This is a mistake, as both serve to magnify the effectiveness of your knowledge and skills in the technology branch. These soft skills have one other huge benefit. Unlike knowledge from the technology branch, the vast majority of soft skills never go obsolete. As a result, over time they become proportionately more valuable and because they too require time to gain, they also represent one of the best ways that older technology workers can have an edge and provide value over those who are younger. Look at it this way, let's say a company is looking for someone to work in a new technology and that the fundamental knowledge required to achieve competence in that technology can be gained in one year, who will they hire? An inexpensive younger person who knows the fundamentals and has some familiarity with the technology or a much more expensive person who has the same knowledge and experience? Remember, this is a new technology, so the older person's experience in other technologies provides little benefit, at least not enough to justify their higher cost. At first glance, you might expect the company to hire the younger person and many companies would, but in many cases, the older person might get the job. Not because they are stronger technologically, but because they may be more skillful at interviewing or demonstrate an appreciation for the business need to deliver on time and a willingness to buy solutions instead of building them if that meets the business need. They may know that sometimes you have to compromise on features or standards or build a technical deficit to meet a deadline. They may even be able to persuade the company that they represent a cost savings, a person who can serve double duty as a team leader or a group manager or an investment in the future, someone who will be able to grow into a management position as the company grows. So as you focus on learning technology, remember the other two branches. They will provide a strength and foundation to your career that technology alone can rarely sustain.

Knowledge, Skills and Practice

Knowledge, Skills and Practice

One of the major characteristics of the Information Age is that information is widely available. No, let me rephrase that. We are overwhelmed by information, we are pummeled by it. Every day we have incoming emails, tweets, chats, posts, blogs, news, texts, and all manner of other bits of information clamoring for our attention. If you want to learn something, the challenge is no longer finding information, it's filtering information. Which content is accurate, which content is relevant, which content is useful, what combination of content represents the most efficient way for you to learn what you want to learn? As a result, much of the time, our progress learning a new technology is much like that of the classic drunken sailor problem. We know we're going to get where we need to go, but we might spend an indeterminate amount of time wandering around in random directions trying to get there. The purpose of this module is to bring some reason and order to the process of learning and to see how one might address learning on all four tracks that we defined earlier.

The Nature of Schools

Schools then and now excel at teaching fundamental knowledge on a topic. This is particularly true for academic institutions as compared to vocational schools where theoretical grounding on a subject is everything and practical experience limited to an occasional lab. There are several reasons why this is the case. Consider who teaches at universities. College professors at academic institutions are primarily there to do research, but what kind of research? They aren't figuring out the best ways to use a particular product or technology. What they are doing is researching fundamental problems and trying to solve fundamental challenges. In other words, they are experts and innovators on the fundamentals track, so that's what they teach. Even today, a university education is one of the best ways to get a solid grounding in technology fundamentals. Their focus on fundamentals can, however, make them a bit weak when it comes to having the latest information. And the fact that they may not always be working with the technology, means they also may be weak when it comes to teaching skills. The story is a bit different at vocational schools where there is a greater balance between the fundamental track and the information track along with some real focus on the skills track, something often lacking at more academic schools. Teachers at vocational schools are more likely to be working in the field and thus be competent in both the information and skills track, but they won't necessarily have the deep understanding of fundamentals that the traditional academic professor would have and are less likely to be involved in innovative research. Schools also provide access to equipment that in the past was far too expensive for individuals to access any other way. Schools offer more than just knowledge and access to equipment, they offer curation. When you enter a degree program, you are guided through a set curriculum with a limited set of options. You must meet those requirements in order to graduate. You don't have to worry about filtering information, the school does it for you. Along the way, the school provides support. There are other students learning the same things that you are, so you can help each other and compare notes. There are professors and teaching assistants available to answer questions and it's a good thing, because let's face it, most academic text books are nearly incomprehensible without them. I was always astonished that the typical academic text in engineering and computer science managed to be perfectly clear for about three chapters and almost unreadable thereafter. Even exams are a form of support, in that they represent hard deadlines that can motivate you to get to work and avoid procrastination. Schools offer certification. Once you take and pass the required classes, the school certifies that you have learned the subject and employers can use that certification as part of their own filter when determining who to hire. Finally, schools offer the opportunity to gain real expertise in some areas should you wish to do so. Student research projects, advanced degrees, and post graduate studies allow you to enter the innovator track, at least when it comes to fundamental knowledge. Fundamentals, information, skills, curation, support, certification, and expertise, these are all part of learning technology and schools have traditionally provided one-stop shopping for all of them. But the world has changed. While schools, in many cases, do remain one-stop shops for learning technology, there are now alternatives and many of them are considerably less expensive and faster. Where before, schools had a near monopoly on access to knowledge and information, today knowledge and information are effectively free. Where before, schools provided access to expensive equipment, today, in many cases, the required equipment is easily affordable. Where before, you had to meet fellow students in person in a classroom, today's social networks provide numerous ways to connect with others regardless of geography. Schools are not the only ones in the curation business and there are other ways to gain a variety of certifications on specific technologies. And while it is still advisable to be part of the academic world if you want to be acknowledged as an expert on the fundamentals track, that is certainly not the case when it comes to anything else. The mistake that many people make is failing to look at all aspects of learning when considering these alternatives. Some think, I don't need to take a class, I'll just read some blog posts and tutorials, failing to realize what they are missing. The resources they find may provide the information they need, but may lack the fundamental concepts and provide little or inadequate practice when it comes to obtaining the required skills. It may be poorly curated, leaving out critical information and focusing on material that is unimportant. They may find that they really need certification or that the learning process is very slow due to lack of support and lack of the discipline that comes from being on a strict academic schedule. The rest of this module is dedicated to exploring the alternatives that are available for each aspect of learning. The goal is that when it comes time for you to plan your own course on learning a technology, you will consider all of these aspects of learning and factor each one into your plan based on your own individual needs and resources.

Where Do You Go for Curation?

Where do you go for curation? I can't tell you how many times someone has come up to me at a conference after a lecture or emailed me and asked what is a good book for beginners on some topic that I covered in a talk or book or blog post. In a world where information is available in vast quantities and low cost, curation has become increasingly valuable. Where do you start, what information is truly important to learn, and in what order should you learn it? If you can't answer those questions, learning a technology becomes infinitely harder than it should be. You waste your time learning things that aren't really important. You fail to learn content that you will need. Worse, you are exposed to content in no logical order. In some cases, you simply can't understand content because you haven't yet learned the prerequisites, the concepts and terminology on which it is based or you have to struggle at great length to understand something that you would have absorbed easily had you first taken the time to learn some fundamental concept. In short, poor curation dramatically increases your cost to learn, both in time and money. Curation is a big problem and one where I expect things will improve. At this time, there are some approaches that you can use to help define your own curriculum. With so much information available for free, many in technology are reluctant to spend money on books. As someone on a budget, I understand this. As an author, I am somewhat frustrated by this. Certainly, technology books do not sell nearly as well as they used to 20 years ago when books served as references and primary sources of information. That is no longer the case, but avoiding books is a false economy. It's not because books contain information that you can't find elsewhere, in most cases they do not, but books still excel in one area. They are a source of curation. A book provides content in a logical order where concepts build on each other. The content of a book is curated by the author who determines which content is and is not important. The curation alone is what justifies the cost of a book and books remain one of the least expensive sources of reliable curation. Websites, blogs, and articles are almost universally awful in this regard. Search is completely useless. Choosing which book to buy is relatively easy, just check out the online reviews and see what people are saying about it and when in doubt, buy more than one book. Having two authors is like having two guides. If you see the same content in both books, it's a safe bet the content is important. If you see it only in one, perhaps less so. And if you get stuck understanding the content in one, perhaps the other author will have a better explanation. When I am about to start learning a new technology, my first step is always to order one or two beginner books on the topic. They are the foundation on which everything is built. Online courses such as this one play the same role. In fact, I've argued that Pluralsight courses are really just a video and audio representation of a book. They, too, have chapters and have curation, but Pluralsight provides yet another level of curation, in that they maintain high standards both for the production of the courses and for who they allow to join as authors. As time goes on, I would expect to see even more curation with recommendations on order of courses to take in a given technology space. Just as curation represents much of the value of books, it represents an increasing percentage of the value of schooling. Degree or certification programs are the very definition of curation, defining a sequence of courses and required knowledge in order to obtain a degree or certificate. The nice thing is, in most cases you don't have to go to the school to obtain the curation. Schools publish the lists of classes required for degrees, the books required for those classes, and you can sometimes even find detailed lesson plans. You can also sometimes find sample exams or homework for classes, each of which serves to define content that the teacher included in the course curriculum. Borrowing curation from an academic program can be a very effective way to build your own learning path, but use caution when taking this approach for industry certifications. As you recall, academic degrees focus on fundamental concepts, so following their plan can give you a strong foundation on a subject, but industry certifications, particularly certifications on individual products, tend to emphasize information that is narrow and version specific, in many cases, relying on memorization more than understanding. What's worse, these certifications tend to deemphasize fundamental concepts, so planning your study around these certifications risks finding yourself one of those individuals who can use the technology without really understanding what you are doing.

Where Do You Go for Knowledge?

Where do you go for knowledge? Whether you are learning fundamental knowledge or information that is specific to a technology, the sources and approaches you use will be similar. Both involve seeking out and trying to comprehend content. Different people prefer different types of content. Some learn best by reading, some by watching videos or listening to audio content. You should consider the type of content that works best for you when creating your own learning plan. Let's take a closer look at some of the common sources of information that are available. I've already mentioned books. They aren't generally free, but they are inexpensive relative to other paid sources of information. While the quality of books does vary, it tends to be good overall because it costs time and money to produce a book, more than you might expect, so most authors and publishers want to make sure the quality is high. That increases the chance that book sales will cover the cost of producing the book and perhaps the book will turn a profit. Books are curated. Buying a book saves you the time and cost of filtering through the vast amounts of available information. While books that cover fundamental knowledge tend to be long-lived, most technology books are tied to specific products or technologies and thus tend to go obsolete quickly. The average life-span of a computer book 20 years ago was perhaps 2 years. Today it is 1, if you are lucky. Fortunately, e-book and print-on-demand technology makes it easier for authors to update books more frequently and websites make it possible to publish updates and corrections as needed. Books are an excellent way to start learning a new technology. Online courses and videos. I'm referring here to courses such as this one, that are webcasts or recordings that you view, as compared to interactive online classes. As mentioned earlier, these are the video equivalent to books and share similar characteristics, in terms of content timeliness, cost, and effectiveness. Online courses are ideal for those who learn better through video and audio as compared to reading. There are countless sites, blogs, forums, and wikis that contain useful information. Many include simple tutorials and explanations. These are ideal for filling in gaps in knowledge, obtaining answers to questions, and drilling down into specific areas of interest. They are generally less useful when it comes to acquiring fundamental knowledge and the curation tends to be poor. It's not that the people posting this content are not good, on the contrary, many of them are experts in the field. The problem is that these posts are almost always short and rarely related to each other. Finding relevant posts and articles on the web can be tricky. The good ones can be lost in the sea of obsolete or incorrect information. One new feature can change the correct answer for a technical question, but those old questions and now incorrect answers will remain on the web forever. Use date filters on searches and include version or product model numbers in queries to help find timely results. Trying to learn a domain using the web alone, is like taking a great book on a subject, ripping out the individual pages, mixing in 10 times as many pages with similar content, inducing content that has errors or is obsolete, then adding a key word index to allow you to pull up pages that contain certain words, and trying to learn the subject by picking random pages and using the index. Okay, it's not that bad, but the reason it's not that bad is because there are some sites that do provide some curation in the form of larger tutorials and links to related articles. And when the technology you learn is based on a product, you'll have access to all of the product documentation. That generally includes, you guessed it, books that are available to download or view online. The very newest and most accurate information for any technology is likely to be in what passes for the official documentation. Supported technologies, meaning a technology that is professionally managed, will maintain accurate and up-to-date reference material. Most commercial products meet this standard, as do the very best open-source projects. Do not rely absolutely on the accuracy of official documentation. One of the sure signs that you are becoming an expert in a field is when you start discovering and reporting on documentation errors and bugs in the technology. Product documentation sometimes suffers from limitations, sometimes it's inadequate and incomplete. This is what you will often see with open-source products, where the developers are far too busy working on the product to keep the documentation up to date. Sometimes there is just too much of it and it is poorly curated. A good example of this is Microsoft's MSDN Library. Now make no mistake, I love MSDN. I consider it the most important developer tool Microsoft offers, but it is overwhelming and while it is a great source of information, it is not always the best tool for learning any given Microsoft technology, which is of course why there are so many books and courses and classes on Microsoft technologies. Product documentation is often written by people who create the technology, but never actually use it or if they use it, it is in a limited and controlled environment, so while it is technically accurate, it can lack critical knowledge of best practices and the behavior and limits of technology in the real world. Or put another way, product documentation, white papers, reference samples, and tutorials will tell you how great the technology is, but will rarely tell you about the flaws, limitations, obstacles, frustrations, hidden costs, and all of the other problems that technologies tend to exhibit once deployed. While schools and classes still have significant value, as far as providing knowledge and information, they are far less useful than in the past. They are strongest when it comes to teaching fundamental concepts. They are weakest when it comes to teaching timely information. Schools have a difficult time staying up at the very leading edge of technology. It depends greatly on the teachers they have available. Their budgets may not cover frequent software and hardware updates. They can also be expensive, sometimes very expensive. Those costs can be justified by the other things that schools provide, but not as sources of knowledge. Schools and classes also bind you to their schedules, so it can take longer to cover the material than you would like, potentially a serious issue if you are on a deadline. There are situations where it does make sense to seek out a class or trainer. If you are looking to learn the fundamentals of a technology, the curation provided by a class can be invaluable and the content can easily be current. If you are an individual who best learns from a teacher, an environment where you are interacting with a teacher will be more efficient and may well be worth the added cost. If you need to learn a very new technology, for which there are not yet any books and the documentation is inadequate, hiring an expert to teach you can be effective and well worth the high cost. Remember, experts are the innovators, the ones who discover and create information on technology, so if you are working with a leading edge technology, this can be the only way to gain access to the knowledge without becoming an expert yourself. This is, by the way, part of the value that you can get from attending a session at a technical conference. The speakers at these conferences usually are experts and the information they share is often so new that it has not yet appeared in print or even on the web.

Where Do You Go for Skills?

Knowledge and information are both very nice, but as you may recall from the previous module, skills are what bring in the paycheck. People will rarely pay you for what you know, they pay you for what you can do, for your ability to use knowledge and information to solve problems. That's why a résumé that has a long list of accomplishments is usually more effective than one containing a list of classes you've taken. I say usually, because hiring and human resource management is a skill as well, and not all of those who do it are good at it. Now you may be one of those rare individuals who can read about something or watch someone do something and be perfectly comfortable doing it yourself, where just having the information is enough for it to come instantly to mind as needed when formulating a solution to a problem. For the rest of us, we gain skills through practice. That's why schools assign homework. It's why math classes assign those hated word problems. You can solve equations all day long, but word problems build the skills you need to figure out which equations can be applied to solving which problems. You don't really know how to configure a computer or network until you've done it. You can read all of the programming books and watch all of the programming videos in the world, but you're not a programmer until you've written code. To gain skills in a technology, you have to use it, you have to solve problems with it, you have to build things with it. In some fields, schools remain the only place to gain the skills needed for a profession. Only schools can provide future doctors with access to high priced medical equipment and sick patients on which to practice, though any

Army medic who has seen active duty might debate the point. Decades ago, schools still had the edge on teaching technical skills. Even personal computers were expensive and more powerful machines were beyond the reach of most individuals. Network hardware was costly, as was any network connectivity beyond that provided by a slow modem, but today that is no longer the case. Development tools and software are free or relatively inexpensive depending on your platform. The hardware needed to run it is also inexpensive. Hardware for more specialized training is inexpensive as well. You can get Cisco lap kits complete with routers and switches, everything you need to practice for their certification exams for just a few hundred dollars on E-bay, and it's even less when you consider that you can resell the hardware when you're done with it. Need to practice configuring a large network? Today's desktops can easily run a half dozen servers as virtual machines and if you need more, you can always use some on Amazon Web Services or sign up for some short-term virtual private servers on any number of hosts or go to E-bay and buy a dozen old laptops or computers to practice on. Here's the deal, I don't care how great a book is or how fantastic a course is here on Pluralsight or any other service, there is no substitute for hands-on experience and no excuse for not getting it. Many books, courses, and tutorials do include walk-throughs, where you are encouraged to follow along step by step with what you are reading or seeing and while these are not bad in terms of building some comfort with using a technology, just following along a tutorial is not enough. To take full advantage of a tutorial, you have to go beyond following the examples, you have to solve problems on your own. The way to really learn skills is through challenging yourself and through play. The first programming course I took was very much like any beginner's course, introducing concepts and giving us simple assignments, but I learned far more during that course than most of the other students. You see, one of my friends in that class and I were both Ham radio operators and it occurred to us that it would be a lot of fun, if not somewhat annoying, to write a program that would replay what we typed on the keyboard in Morse code, so each of us tackled the problem to see who would come up with the best solution. Our first attempts had well over 100 lines of code with numerous redundant statements and large numbers of if statements, one for each character. It quickly became a competition to see who could implement the solution in the fewest lines of code. Curiosity and a desire to win drove us to learn array handling, subroutines, and character encoding. Ultimately, we were able to implement solutions that used just a few lines of code. As you can imagine, the course itself was an easy A. Within a week or two, we were far beyond those students who limited their attention to the course curriculum. You learn to solve problems by solving problems. And if you don't have any problems of your own to solve, go solve someone else's. I doubt there's a single technology that does not have associated with it at least one, if not dozens, of open forums. Stockexchange. com and stockoverflow. com are two of the best known, but there are numerous other technology, industry, and product-specific sites as well. On all of them, you will find people asking questions and asking for help, usually for real world problems. It is a boundless source for inspiration. Start seeing if you can answer the question and don't look at the existing answers, that's cheating. Only look at those after you get stuck. And don't just figure out the answer to a question, but prove the answer by implementing it. If after you've done so and you see that nobody else has answered, go ahead and answer the question yourself. This is a great habit to get into. When I have spare time, I'll often hit up one of the Force. com forums and look for some unanswered questions to tackle and review answers on existing questions just to keep a sense of what kinds of issues people are running into. Another great way for software developers to learn skills, is to get involved and contribute to an open-source project. I know more than one developer who doesn't even bother with a résumé. When asked, they simply point their prospective employers to the work they have produced on GitHub. If you do decide to go to school or take a class, before doing so, be sure to verify that they have a strong focus on teaching skills and not just academics. The only time where this is less important is if your initial goal is to focus on fundamental concepts. A high school graduate studying computer science may graduate with fewer programming skills than someone who takes a hands-on programming class at a community college, but the foundation he picks up along the way will make it easier to learn those skills and more important, to adapt them across multiple languages, platforms, and technologies.

Where Do You Go for Support and Discipline?

Where do you go for support and discipline? I never liked quizzes, midterms, and final exams, but I'll say this much for them, they do focus your attention. Knowing that a test was coming up in a few days that could impact your entire future, can get you to sit down and study like nothing else. Exams do provide deadlines and discipline. Learning with others provides discipline as well. If you are part of a study group, you may feel an obligation to the rest of the group to carry your share of the load. If you have a teacher, you may feel a desire to meet his or her expectations. There are other advantages to being part of a group. You can support and encourage each other, you can answer each other's questions, you can challenge each other. There are advantages to having a teacher. A good teacher can adapt the curriculum to meet your needs. They can speed up if you're grasping the material quickly. They can slow down and repeat the content or try different approaches if you're having trouble understanding something. They can set out challenges to sharpen your skills and monitor, evaluate, and correct your work to ensure that you are truly gaining those skills. Having teachers and learning with a group can dramatically improve your learning efficiency in every learning track, whether it is fundamentals, information, skills, and even expertise. And the most common way and often the easiest way to find a teacher and a group is to take a class. It is also typically the most expensive approach. This is one area where schools maintain a strong advantage, but there are alternatives. One of the best ways to find both teachers and a group to learn with is at work. Tech companies often have knowledgeable and skilled employees and you can often find others who want to learn the same material. You may be able to recruit someone to teach something or at least lead a session. Even if you can't find a teacher, you can watch an online course together. I've heard of more than one study group that watches Pluralsight courses together and then works through the material. If you can't form a study group at work, try reaching out to the local community or form one through a social network. You can obtain some level of support online, at least when it comes to getting answers to your questions if you get stuck along the way. This is one area where online courses do have an edge over books, in that most online courses have discussion forums attached where you can ask questions or discuss issues with others who are taking the course. You may even get an answer from the course author. This is less common with books.

Where Do You Go to Gain Expertise?

Where do you go to gain expertise? Remember what I said in the last module, seeking out expertise is rarely a good use of your time. That said, it's not uncommon for people to become experts in spite of themselves. In the previous clip on skills, I described how the best way to gain skills is to seek out questions and problems and try to solve them. As you do this, you will become more and more comfortable with the technology. You will seek out and gain more and more knowledge as you search for answers to problems. You will gain more and more experience as you implement the solutions. In time, you may find yourself answering other people's questions. You many have noticed something a bit odd about my comment in the last clip about how I engage in online communities where I encouraged those learning skills to look for questions and ignore existing answers, I noted that I look for questions that don't have answers. That's because in that particular space I am an expert and the problems that interest me are the tough ones, the ones where I get to participate in coming up with new or innovative solutions or debate with others about the best approaches for solving problems or figuring out whether they can be solved at all. If you want to gain expertise, just keep seeking out and solving problems. Expertise will come in its own good time. One day you'll wake up and realize that you're answering a lot more questions than you are asking and that you can answer most questions without any real effort. Enjoy the feeling, it only lasts until the next version release.

Where Do You Go for Certification?

Where do you go to get certification? Wouldn't it be great if you could accurately put on a résumé all of the things you know and all of your skills, hand it over to a potential employer or client, and have them understand and accept it as written. That would be something. Of course, the world doesn't work that way. First, it's virtually impossible to cram everything you know and all of your skills onto a résumé, at least, not if you expect to keep it down to a page or two, and all too often the people reading the résumé don't understand it even if they believe every word. Certification is one of the most common ways to address this problem. It serves two purposes. First, it defines a domain of knowledge or competence and second, it attests that you have met the requirements of the certificate. In other words, that you have established competence within that domain. This is the intent of certification. How well a given certificate actually measures one's competence varies considerably. A simple exam-based certificate may do no more than measure your ability to memorize some facts the night before or quickly look up information while taking a test. A certificate based on peer review and documented work, may demonstrate real competence, if not, expertise in a domain. The value of a certificate can vary greatly depending on the domain it covers. A college degree generally carries a high value as it represents your ability not just to pass one test, but to pass multiple exams on multiple subjects over time and in some cases accomplish practical work as well. A certificate on a single topic may also prove valuable and it may be the deciding factor as to whether or not you get a job and strongly influence your pay. The value of a certificate also depends on who issues it. A degree from a community college does not carry the weight of a bachelor's degree from a major university. A certificate from a specific company can carry a great deal of weight. For example, networking professionals who work on Cisco equipment may work to obtain one of a number of levels of certification by taking a variety of tests. As the tests are designed and monitored by Cisco, they are highly regarded. Each certification defines a level of expertise and scope of knowledge and employers and clients can have a high degree of confidence that if you hold a particular certificate you have the level of competence defined by that certificate. Certification is often a commitment. For example, the Force. com platform that I focus on these days has major updates every four months, so if you wish to remain certified, you have to take an exam to keep your certification with each release. Certification thus serves a number of purposes beyond just proving that you know something and helping you get a better job. It imposes a certain discipline in terms of keeping your knowledge and skills up to date. It represents yet another form of curation, defining a domain of knowledge and skills. Reviewing the requirements for certification can thus be helpful in designing your own course of learning even if you never take the exam.

Whether you should or should not get certified is a choice you need to make based on your own career goals and the technology you are working in. For example, I know a number of software developers who dropped out of college or deferred college in order to work at a start-up. The start-up didn't care if they had their degree and if the start-up succeeds, they may end up wealthy enough so they can, if they wish, go to college later at their convenience and pay for it out of petty cash. I know other developers who have outstanding degrees and certificates, but have a tough time getting a job. In some cases, it's because they lack the soft skills discussed in module 2. Being able to work well with others and communicate well in an interview are skills that can get one hired even without the right certificate. lack of those skills can prevent you from getting a job even with the best ones. There are some fields where certificates exist, but nobody cares. Most of the Microsoft-certified professionals I know obtained the certificate because they were consultants and it helped them get more work and sometimes charge more. It seems less common among employees who relied more on past experience than on a certificate. It's fairly easy to find out if a certificate is important for your field. Look at the want ads. If you find that many of them are looking for specific certificates, it's a safe bet that they have value. this, again, attests to the value of a college degree, a requirement for many technical jobs. Certificates are less valuable when the economy is booming and demand for talent is high. They can be essential during down times when any way of differentiating yourself from the crowd is important. Certifications are not magic. All they represent is a form of evidence that you have a degree of competence in a particular space. I mentioned how certifications on the Force. com platform, of which there are many, have to be renewed every four months. I personally don't have any Force. com certifications. When I mentioned this to someone the other day, he was a bit surprised at first, but then I held up a copy of my book, advanced Apex Programming for Salesforce. com and Force. com, the first and so far only, advanced book on Apex programming and I told him, look, the purpose of a certificate is to provide evidence that I know something. You've read the book, what do you think, do I know something about Force. com development. He laughed and agreed. My book is my certification. Work on an open-source project can be a certification. Verifiable real-world experience and accomplishment can be certification. Now I do have degrees in computer science and electrical engineering, but aside from that, I've never taken a certification test. I have, however, helped write a few over the years and I suppose, come to think of it, that being asked to contribute to a certification test is also a form of certification. When you define your learning program for a technology, find out the role of certification in that space. If they exist, find out what they cover and incorporate that information into your plan. Find out their value in the marketplace before deciding whether it's worth your time, effort, and cost to become certified and which certifications make sense for you to pursue.

Where Do You Go for Fun?

Where do you go for fun? Personally, I like to go on a cruise where the internet is so expensive that it encourages me to disconnect completely. seriously though, there are so many amazing technologies out there, try not to focus exclusively on the ones you have to do for work. Even if you are passionate about your work, take the time, if you can, to play around with some other technologies. We have an IT guy at work who programs Philip's Hue light bulbs using Python just for fun. How cool is that?

What Do You Want to Learn Today?

What Do You Want to Learn Today?

In the last module, I described online courses such as this one, as the modern equivalent of books. Online courses and books are both good at curation and providing information. They are not as good at teaching skills or providing certification and are virtually worthless when it comes to providing support and discipline. As far as this course is concerned, I'm less concerned about support and discipline. The fact that you're watching suggests that you at least care about learning and I hope I've done a good job at providing some knowledge at least as far as helping build a mental framework, a way at looking at learning in the Information Age. My goal in this module is to help you develop some real skills when it comes to designing a learning plan for yourself. Of course, I can't address your particular situation or watch what you're doing, or offer feedback, but I can provide you directions and instructions that you should be able to adapt to your needs. The first question to ask yourself is, which track do you want to focus on and how far do you wish to go on each? This is another way of saying that you need to define your goals. If you are starting out on a new platform or domain, there's a good chance you'll want to first focus on the fundamentals track, especially if it's a domain where you intend to be spending a fair amount of time. For example, if you plan to maintain, configure, support, and customize WordPress websites, you might want a background that includes an understanding of technology, such as domain registration, DNS, Apache, IIS configuration, URL rewriting, HTML, JavaScript, PHP, and MySQL. On the information track, you would want to invest time on learning WordPress configuration, researching common plugins, then move on to the WordPress architecture and API and plugin and theme development. You'll also want to become familiar with the WordPress database schema. On the skills track, you would want to configure a WordPress site from start to finish and if not create a plugin from scratch, at least modify one. On the other hand, if you've just been called in to help with a particular WordPress site, you might skip the fundamentals entirely and use search to try to address the specific issue at hand. With luck, you might find that someone else has solved the problem and you can just follow their solution without a deep understanding of what you're doing. The better your existing understanding of the fundamental technologies discussed earlier, the better the chances that you'll be able to successfully and quickly find and deploy a solution. Implementing the solution might take some trial and error, certainly more than if you had already learned the skills, but it may not matter. It's hardly worth spending a great deal of time and effort to learn a technology that you're only going to use once or twice. Your goals should be the driving force in creating a learning plan. The rest of this module will address how to build the plan for each learning track.

Learning Fundamentals

The first challenge in learning fundamentals is that of curation. When starting on a new technology, you don't yet know the scope of what there is to learn. If you don't know what the fundamentals are, how can you choose among them? Where to find curation depends greatly on the nature of the technology you are trying to learn and how much of a beginner you are. For example, in the previous section I listed PHP and MySQL as fundamental knowledge for a WordPress developer. That is because those are domains that are broadly applicable and not specific to WordPress, whereas developing a WordPress plugin is definitely on the information track. It is very specific to WordPress. Yet, PHP and MySQL are themselves on the information track for someone who is just starting to learn programming or database technology. You can think of learning technology as a pyramid where information is built on a base of fundamentals and that information becomes the fundamentals base for the next level in the pyramid. The easiest way to start researching curation for any given technology is to seek out the certifications that include that technology. This is remarkably easy. Most community colleges publish lists of the certificates that they offer along with a course listing for each one. For example, De Anza Community College in Cupertino is considered one of the best community colleges in this area. Not surprising, as it is in the heart of Silicon Valley. Here's what their certificate list in Computer and Information Systems looks like at the time this course was published. You'll find similar lists on other college sites for computer science, computer engineering, and other engineering degrees, and for specializations within those degrees. Now, the requirements for each certificate will include both fundamental knowledge and specific information. The trick is to look for courses that appear on the requirements list for more than one certificate. Those are the ones most likely to represent fundamental knowledge. If the technology you are learning does not have a certification path, another great source of curation are books and online courses. Here again, the idea is not to go out and buy books and take courses, that comes later. Your first step is to examine the table of contents on a selection of books and courses that covers the technology you want to learn. Look for those topics that appear towards the beginning of each resource, in particular those that appear frequently. Those are the topics that are most likely to represent fundamental knowledge in the domain. Now take a break from watching this course and try it. Pick any technology that you've been curious about learning and see if you can find a school or company that provides certification in that space. Track down the programmer requirements to obtain that certification. Remember, you aren't limited to one school. On the contrary, compare the certificate requirements for programs from several different schools and list the curriculum elements that they have in common. Now look for books on the technology, find several. Amazon. com has a look-inside feature on most books that allows you to examine the table of contents easily. Use it. If there is a Pluralsight course on the topic, look at the list of modules and clips. Compare them to the contents of the books you found. It should not take you long to come up with a list of subjects. Those that appear in most of the resources are most likely to represent the fundamental concepts that you must learn. Those subjects that appear in only some of the resources and towards the end of the book or course or curriculum are more likely to reflect specific information or guides to help you develop skills. When it comes to choosing the content that you will use in your own learning plan, it's often better to use multiple resources. For example, if you wish to learn iPhone app development, don't look for the best book on iPhone development to study in depth. Instead, purchase several books that have decent reviews and go through them without worrying if you understand every concept in each one. Don't watch one Pluralsight course on a topic, watch any that are related. Seeing the same material presented several different ways increases the odds that you will find an author that presents it in a way that works for you and will help you to have a better understanding of the concepts.

Acquiring Information

The greatest challenge with gaining knowledge on the information track is timeliness. Information goes obsolete very quickly. For that reason, the internet tends to be the best source of information. It's at its best when it comes to finding answers to questions and solutions to problems. It's not as good a source for curation. It takes time for authors to figure out what information is important and what isn't and that's always a moving target as new information appears and old information becomes obsolete. Even if you can find a website that offers recommendations for learning a given technology, there's every chance that the recommendations are already obsolete. Remember to use date filters on your searches and to check the post time for any articles. On the information track, any article more than a year old should be considered suspect. If the technology you are learning has certifications available, look at sample certification tests. Many certification tests have an unfortunate tendency to focus more on memorizing facts than on measuring fundamental knowledge or skill. This is understandable as it's much easier to create an exam that tests your knowledge of facts, but this very tendency can work in your favor as the sample tests will tell you what facts, what information, the test authors felt you need to know. If you have a recent book or course on a technology, you can usually trust the curation of the author. This is a large part of the value of courses here on Pluralsight. Those focused on a specific technology will typically combine fundamental knowledge with up-to-date information in a way that is curated by an expert in the technology. Look in the same books and courses you looked at when seeking out topics for the fundamentals path. This time, look for subjects that are very specific to the technology and for those that don't appear in every resource. Where you might want to develop a fairly robust subject list for your learning plan in the fundamental track, you should not go into too much detail or worry about how complete your list is for the information track. Instead, allow the information you learn to be driven by your efforts to gain skills.

Gaining Skills

Your first step in the skills track is simple. You must set up a full development or test environment. If you are studying network engineering, buy or borrow access to the necessary equipment. If you are studying database administration, install the database and management tools. If you are studying software development, set up a development environment. There is no substitute for this. If the courses you are taking or books you are reading include tutorials, consider following along, but remember that following the tutorials is not enough. If you see something that you don't understand, stop and do the research needed to understand the topic or at the very least, return to that research after you've completed the example. As you are doing the examples, think of variations to try. Experiment, play around, that's where you'll really be learning skills. Following the examples only serves to create some comfort and familiarity with the technology. The examples themselves do not build skills so much as they provide a form of curation, pointing out the skills you need to develop. If you do not go beyond the tutorial and do not research the concepts illustrated in the tutorial, you will end up knowing how to build the solution demonstrated in the tutorial and lack any ability to go beyond that or implement variations of the problem that was illustrated. This happens all the time. Personally, I almost never bother with tutorials beyond those that illustrate how to set up the initial test or development environment. I find the other approaches described later in this clip to be much more efficient. To gain skills and competence, you must use the technology. The most common question I hear is, what should I do? If you're taking a hands-on class or workshop, what you do will be dictated by the projects or homework you are given, but where do you go if you're not taking a formal class? The best solution is to seek out public forums relating to the technology where people ask questions on how to solve problems. Find out where people are asking questions on the technology, but don't look at the answer. Instead, try to solve the problem. Don't answer the question, build out the answer to the question. If someone posted sample code that isn't working, fix it. They say that the best way to learn something is to teach it, so if you do find solutions to problems and nobody has posted a better answer, go ahead and post an answer. As people recognize your ability, you'll find yourself getting more questions, which will help you gain even more skills. Solving real-world problems is the only way to gain real-world experience. To solve the problems, you'll need to research information and you may need to go back and learn fundamental concepts. At a certain point, if time and your job permits, consider doing some work on one of the many sites that match up consultants with clients, such as Elance. com or oDesk. com. Tackling a few smaller, real-world projects can really sharpen your skills. Another approach is to study and then even contribute to an open-source project.

Becoming an Expert

In many cases, becoming an expert is something that happens to you as compared to something that you plan, but if your goal is really to become an acknowledged expert in a technology, it's really not that hard. First, you'll need to keep building skills as described in the previous clip. Answer questions on forums, contribute to open-source projects, blog, write, present at users groups, attend and speak at conferences, write a book, produce a course for Pluralsight. The more visible you are, the more opportunities will come your way. Just remember, being an expert in a subject is a lot of work that doesn't necessarily pay any better than just being competent in a technology, and to take full advantage of your expertise will require gaining soft skills as well.

Executing on Your Learning Plan and Conclusion

Now you know how to build a learning plan that includes three and possibly four tracks. I presented these tracks in a rather traditional order. One could argue that it is perfectly logical to start with the fundamentals, then build on them to learn specific information, and then try them out by finding ways to practice those skills. It may be logical, but it's wrong, absolutely wrong. When you start executing on the plan, I strongly encourage you to mix things up. As soon as you grasp a fundamental concept, seek out how it can be used and do it. Remember how you learned as a kid. It was much more fun to dive in and start doing things than to sit through long boring lectures that suck all the excitement out of what you're learning, so that by the time you finally have a chance to try it out you'd rather just go home and play video games. Remember, the skills track is the one that pays the bills. It's the one that's most fun, so it should be the one that drives the agenda. Try something, do something. If you fail, look for the relevant information you need to succeed. Then, if there's anything about what you did that you did not thoroughly understand, take the time to learn the fundamental concepts behind what you did and don't stop until you've found them. Your learning plan is not a set of ordered lists, it's what programmers call a graph, a set of nodes that are interconnected in an arbitrary manner. The path you take isn't important and it's okay to revisit a node or topic multiple times until you understand it. The purpose of a plan is not to strictly define the order in which you learn things, though having an order can certainly help you identify which concepts are prerequisites to others. The main purpose of the plan is to define the scope, to make sure that you do learn everything you need to learn in order to meet your goals. One point that I make in all my courses about my own background is that while I have done many things in my career, the one consistency is that I've been a software developer the entire time. I write code and I ship it and people pay for my work and products. Now think for a minute what that really means, given that the first production code I shipped was about 35 years ago as of the time this course was published. The number of platforms, systems, and languages I've worked in is positively scary and I'm still doing it. Even today as I finish this course, there are several new technologies that I'm dying to learn and will learn as soon as I have some extra time on my hands. And that's the thing, time is the real cost of learning and we don't have time to waste. If this course helps you to become even slightly more efficient learning technology going forward, then it will ultimately be worth the hour or so you spent watching. That is my hope and I wish you all success in your journey towards skills and knowledge.

Course author

Dan Appleman

Dan Appleman is a well known author, software developer, and speaker. Currently the CTO of Full Circle Insights, he is the author of numerous books, ebooks, and online courses on various topics...

Course info

Level	Beginner
Rating	★★★★☆ (1614)
My rating	★★★★★
Duration	1h 35m
Released	1 Jan 2014

Share course