Innovation—Create the Primordial Ooze

On a sunny spring day in March 2010, Eric sat in his car at the intersection of Embarcadero Avenue and El Camino Real in Palo Alto, looking at the trees surrounding Stanford's football stadium and reflecting on the past couple of hours and waiting for the light to change. He had just come from having coffee with Apple CEO Steve Jobs at a restaurant called Calafia.[153] The two had sat outside at the California cuisine–oriented café, discussing Google's growing mobile operating system, Android. Steve was convinced that the open-source operating system was built on intellectual property created by Apple. Eric responded that we hadn't used Apple's IP and had in fact built Android on our own. But his argument was to no avail. "They are going to fight us," he thought.

Eric first met and started working with Steve Jobs in 1993, when Eric was at Sun and Steve was at NeXT Computer. NeXT was built using a computer language called Objective-C, and Eric and a few others traveled to NeXT's offices to hear Steve give them an update. Steve started extolling the virtues of Objective-C and tried to convince the computer scientists from Sun that they needed to use it in the next-generation programming framework they were developing. Eric knew that Steve was wrong on some of his technical points, but Steve's arguments were so persuasive that Eric and his Sun colleagues couldn't figure out exactly *how* he was wrong. They stood by their car in the NeXT parking lot after the meeting, dissecting the discussion and trying to escape what was often called the Jobs "reality distortion field." No such luck. Steve saw them in the parking lot and rushed out to continue the conversation. For another hour.

Steve and Eric built a friendship over the years, and in the summer of 2006 Eric was invited to join the Apple board. Before he accepted the job, he and Steve had a conversation about the potential conflicts of interest between Apple and Google. Apple was working on the iPhone, and Google had been working on a mobile operating system for about a year, having purchased Andy Rubin's company, Android, in August 2005. The definition of what Android would produce was still in flux, but at the time it looked like it would be an open-source operating system without a user interface (other companies would build the UI). Our hope was that Android could become the software guts for phones built by companies like Motorola, Nokia, or Samsung, who would build their own designs and applications. Both Android and the iPhone were in their early stages, and Eric joined the Apple board in August.

The Apple iPhone launched in June 2007, and it was nearly perfect. It was designed and optimized to be connected to the Internet, and it worked with a level of seamless convenience that was just great. Later that year, the Android team posted a YouTube video that unveiled what they had been working on. Steve scrutinized the video and concluded that the user experience it portrayed was too much like the iPhone's and its underlying operating system, iOS. Eric ultimately stepped down from the Apple board in August 2009, and the legal skirmishes between the companies and their partners continue today.

Eventually, everyone who wants to will have some sort of smartphone, and most of those phones (at least for the next decade) will run on either iOS or Android. More engineers are developing apps for these platforms than for any other computing platform in history (it helps to have a potential user base of just about everyone on the planet). Dozens of manufacturers are competing, and their ups and downs aren't just chronicled in trade magazines, but in

mainstream media too. This is obvious to us now, but it was only a few years ago that personal computing meant a PC running Microsoft Windows. Regardless of whose side you are on in the patent arguments (to be clear: We are on our side), there is no disputing the fact that these two platforms have spurred enormous economic development and innovation and have changed people's lives for the better around the globe.

Steve Jobs saw this future with great clarity. There is no better example of the impact a smart creative can have on the world than him. He embodied a combination of technical depth, artistic and creative talent, and business savvy that allowed him to create computing products with which people actually fell in love. He merged beauty and science in a tech community that had a lot of nerds and business people, but very few artists. The two of us learned a lot about smart creatives from working with and observing Steve, about how much personal style can influence company culture and about how that culture is directly tied to success.

Yet the most important aspect of the Android/iOS saga is how it demonstrates two *different* ways to achieve innovation. Both platforms—and companies—are tremendously innovative, and there are a few key similarities in our approaches. Apple and Google both operate in industries with extraordinarily fast product life cycles. In both the Internet and mobile computing businesses, the next big thing becomes old hat pretty quickly, so we both need to innovate constantly or fall behind. Both companies tend to eschew traditional market research and rely on our own abilities to figure out what consumers will want. We trust our vision. And both companies place the highest priority on creating the best experience for our consumers.

But beyond these similarities, our approaches to innovation are quite different, primarily when it comes to control. With Android, Google bet on the superior economics of an open platform and on our ability to navigate the fragmentation that results from such openness. Android is—and we mean this in the most positive way—out of control. The software source code is available to anyone to use for free, under the Apache license agreement.[154] This "open source" model means that anyone can take the operating system and do what they want with it; Android is the sand in the sandbox.

We also let anyone create and sell applications that run on Android-based devices; they don't require Google's approval. We encourage makers of Android devices (such as Samsung, HTC, and Motorola) to build those devices so that they are compatible at the application level, so that all Android apps run well on all Android devices. We are highly, but not 100 percent, effective in achieving that goal, which is to be expected in an environment where there is no cost to entry and no controlling power. Android is growing and expanding in ways that we could have never predicted. It is used to power electronic readers (including Amazon's), tablets, game players, and phones. It also can be found in refrigerators, treadmills, TVs, and toys. Many pundits talk about the coming "Internet of things," when all sorts of devices—not just tablets and phones—will be connected to the Internet. We hope that many of those things will run on Android.

Apple represents the opposite approach. iOS code is closed, and applications that want to be on the App Store must receive Apple's formal approval. Steve always believed that the best experience for the consumer comes from maintaining complete control of everything. He paid tremendous attention to the details in everything he and his company did, with a singular purpose of creating the best possible products. This showed in his extraordinary product presentations to his board, which were always highly orchestrated and produced as if they were Broadway shows. New products weren't just demonstrated at board meetings, they were unveiled. At Google we regularly had APMs demo new products to our board, not just

because they did it so well (though not as well as Steve!) but also for the sheer theatricality of having kids a year or two out of college (sometimes wearing their alma mater's sweatshirt) show board members our newest innovations. We learned the value of that type of showmanship from Steve.

Apple's control model works not just because of Steve Jobs's excellence, but also because of how he organized the company. At Apple—just like Google—the leaders are product people with technical backgrounds. When you build a team of great smart creatives, and put the world's uber-smart creative in charge, then you have a good chance of being right most of the time. And when you are right most of the time, then a highly controlled model can yield tremendous innovation.

As we were writing this chapter on innovation and sharing it with friends and family, whenever someone took issue with our principles they would often use Steve Jobs as the counterexample. "Yes, but Steve Jobs didn't do it that way, and look how many great things he created." They were right of course, so we generally responded by saying that if they were the equal of Steve Jobs, with instinct and insights matched by few people, then they should go ahead and Android team posted a YouTube video that unveiled what they had been working on. Steve scrutinized the video and concluded that the user experience it portrayed was too much like the iPhone's and its underlying operating system, iOS. Eric ultimately stepped down from the Apple board in August 2009, and the legal skirmishes between the companies and their partners continue today.

Eventually, everyone who wants to will have some sort of smartphone, and most of those phones (at least for the next decade) will run on either iOS or Android. More engineers are developing apps for these platforms than for any other computing platform in history (it helps to have a potential user base of just about everyone on the planet). Dozens of manufacturers are competing, and their ups and downs aren't just chronicled in trade magazines, but in mainstream media too. This is obvious to us now, but it was only a few years ago that personal computing meant a PC running Microsoft Windows. Regardless of whose side you are on in the patent arguments (to be clear: We are on our side), there is no disputing the fact that these two platforms have spurred enormous economic development and innovation and have changed people's lives for the better around the globe.

Steve Jobs saw this future with great clarity. There is no better example of the impact a smart creative can have on the world than him. He embodied a combination of technical depth, artistic and creative talent, and business savvy that allowed him to create computing products with which people actually fell in love. He merged beauty and science in a tech community that had a lot of nerds and business people, but very few artists. The two of us learned a lot about smart creatives from working with and observing Steve, about how much personal style can influence company culture and about how that culture is directly tied to success.

Yet the most important aspect of the Android/iOS saga is how it demonstrates two *different* ways to achieve innovation. Both platforms—and companies—are tremendously innovative, and there are a few key similarities in our approaches. Apple and Google both operate in industries with extraordinarily fast product life cycles. In both the Internet and mobile computing businesses, the next big thing becomes old hat pretty quickly, so we both need to innovate constantly or fall behind. Both companies tend to eschew traditional market research and rely on our own abilities to figure out what consumers will want. We trust our vision. And both companies place the highest priority on creating the best experience for our consumers.

But beyond these similarities, our approaches to innovation are quite different, primarily when it comes to control. With Android, Google bet on the superior economics of an open platform and on our ability to navigate the fragmentation that results from such openness. Android is—and we mean this in the most positive way—out of control. The software source code is available to anyone to use for free, under the Apache license agreement.[154] This "open source" model means that anyone can take the operating system and do what they want with it; Android is the sand in the sandbox.

We also let anyone create and sell applications that run on Android-based devices; they don't require Google's approval. We encourage makers of Android devices (such as Samsung, HTC, and Motorola) to build those devices so that they are compatible at the application level, so that all Android apps run well on all Android devices. We are highly, but not 100 percent, effective in achieving that goal, which is to be expected in an environment where there is no cost to entry and no controlling power. Android is growing and expanding in ways that we could have never predicted. It is used to power electronic readers (including Amazon's), tablets, game players, and phones. It also can be found in refrigerators, treadmills, TVs, and toys. Many pundits talk about the coming "Internet of things," when all sorts of devices—not just tablets and phones—will be connected to the Internet. We hope that many of those things will run on Android.

Apple represents the opposite approach. iOS code is closed, and applications that want to be on the App Store must receive Apple's formal approval. Steve always believed that the best experience for the consumer comes from maintaining complete control of everything. He paid tremendous attention to the details in everything he and his company did, with a singular purpose of creating the best possible products. This showed in his extraordinary product presentations to his board, which were always highly orchestrated and produced as if they were Broadway shows. New products weren't just demonstrated at board meetings, they were unveiled. At Google we regularly had APMs demo new products to our board, not just because they did it so well (though not as well as Steve!) but also for the sheer theatricality of having kids a year or two out of college (sometimes wearing their alma mater's sweatshirt) show board members our newest innovations. We learned the value of that type of showmanship from Steve.

Apple's control model works not just because of Steve Jobs's excellence, but also because of how he organized the company. At Apple—just like Google—the leaders are product people with technical backgrounds. When you build a team of great smart creatives, and put the world's uber-smart creative in charge, then you have a good chance of being right most of the time. And when you are right most of the time, then a highly controlled model can yield tremendous innovation.

As we were writing this chapter on innovation and sharing it with friends and family, whenever someone took issue with our principles they would often use Steve Jobs as the counterexample. "Yes, but Steve Jobs didn't do it that way, and look how many great things he created." They were right of course, so we generally responded by saying that if they were the equal of Steve Jobs, with instinct and insights matched by few people, then they should go ahead and try it his way. But if you are like most of the rest of us, then when it comes to innovation we have some other ideas you might want to try out.

Android team posted a YouTube video that unveiled what they had been working on. Steve scrutinized the video and concluded that the user experience it portrayed was too much like the iPhone's and its underlying operating system, iOS. Eric ultimately stepped down from the Apple board in August 2009, and the legal skirmishes between the companies and their partners continue today.

Eventually, everyone who wants to will have some sort of smartphone, and most of those phones (at least for the next decade) will run on either iOS or Android. More engineers are developing apps for these platforms than for any other computing platform in history (it helps to have a potential user base of just about everyone on the planet). Dozens of manufacturers are competing, and their ups and downs aren't just chronicled in trade magazines, but in mainstream media too. This is obvious to us now, but it was only a few years ago that personal computing meant a PC running Microsoft Windows. Regardless of whose side you are on in the patent arguments (to be clear: We are on our side), there is no disputing the fact that these two platforms have spurred enormous economic development and innovation and have changed people's lives for the better around the globe.

Steve Jobs saw this future with great clarity. There is no better example of the impact a smart creative can have on the world than him. He embodied a combination of technical depth, artistic and creative talent, and business savvy that allowed him to create computing products with which people actually fell in love. He merged beauty and science in a tech community that had a lot of nerds and business people, but very few artists. The two of us learned a lot about smart creatives from working with and observing Steve, about how much personal style can influence company culture and about how that culture is directly tied to success.

Yet the most important aspect of the Android/iOS saga is how it demonstrates two *different* ways to achieve innovation. Both platforms—and companies—are tremendously innovative, and there are a few key similarities in our approaches. Apple and Google both operate in industries with extraordinarily fast product life cycles. In both the Internet and mobile computing businesses, the next big thing becomes old hat pretty quickly, so we both need to innovate constantly or fall behind. Both companies tend to eschew traditional market research and rely on our own abilities to figure out what consumers will want. We trust our vision. And both companies place the highest priority on creating the best experience for our consumers.

But beyond these similarities, our approaches to innovation are quite different, primarily when it comes to control. With Android, Google bet on the superior economics of an open platform and on our ability to navigate the fragmentation that results from such openness. Android is—and we mean this in the most positive way—out of control. The software source code is available to anyone to use for free, under the Apache license agreement.[154] This "open source" model means that anyone can take the operating system and do what they want with it; Android is the sand in the sandbox.

We also let anyone create and sell applications that run on Android-based devices; they don't require Google's approval. We encourage makers of Android devices (such as Samsung, HTC, and Motorola) to build those devices so that they are compatible at the application level, so that all Android apps run well on all Android devices. We are highly, but not 100 percent, effective in achieving that goal, which is to be expected in an environment where there is no cost to entry and no controlling power. Android is growing and expanding in ways that we could have never predicted. It is used to power electronic readers (including Amazon's), tablets, game players, and phones. It also can be found in refrigerators, treadmills, TVs, and toys. Many pundits talk about the coming "Internet of things," when all sorts of devices—not just tablets and phones—will be connected to the Internet. We hope that many of those things will run on Android.

Apple represents the opposite approach. iOS code is closed, and applications that want to be on the App Store must receive Apple's formal approval. Steve always believed that the best experience for the consumer comes from maintaining complete control of everything. He

paid tremendous attention to the details in everything he and his company did, with a singular purpose of creating the best possible products. This showed in his extraordinary product presentations to his board, which were always highly orchestrated and produced as if they were Broadway shows. New products weren't just demonstrated at board meetings, they were unveiled. At Google we regularly had APMs demo new products to our board, not just because they did it so well (though not as well as Steve!) but also for the sheer theatricality of having kids a year or two out of college (sometimes wearing their alma mater's sweatshirt) show board members our newest innovations. We learned the value of that type of showmanship from Steve.

Apple's control model works not just because of Steve Jobs's excellence, but also because of how he organized the company. At Apple—just like Google—the leaders are product people with technical backgrounds. When you build a team of great smart creatives, and put the world's uber-smart creative in charge, then you have a good chance of being right most of the time. And when you are right most of the time, then a highly controlled model can yield tremendous innovation.

As we were writing this chapter on innovation and sharing it with friends and family, whenever someone took issue with our principles they would often use Steve Jobs as the counterexample. "Yes, but Steve Jobs didn't do it that way, and look how many great things he created." They were right of course, so we generally responded by saying that if they were the equal of Steve Jobs, with instinct and insights matched by few people, then they should go ahead and try it his way. But if you are like most of the rest of us, then when it comes to innovation we have some other ideas you might want to try out.

Android team posted a YouTube video that unveiled what they had been working on. Steve scrutinized the video and concluded that the user experience it portrayed was too much like the iPhone's and its underlying operating system, iOS. Eric ultimately stepped down from the Apple board in August 2009, and the legal skirmishes between the companies and their partners continue today.

Eventually, everyone who wants to will have some sort of smartphone, and most of those phones (at least for the next decade) will run on either iOS or Android. More engineers are developing apps for these platforms than for any other computing platform in history (it helps to have a potential user base of just about everyone on the planet). Dozens of manufacturers are competing, and their ups and downs aren't just chronicled in trade magazines, but in mainstream media too. This is obvious to us now, but it was only a few years ago that personal computing meant a PC running Microsoft Windows. Regardless of whose side you are on in the patent arguments (to be clear: We are on our side), there is no disputing the fact that these two platforms have spurred enormous economic development and innovation and have changed people's lives for the better around the globe.

Steve Jobs saw this future with great clarity. There is no better example of the impact a smart creative can have on the world than him. He embodied a combination of technical depth, artistic and creative talent, and business savvy that allowed him to create computing products with which people actually fell in love. He merged beauty and science in a tech community that had a lot of nerds and business people, but very few artists. The two of us learned a lot about smart creatives from working with and observing Steve, about how much personal style can influence company culture and about how that culture is directly tied to success.

Yet the most important aspect of the Android/iOS saga is how it demonstrates two *different* ways to achieve innovation. Both platforms—and companies—are tremendously innovative, and there are a few key similarities in our approaches. Apple and Google both

operate in industries with extraordinarily fast product life cycles. In both the Internet and mobile computing businesses, the next big thing becomes old hat pretty quickly, so we both need to innovate constantly or fall behind. Both companies tend to eschew traditional market research and rely on our own abilities to figure out what consumers will want. We trust our vision. And both companies place the highest priority on creating the best experience for our consumers.

But beyond these similarities, our approaches to innovation are quite different, primarily when it comes to control. With Android, Google bet on the superior economics of an open platform and on our ability to navigate the fragmentation that results from such openness. Android is—and we mean this in the most positive way—out of control. The software source code is available to anyone to use for free, under the Apache license agreement.[154] This "open source" model means that anyone can take the operating system and do what they want with it; Android is the sand in the sandbox.

We also let anyone create and sell applications that run on Android-based devices; they don't require Google's approval. We encourage makers of Android devices (such as Samsung, HTC, and Motorola) to build those devices so that they are compatible at the application level, so that all Android apps run well on all Android devices. We are highly, but not 100 percent, effective in achieving that goal, which is to be expected in an environment where there is no cost to entry and no controlling power. Android is growing and expanding in ways that we could have never predicted. It is used to power electronic readers (including Amazon's), tablets, game players, and phones. It also can be found in refrigerators, treadmills, TVs, and toys. Many pundits talk about the coming "Internet of things," when all sorts of devices—not just tablets and phones—will be connected to the Internet. We hope that many of those things will run on Android.

Apple represents the opposite approach. iOS code is closed, and applications that want to be on the App Store must receive Apple's formal approval. Steve always believed that the best experience for the consumer comes from maintaining complete control of everything. He paid tremendous attention to the details in everything he and his company did, with a singular purpose of creating the best possible products. This showed in his extraordinary product presentations to his board, which were always highly orchestrated and produced as if they were Broadway shows. New products weren't just demonstrated at board meetings, they were unveiled. At Google we regularly had APMs demo new products to our board, not just because they did it so well (though not as well as Steve!) but also for the sheer theatricality of having kids a year or two out of college (sometimes wearing their alma mater's sweatshirt) show board members our newest innovations. We learned the value of that type of showmanship from Steve.

Apple's control model works not just because of Steve Jobs's excellence, but also because of how he organized the company. At Apple—just like Google—the leaders are product people with technical backgrounds. When you build a team of great smart creatives, and put the world's uber-smart creative in charge, then you have a good chance of being right most of the time. And when you are right most of the time, then a highly controlled model can yield tremendous innovation.

As we were writing this chapter on innovation and sharing it with friends and family, whenever someone took issue with our principles they would often use Steve Jobs as the counterexample. "Yes, but Steve Jobs didn't do it that way, and look how many great things he created." They were right of course, so we generally responded by saying that if they were the equal of Steve Jobs, with instinct and insights matched by few people, then they should

go ahead and try it his way. But if you are like most of the rest of us, then when it comes to innovation we have some other ideas you might want to try out.

Dozens of people rush, creating a mosh pit of dancing fools where once there had been only one. Derek calls this the "first follower" principle: When creating a movement, attracting the first follower is the most crucial step. "The first follower is what transforms a lone nut into a leader." The primordial ooze of innovation needs to encourage the people who want to be innovative—the lone dancing fool on the side of the hill—to do their thing. But just as important, it also needs to encourage the people who want to join something that is innovative—dancing fools two through two hundred—to do their thing as well. This is why innovation needs to be integrated into the fabric of the company, across every function and region. When you isolate it under a particular group, you may attract innovators to that group, but you won't have enough first followers.

Robert Noyce, cofounder of Fairchild Semiconductor and Intel, said, "Optimism is an essential ingredient for innovation. How else can the individual welcome change over security, adventure over staying in a safe place?"[162] Hire people who are smart enough to come up with new ideas and crazy enough to think they just might work. You need to find and attract those optimistic people, then give them the place to create change and adventure.

Focus on the user…

In late 2009, a team of search engineers showed us a prototype of a feature that had been percolating for a while. It was based on a simple idea: What if we started to populate search results while the user was typing the query, rather than waiting for her to hit the return key? We had always believed that speed was one of the core factors in determining the quality of search and were proud that we answered the vast majority of searches in under a tenth of a second. But that clock didn't start ticking until the user actually entered the query. Typing the query could still take several seconds. What if we didn't have to wait? What if we showed results as the user typed? Once we saw the prototype, the go decision was a no-brainer. Both the organic and paid search teams got to work, and several months later the feature launched to the world as Google Instant.

A few weeks before the launch Jonathan was in his staff meeting when a basic question occurred to him: Would Instant affect revenue at all? Perhaps, when results populated as the user typed, the user might be less likely to click on ads, so he asked his team if we had enough good data about potential revenue impact. No, was the answer, and everyone agreed that someone should investigate it. Then they all proceeded with planning the launch. In every other company where Jonathan had worked, a financial analysis was one of the most important hoops that a product needed to jump through before getting approved. How much revenue will the product make? What will the return on investment (ROI) be? The payback period? Yet here we were, a few weeks away from launching a major change to our core product, and no one had performed a detailed financial analysis. The product was obviously great for the user, so we all knew that launching it was the best business decision.

When Instant launched, it had only a modest effect on revenue, but Google has launched plenty of other features with a more significant financial impact. Knowledge Graph, which launched in 2012, populates the right-hand side of the web results page for queries about people, places, and things with a concisely formatted panel of algorithmically curated information about that entity. It pulls all the most relevant facts together into one, easy-to-read box. For most queries, the panel replaces the ads that previously appeared on that part of the page. That one hurt revenue a little bit. In early 2011, we made a series of changes to our

search algorithms that reduced the quality scores of certain types of websites. We didn't want users clicking on links only to find themselves on crummy sites. That release, called "panda," affected nearly 12 percent of queries, and because many of the affected sites were part of our ad network, the change hurt Google's revenue.

Google knows that in the Internet Century user trust is just as important as dollars, euros, pounds, yen, or any other currency. Product excellence is the only way for a company to be consistently successful, so our prime directive when it comes to product strategy is to focus on the user (while not interfering with the internal development of alien civilizations). As Larry and Sergey put it in their IPO letter, "Serving our end users is at the heart of what we do and remains our number one priority."

But focus on the user is only half the story. The full sentence should read "focus on the user and all else will follow." This means that we will always do what's right for the user, and we trust that our smart creatives will figure out how to make money from it. It could take a while, so sticking it out requires a lot of confidence.[163] But it is usually worth it.

In 2004, Jonathan and fellow Googler Jeff Huber took Sergey on a field trip to meet with a small start-up called Keyhole. Jeff, who had been the first product manager Jonathan hired at Excite@Home, was one of the lead engineers on the ads team.[164] One of Keyhole's cofounders was Brian McClendon, with whom Jonathan and Jeff had worked at Excite@Home. Keyhole had developed some extremely cool ways to visualize and interact with maps, and Sergey immediately decided that Google should buy the company.

A few weeks later Sergey brought the acquisition to the board for its approval, and when they asked him how we would ever make money from the technology, his response was simple: "I'll let Jonathan answer that question; he's the business guy." Jonathan had been leaning back, enjoying Sergey's presentation, and had given exactly zero thought to how buying Keyhole might make Google money (clear evidence that after two years, Jonathan was now guzzling from the Google Kool-Aid dispenser). Jonathan plowed forward with some muddled answer that was memorable only in how forgettable it was. The truth was that we had no idea how buying Keyhole would benefit Google's bottom line.

The board decided to trust Sergey's judgment and let him proceed with the deal. About eight months later, Google Earth, which was based on Keyhole technology, launched. It was an immediate hit with users, and also generated millions of dollars. How could that be, since there were no ads on the app and it was free? Not long after we launched it, one of our smart creatives, Sundar Pichai, realized that all those people who were downloading and installing Google Earth might be interested in Google Toolbar as well. Toolbar was a simple utility that integrated with the browser. It had a lot of interesting features for users, one of which was a little Google search box that constantly resided in the browser's interface. People with Toolbar could initiate a Google search without going to Google.com, so they tended to conduct more searches, click on more ads, and generate more revenue. Sundar's idea met some resistance, but, with a push from Urs Hölzle, it was quickly implemented.

This simple insight—that people downloading Earth might be interested in getting Toolbar as well—increased Toolbar's user base significantly and generated lots of revenue, but there's practically no way Jonathan could have anticipated it when he stood before the board that day. Looking back, we realize the correct answer would have been, "I have no idea… but I'm sure we'll figure it out."

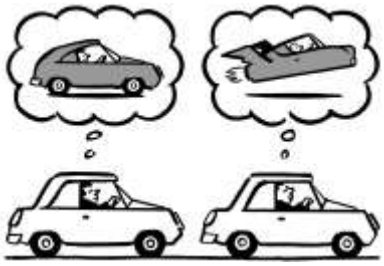
Focus on the user… and the money will follow. This can be particularly challenging in environments where the user and customer are different, and when your customer doesn't share your focus-on-the-user ethos. When Google acquired Motorola in 2012, one of the first Motorola meetings Jonathan attended was a three-hour product review, where the company's

managers presented the features and specifications for all of Motorola's phones. They kept referring to the customer requirements, most of which made little sense to Jonathan since they were so out of tune with what he knew mobile users wanted. Then, over lunch, one of the execs explained to him that when Motorola said "customers," they weren't talking about the people who use the phones but about the company's real customers, the mobile carriers such as Verizon and AT&T, who perhaps weren't always as focused on the user as they should have been. Motorola wasn't focusing on its users at all, but on its partners.

At Google, our users are the people who use our products, while our customers are the companies that buy our advertising and license our technology. There are rarely conflicts between the two, but when there are, our bias is toward the user. It has to be this way, regardless of your industry. Users are more empowered than ever, and won't tolerate crummy products.

Think big

Our colleague Vint Cerf has been working on a new suite of network protocols that will work in the harsh and very, very large environment of space. According to Vint, he started this project after asking himself what he could work on that would be needed twenty-five years later.[165] His answer: the interplanetary Internet. No one can accuse Vint of not thinking big enough.



For the rest of us humans, though, it can be a different story. Perhaps it's human nature, or just corporate nature, but most people tend to think incrementally rather than transformationally or galactically.[166] Our colleague Regina Dugan, who was the director of the Defense Advanced Research Projects Agency (DARPA)[167] before coming to Motorola and then Google, talks about how innovation often happens when people are working in "Pasteur's Quadrant,"[168] which is where they try to advance basic science while solving real-world problems. But most companies end up in the opposite quadrant, "where the science is not interesting and no one cares about the goals being pursued. Talent exits, and projects fail more often, not less."[169]

That's why one of Eric and Larry's challenges to engineers and product managers in Google product reviews was always "you aren't thinking big enough." In the Internet Century, with infinite information, reach, and computing power, global scale is available to just about everyone. But too many people are stuck in the old, limited mindset. "You aren't thinking big enough"—later replaced by the Larry Page directive to "think 10X"—helps fix that. It encompasses the art of the possible… and the impossible.

The obvious benefit of thinking big is that it gives smart creatives much more freedom. It removes constraints and spurs creativity. Astro Teller, the head of Google[x], notes that if you want to create a car that gets 10 percent better mileage, you just have to tweak the current design, but if you want to get one that gets five hundred miles per gallon, you need to start

over. Just the thought process—How would I start over?—can spur ideas that were previously not considered.

There are other, more subtle benefits to thinking big. Big bets often have a greater chance for success by virtue of their size: The company can't afford to fail. On the other hand, when you make a bunch of smaller bets, none of which are life threatening, you can end up with mediocrity. We see this all the time in business: the company with a large line of not-great products. When Google bought Motorola and Jonathan started helping new CEO Dennis Woodside, he discovered that the company had dozens of different phones, with each device targeted at a particular, market-research-defined segment, such as millennials, Gen X, boomers, and soccer moms. There was some logic behind this—different carriers wanted their own unique models—but it led to a sprawl of mediocrity. Each phone had its own set of product people who would work hard to make their product great, but they also knew that if their product was only good, the company would survive. (Dennis largely fixed these problems and created a much more user-focused Motorola before Google sold the company to Lenovo in 2014.)

On the other hand, the iPhone is a popular product precisely because it is the only phone Apple makes. If there's a problem in the development of the next-generation iPhone, no one on that team goes home until there's a plan to fix it. It's no coincidence that Apple has only a few product lines. None of them can afford to fail.

It can also be easier to take on big problems because bigger challenges attract big talent. There is a symbiotic relationship between big challenges and highly smart, skilled people: The challenges get solved and the people get happy. Give the wrong people a big challenge, and you'll induce anxiety. But give it to the right people, and you'll induce joy.[170] They enjoy rising to the challenge for the sake of it, but also, as sociologist and management guru Rosabeth Moss Kanter has pointed out, for the very real benefits it can bring: new skills, new connections with colleagues in the field, enhanced reputation—what economists would describe as investing in their human capital.[171] For these reasons, thinking big is actually a very powerful tool for attracting and retaining smart creatives. Put it this way: Let's assume you are a brilliant smart creative just graduating from college. You have two competing job offers, which are virtually identical except for one difference. One of the companies tells you that they like to try to make things 10X better, while the other settles for 10 percent improvement. Which one will you choose?

Our friend Mike Cassidy is a great example of the power of 10X thinking in retaining smart creatives. A cofounder of a company called Ruba, Mike joined Google in 2010 after we acquired Ruba's intellectual property and hired its team. Mike is a serial entrepreneur—Ruba was his fourth company; his second one was a search engine called Direct Hit that briefly competed with Google before it was sold to Ask Jeeves. So we figured that it was only a matter of time before Mike would leave the Google mother ship to start something new. Over time we lost track of what Mike was working on, but we would occasionally see him around campus so we knew he was still a Googler. Then, in June 2013, Google announced Project Loon, the aforementioned Google[x] project that aims to use helium balloons to bring broadband Internet access to the five billion people who don't yet have it. We soon learned that Mike, who has a degree in aerospace engineering, was one of the Loon project leads and had been working on it for over a year. If it weren't for that opportunity to do something audacious, he likely would have left Google, so by continuing to think big and push the boundaries of technology we got to hang on to at least one great smart creative.

And doing big things that matter, as Larry often puts it, inspires people even if they aren't the Mike Cassidys who get to be directly involved in them. We often hear people around

Google talk about "10Xing" their job, even though most of their jobs are far removed from the audacious projects the company has become famous for. These are salespeople, lawyers, financial folks, all of whom are inspired by the company's prevalent attitude to shoot for the moon. Thinking big is not only a very powerful recruiting and retention tool, it's contagious.

Set (almost) unattainable goals

The well-worn corporate manager has mastered many skills, and high on that list is the setting of annual and quarterly goals. This requires a certain finesse. Set the objectives too low and you are obviously trying to make yourself look good by "miraculously" exceeding them at the end of the quarter.[172] But set them too high and you run the risk of failure. The trick is to find the sweet spot by creating objectives that look difficult but are actually easily doable. The perfect scorecard at the end of the quarter and year is one that is full of 100 percent marks.

In late 1999, John Doerr gave a presentation at Google that changed the company, because it created a simple tool that let the founders institutionalize their "think big" ethos. John sat on our board, and his firm, Kleiner Perkins, had recently invested in the company. The topic was a form of management by objectives called OKRs (to which we referred in the previous chapter), which John had learned from former Intel CEO Andy Grove.[173] There are several characteristics that set OKRs apart from their typical under promise-and-over deliver corporate-objective brethren.

First, a good OKR marries the big-picture objective with a highly measurable key result. It's easy to set some amorphous strategic goal (make usability better… improve team morale… get in better shape) as an objective and then, at quarter end, declare victory. But when the strategic goal is measured against a concrete goal (increase usage of features by X percent… raise employee satisfaction scores by Y percent… run a half marathon in under two hours), then things get interesting. For example, one of our platform team's recent OKRs was to have "new WW systems serving significant traffic for XX large services with latency < YY microseconds @ ZZ% on Jupiter."[174] (Jupiter is a code name, not the location of Google's newest data center.) There is no ambiguity with this OKR; it is very easy to measure whether or not it is accomplished. Other OKRs will call for rolling out a product across a specific number of countries, or set objectives for usage (e.g., one of the Google+ team's recent OKRs was about the daily number of messages users would post in hangouts) or performance (e.g., median watch latency on YouTube videos).

Second—and here is where thinking big comes in—a good OKR should be a stretch to achieve, and hitting 100 percent on all OKRs should be practically unattainable. If your OKRs are all green, you aren't setting them high enough. The best OKRs are aggressive, but realistic. Under this strange arithmetic, a score of 70 percent on a well-constructed OKR is often better than 100 percent on a lesser one.

Third, most everyone does them. Remember, you need everyone thinking in your venture, regardless of their position.

Fourth, they are scored, but this scoring isn't used for anything and isn't even tracked. This lets people judge their performance honestly.

Fifth, OKRs are not comprehensive; they are reserved for areas that need special focus and objectives that won't be reached without some extra oomph. Business-as-usual stuff doesn't need OKRs.

As your venture grows, the most important OKRs shift from individuals to teams. In a small company, an individual can achieve incredible things on her own, but as the company

grows it becomes harder to accomplish stretch goals without teammates. This doesn't mean that individuals should stop doing OKRs, but rather that team OKRs become the more important means to maintain focus on the big tasks.

And there's one final benefit of an OKR-driven culture: It helps keep people from chasing competitors. Competitors are everywhere in the Internet Century, and chasing them (as we noted earlier) is the fastest path to mediocrity. If employees are focused on a well-conceived set of OKRs, then this isn't a problem. They know where they need to go and don't have time to worry about the competition.

70/20/10

When someone pitches you a new idea, are you inclined to say yes or no? If you have spent too much time in the wrong organization, your knee-jerk reaction is likely to say no. Or "NO!" Organizations have a way of breeding antibodies whose sole purpose is to preach from the gospel of "Thou shalt not." Saying no lets managers avoid risk and reserve their resources (by "resources," we mean headcount, or for those of you who speak human, "people") for projects that are more likely to succeed. Do I really want to dedicate precious smart creatives to that crazy project? What if it fails? They might pull my headcount next year! I think I'll just say no and let my team keep working on streamlining the widgets.

As much as people might preach about creating a culture of yes (including some very smart, handsome people who like to write books), it is very difficult to do without a structural framework to enable it. And when your most precious resource is your people—which is almost always the case—then developing a smart system to allocate those resources is a critical element to success.

In 2002, we still managed resource allocation and our project portfolio by maintaining the top-100 list, but as the company grew we all became concerned that this simple system would not scale well enough. We worried that a creeping culture of no might take hold. So one afternoon, Sergey examined the top-100 list and put the projects in three different buckets. About 70 percent of the projects were related to the core businesses of search and search advertising, about 20 percent were related to emerging products that had achieved some early success, and about 10 percent involved completely new things that had a high risk of failure but a big payoff if successful. That started a lengthy discussion, the end result of which was that 70/20/10 became our rule for resource allocation: 70 percent of resources dedicated to the core business, 20 percent on emerging, and 10 percent on new.

While the 70/20/10 rule ensured that our core business would always get the bulk of the resources and that the promising, up-and-coming areas would also get investment, it also ensured that the crazy ideas got some support too, and were protected from the inevitable budget cuts. And 10 percent isn't a lot of resources, which is fine, because overinvesting in a new concept is just as problematic as underinvesting, since it can make it much harder to admit failure later on. Million-dollar ideas are a lot harder to kill than thousand-dollar ones, so overinvestment can create a situation wherein willful confirmation bias—the tendency to see only the good things in projects in which a lot has been invested—obscures sound decision-making.

Jonathan saw this happen when he was at Apple and worked on the infamous Newton. (For those of you too young to remember, or who worked at Apple and suppressed the memory, the Newton was a digital notebook that was a precursor to today's tablets, except for the fact that it was an #epicfail. A fact that may be interesting only to us: The Newton was manufactured in part by Motorola.) The company had poured a lot of resources into the

product and therefore chose to overlook a major shortcoming. One of the Newton's major features was handwriting recognition—you could write whatever you wanted on the screen and it would recognize what you were saying… theoretically. The problem was that, for most people, it didn't work. In fact, about the only people whose handwriting it could read well were the people who had developed and tested the product, and even they had to adjust their handwriting to achieve their results. Nevertheless, a lot of money had been invested in the project, and the success of the handwriting recognition feature with this small, pliant sample group provided Apple with the answer it wanted to hear. The company plowed ahead with launch plans, and the rest, as the Newton itself might transcribe, is hamstery.

Ten percent also works because creativity loves constraints.[175] It's why pictures have frames and sonnets have fourteen lines. It's why Henry Ford set pricing for his cars so low, because he knew that "We make more discoveries concerning manufacturing and selling under this forced method than by any method of leisurely investigation".[176] A lack of resources forces ingenuity.

In 2002, Larry Page began wondering if it was possible to make every book ever published searchable online—not just the most popular titles or some other subset, but every single book. (We later calculated there were precisely 129,864,880 different book titles in the world.)[177] When every book ever published was available online, he reasoned, *and* when universal translation became available, then all of the world's knowledge would be accessible to every person.

As the cofounder, Larry could have assigned a team of engineers to the problem and given them a nice budget. Instead he got a digital camera, rigged it to a tripod, and set the contraption up on a table in his office. He pointed the camera down at the table, turned on a metronome to pace his movements, and started snapping pictures while Marissa Mayer turned the pages. Based on this crude prototype, they were able to estimate what it took to digitize a book, and made some calculations that the audacious project was indeed feasible. Google Books was born. (Sergey later employed a similar approach to see if Google's Street View project was feasible. He took a drive around town with a camera and snapped a photo every few seconds. He showed off the pictures in Eric's next staff meeting to rally support for what is now called Street View. Today Street View covers over five million miles of roads.)

It's possible that Google Books would have happened if Google had funded it with a coterie of engineers and a healthy budget. But it's also possible that being well funded might have hobbled the project before it could get started. Larry's scrappy digitizing system, built from parts purchased at Fry's,[178] proved to be a lot more cost efficient than the more advanced systems he might have purchased if he had allotted the time and budget. When you want to spur innovation, the worst thing you can do is overfund it. As Frank Lloyd Wright once observed, "The human race built most nobly when limitations were greatest."[179]

20 percent time

In the summer of 2004, a Google engineer named Kevin Gibbs came up with an idea. He described it as a system to "perform real-time completion against all URLs in our repository and all historical Google search queries, with results sorted by their overall popularity." In English, this means that Google would try to anticipate what your query was and suggest ways to complete it. Working in his spare time, Kevin developed a prototype and sent a description of his incipient project to an email list for people who wanted to share new ideas.[180] The note included a link to his prototype, where people could enter queries into Google search and watch the system perform the real-time completion.

The prototype drew the interest of several other engineers, who joined Kevin's project. (Derek Sivers would call these engineers Kevin's first followers.) This feature, now called Google Suggest, is why, when you type "we", Google suggests that you are looking for the weather forecast and provides you with a drop-down menu to click on the full query without needing to type the whole thing out yourself. Google Suggest shaves seconds off search times and helps users get exactly what they need even more quickly. One guy, from idea to launch to global availability to "how did we ever live without it?" for billions of people, in just a few years.

This is the power of 20 percent time,[181] the Google program whereby engineers can spend 20 percent of their time working on whatever they choose. Twenty percent time has spawned a host of great products—Google Now, Google News, transit information on Google Maps, and many more—but it is generally misunderstood. It's not about time, it's about freedom.[182] The program doesn't mean that the campus turns into summer camp every Friday, with all the engineers goofing off in (hopefully) creative ways. In fact, 20 percent time is more like 120 percent time, since it often occurs on nights and weekends. But it can also be stored up and used all at once—Jonathan had one product manager take a summer to work on a 20 percent project. Regardless of when you take your 20 percent time, assuming it doesn't get in the way of doing your regular job, no one can stop you from doing it. Twenty percent time is a check and balance on imperial managers, a way to give people permission to work on stuff they aren't supposed to work on. It helps bring to life the Steve Jobs maxim that "you have to be run by ideas, not hierarchy."[183] And we have found that when you trust people with freedom, they generally do not waste it on extravagant pies in the sky. You don't get software engineers writing operas—they write code.[184]

The Street View trike, which lets us capture ground-level photos along streets and paths too narrow for cars, got its start when Street View cars engineer Dan Ratner went on a trip to Spain. When Dan had to walk the last stretch to his Barcelona hotel through narrow alleyways where his cab couldn't drive, he realized there was a lot of great stuff that the Street View cars couldn't access. When he returned home, he initiated a 20 percent project building a tricycle that could navigate these places, and the Street View trike was born. It has since been adapted to snowmobiles (to chronicle the ski runs at the Vancouver Olympics, for example) and pushcarts (to walk the halls of some of the world's great museums). Up next: Street View skateboards?

There's no doubt that when you give people a lot of freedom, they can be very difficult to rein in. A stubborn smart creative sometimes won't take no for an answer. When is this OK? There is no absolute answer to this question—like all aspects of leadership, judgment matters—but it sure does help if the employee turns out to be right.

When Paul Buchheit decided that email could be a lot better, he started a 20 percent project he called Caribou. At some point, he decided that his new product, which is now called Gmail and has hundreds of millions of users, should generate revenue, so he suggested placing ads next to emails based on the content of the note. We didn't initially agree, and told him to just concentrate on making Gmail great. We'd worry about revenue later.

But Gmail was Paul's baby, and he ignored us. He hacked the internal system to talk to the AdWords ad server (Gmail + AdWords = combinatorial innovation), and one morning we came in to work to see ads alongside our emails. At first people were angry, but then we started noticing that the ads were actually pretty useful. At the time, Jonathan happened to be trading emails with his siblings about what to get their parents for their fiftieth wedding anniversary, and a Gmail ad for Williams-Sonoma appeared in Jonathan's browser next to the note. His sister had brought up their mom's love of gardening, and the ad helpfully suggested

a garden bench. Jonathan proposed the idea to his brothers and sister, and not only did his parents end up with a nice gift, but Jonathan got credit for being sensitive and thoughtful. (This is atypical.)

Gmail launched a few months later. Its ads did not generate much revenue, but the technology that Paul developed to match ads with emails was later refined to improve our AdSense product, which is now a multibillion-dollar business. Needless to say, Paul was not punished for his insubordination.

That example notwithstanding, this doesn't mean that if you work for us and you have an idea we hate, you can just plow ahead with it unhindered by our boorish, wrongheaded, unenlightened managerial opinions. The first step toward bringing a good idea to fruition is to build a team of people who are committed to it, and while *we* may be clueless, your peers probably are not. Our constant advice to anyone who wants to launch a 20 percent project is to start by building a prototype, because that's how you get people excited about the project. Coming up with an idea is pretty easy. Getting a few of your colleagues to join your project and add their 20 percent time to your 20 percent time is a lot harder. This is where the Darwinian process begins.

Finding collaborators can be difficult in a nonhierarchical organization, especially for newcomers, because it's much harder to figure out where to go to get things done. Relationships become highly critical, and since they take time to build (and this isn't everyone's natural skill set), you can end up with a lot of great ideas dying on the vine.

One of our search teams tries to circumvent this through a program called "demo days." The concept is pretty simple: A team spends a week building prototypes of new ideas that they will be expected to demo by the end of the week. Before the week starts, the engineers clear their calendars of all meetings and launches, no exceptions. This not only makes the demo days logistically possible, it serves as a forcing function to get everyone to commit. Since some people may be expected to work in unfamiliar areas, they can get training in the systems they may want to use. All systems are set up and ready to go, so there's no time wasted. They need to recruit at least one other person to their project—no teams of one are allowed—and are encouraged to collaborate with people they don't normally work with on a daily basis. Then the week starts and they get on it.

The result at the end of the week is a set of prototypes, generally shared in a sort of science-fair open house on a Friday afternoon. Most of those prototypes won't progress any further—"I have a lot of ideas and throw away the bad ones," said Linus Pauling[185]—so the teams also learn that it's OK to fail.

Steve Jobs once told Eric that he did something similar when he was running NeXT. Every six months or so, the engineering team would stop what they were doing and dedicate themselves to creating applications for the NeXT platform. This was a critical tactic to building their ecosystem, but it also gave everyone fresh insights into the work they were doing in their "day jobs."

The most valuable result of 20 percent time isn't the products and features that get created, it's the things that people learn when they try something new. Most 20 percent projects require people to practice or develop skills outside Dozens of people rush, creating a mosh pit of dancing fools where once there had been only one. Derek calls this the "first follower" principle: When creating a movement, attracting the first follower is the most crucial step. "The first follower is what transforms a lone nut into a leader." The primordial ooze of innovation needs to encourage the people who want to be innovative—the lone dancing fool on the side of the hill—to do their thing. But just as important, it also needs to encourage the people who want to join something that is innovative—dancing fools two

through two hundred—to do their thing as well. This is why innovation needs to be integrated into the fabric of the company, across every function and region. When you isolate it under a particular group, you may attract innovators to that group, but you won't have enough first followers.

Robert Noyce, cofounder of Fairchild Semiconductor and Intel, said, "Optimism is an essential ingredient for innovation. How else can the individual welcome change over security, adventure over staying in a safe place?"[162] Hire people who are smart enough to come up with new ideas and crazy enough to think they just might work. You need to find and attract those optimistic people, then give them the place to create change and adventure.

Focus on the user…

In late 2009, a team of search engineers showed us a prototype of a feature that had been percolating for a while. It was based on a simple idea: What if we started to populate search results while the user was typing the query, rather than waiting for her to hit the return key? We had always believed that speed was one of the core factors in determining the quality of search and were proud that we answered the vast majority of searches in under a tenth of a second. But that clock didn't start ticking until the user actually entered the query. Typing the query could still take several seconds. What if we didn't have to wait? What if we showed results as the user typed? Once we saw the prototype, the go decision was a no-brainer. Both the organic and paid search teams got to work, and several months later the feature launched to the world as Google Instant.

A few weeks before the launch Jonathan was in his staff meeting when a basic question occurred to him: Would Instant affect revenue at all? Perhaps, when results populated as the user typed, the user might be less likely to click on ads, so he asked his team if we had enough good data about potential revenue impact. No, was the answer, and everyone agreed that someone should investigate it. Then they all proceeded with planning the launch. In every other company where Jonathan had worked, a financial analysis was one of the most important hoops that a product needed to jump through before getting approved. How much revenue will the product make? What will the return on investment (ROI) be? The payback period? Yet here we were, a few weeks away from launching a major change to our core product, and no one had performed a detailed financial analysis. The product was obviously great for the user, so we all knew that launching it was the best business decision.

When Instant launched, it had only a modest effect on revenue, but Google has launched plenty of other features with a more significant financial impact. Knowledge Graph, which launched in 2012, populates the right-hand side of the web results page for queries about people, places, and things with a concisely formatted panel of algorithmically curated information about that entity. It pulls all the most relevant facts together into one, easy-to-read box. For most queries, the panel replaces the ads that previously appeared on that part of the page. That one hurt revenue a little bit. In early 2011, we made a series of changes to our search algorithms that reduced the quality scores of certain types of websites. We didn't want users clicking on links only to find themselves on crummy sites. That release, called "panda," affected nearly 12 percent of queries, and because many of the affected sites were part of our ad network, the change hurt Google's revenue.

Google knows that in the Internet Century user trust is just as important as dollars, euros, pounds, yen, or any other currency. Product excellence is the only way for a company to be consistently successful, so our prime directive when it comes to product strategy is to focus on the user (while not interfering with the internal development of alien civilizations). As

Larry and Sergey put it in their IPO letter, “Serving our end users is at the heart of what we do and remains our number one priority.”

But focus on the user is only half the story. The full sentence should read “focus on the user and all else will follow.” This means that we will always do what’s right for the user, and we trust that our smart creatives will figure out how to make money from it. It could take a while, so sticking it out requires a lot of confidence.[163] But it is usually worth it.

In 2004, Jonathan and fellow Googler Jeff Huber took Sergey on a field trip to meet with a small start-up called Keyhole. Jeff, who had been the first product manager Jonathan hired at Excite@Home, was one of the lead engineers on the ads team.[164] One of Keyhole’s cofounders was Brian McClendon, with whom Jonathan and Jeff had worked at Excite@Home. Keyhole had developed some extremely cool ways to visualize and interact with maps, and Sergey immediately decided that Google should buy the company.

A few weeks later Sergey brought the acquisition to the board for its approval, and when they asked him how we would ever make money from the technology, his response was simple: “I’ll let Jonathan answer that question; he’s the business guy.” Jonathan had been leaning back, enjoying Sergey’s presentation, and had given exactly zero thought to how buying Keyhole might make Google money (clear evidence that after two years, Jonathan was now guzzling from the Google Kool-Aid dispenser). Jonathan plowed forward with some muddled answer that was memorable only in how forgettable it was. The truth was that we had no idea how buying Keyhole would benefit Google’s bottom line.

The board decided to trust Sergey’s judgment and let him proceed with the deal. About eight months later, Google Earth, which was based on Keyhole technology, launched. It was an immediate hit with users, and also generated millions of dollars. How could that be, since there were no ads on the app and it was free? Not long after we launched it, one of our smart creatives, Sundar Pichai, realized that all those people who were downloading and installing Google Earth might be interested in Google Toolbar as well. Toolbar was a simple utility that integrated with the browser. It had a lot of interesting features for users, one of which was a little Google search box that constantly resided in the browser’s interface. People with Toolbar could initiate a Google search without going to Google.com, so they tended to conduct more searches, click on more ads, and generate more revenue. Sundar’s idea met some resistance, but, with a push from Urs Hölzle, it was quickly implemented.

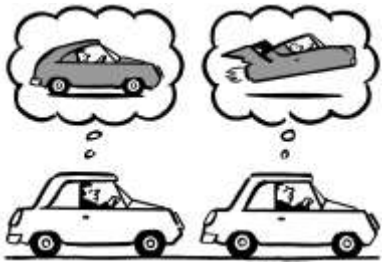
This simple insight—that people downloading Earth might be interested in getting Toolbar as well—increased Toolbar’s user base significantly and generated lots of revenue, but there’s practically no way Jonathan could have anticipated it when he stood before the board that day. Looking back, we realize the correct answer would have been, “I have no idea… but I’m sure we’ll figure it out.”

Focus on the user… and the money will follow. This can be particularly challenging in environments where the user and customer are different, and when your customer doesn’t share your focus-on-the-user ethos. When Google acquired Motorola in 2012, one of the first Motorola meetings Jonathan attended was a three-hour product review, where the company’s managers presented the features and specifications for all of Motorola’s phones. They kept referring to the customer requirements, most of which made little sense to Jonathan since they were so out of tune with what he knew mobile users wanted. Then, over lunch, one of the execs explained to him that when Motorola said “customers,” they weren’t talking about the people who use the phones but about the company’s real customers, the mobile carriers such as Verizon and AT&T, who perhaps weren’t always as focused on the user as they should have been. Motorola wasn’t focusing on its users at all, but on its partners.

At Google, our users are the people who use our products, while our customers are the companies that buy our advertising and license our technology. There are rarely conflicts between the two, but when there are, our bias is toward the user. It has to be this way, regardless of your industry. Users are more empowered than ever, and won't tolerate crummy products.

Think big

Our colleague Vint Cerf has been working on a new suite of network protocols that will work in the harsh and very, very large environment of space. According to Vint, he started this project after asking himself what he could work on that would be needed twenty-five years later.[165] His answer: the interplanetary Internet. No one can accuse Vint of not thinking big enough.



For the rest of us humans, though, it can be a different story. Perhaps it's human nature, or just corporate nature, but most people tend to think incrementally rather than transformationally or galactically.[166] Our colleague Regina Dugan, who was the director of the Defense Advanced Research Projects Agency (DARPA)[167] before coming to Motorola and then Google, talks about how innovation often happens when people are working in "Pasteur's Quadrant,"[168] which is where they try to advance basic science while solving real-world problems. But most companies end up in the opposite quadrant, "where the science is not interesting and no one cares about the goals being pursued. Talent exits, and projects fail more often, not less."[169]

That's why one of Eric and Larry's challenges to engineers and product managers in Google product reviews was always "you aren't thinking big enough." In the Internet Century, with infinite information, reach, and computing power, global scale is available to just about everyone. But too many people are stuck in the old, limited mindset. "You aren't thinking big enough"—later replaced by the Larry Page directive to "think 10X"—helps fix that. It encompasses the art of the possible… and the impossible.

The obvious benefit of thinking big is that it gives smart creatives much more freedom. It removes constraints and spurs creativity. Astro Teller, the head of Google[x], notes that if you want to create a car that gets 10 percent better mileage, you just have to tweak the current design, but if you want to get one that gets five hundred miles per gallon, you need to start over. Just the thought process—How would I start over?—can spur ideas that were previously not considered.

There are other, more subtle benefits to thinking big. Big bets often have a greater chance for success by virtue of their size: The company can't afford to fail. On the other hand, when you make a bunch of smaller bets, none of which are life threatening, you can end up with mediocrity. We see this all the time in business: the company with a large line of not-great products. When Google bought Motorola and Jonathan started helping new CEO Dennis

Woodside, he discovered that the company had dozens of different phones, with each device targeted at a particular, market-research-defined segment, such as millennials, Gen X, boomers, and soccer moms. There was some logic behind this—different carriers wanted their own unique models—but it led to a sprawl of mediocrity. Each phone had its own set of product people who would work hard to make their product great, but they also knew that if their product was only good, the company would survive. (Dennis largely fixed these problems and created a much more user-focused Motorola before Google sold the company to Lenovo in 2014.)

On the other hand, the iPhone is a popular product precisely because it is the only phone Apple makes. If there's a problem in the development of the next-generation iPhone, no one on that team goes home until there's a plan to fix it. It's no coincidence that Apple has only a few product lines. None of them can afford to fail.

It can also be easier to take on big problems because bigger challenges attract big talent. There is a symbiotic relationship between big challenges and highly smart, skilled people: The challenges get solved and the people get happy. Give the wrong people a big challenge, and you'll induce anxiety. But give it to the right people, and you'll induce joy.[170] They enjoy rising to the challenge for the sake of it, but also, as sociologist and management guru Rosabeth Moss Kanter has pointed out, for the very real benefits it can bring: new skills, new connections with colleagues in the field, enhanced reputation—what economists would describe as investing in their human capital.[171] For these reasons, thinking big is actually a very powerful tool for attracting and retaining smart creatives. Put it this way: Let's assume you are a brilliant smart creative just graduating from college. You have two competing job offers, which are virtually identical except for one difference. One of the companies tells you that they like to try to make things 10X better, while the other settles for 10 percent improvement. Which one will you choose?

Our friend Mike Cassidy is a great example of the power of 10X thinking in retaining smart creatives. A cofounder of a company called Ruba, Mike joined Google in 2010 after we acquired Ruba's intellectual property and hired its team. Mike is a serial entrepreneur—Ruba was his fourth company; his second one was a search engine called Direct Hit that briefly competed with Google before it was sold to Ask Jeeves. So we figured that it was only a matter of time before Mike would leave the Google mother ship to start something new. Over time we lost track of what Mike was working on, but we would occasionally see him around campus so we knew he was still a Googler. Then, in June 2013, Google announced Project Loon, the aforementioned Google[x] project that aims to use helium balloons to bring broadband Internet access to the five billion people who don't yet have it. We soon learned that Mike, who has a degree in aerospace engineering, was one of the Loon project leads and had been working on it for over a year. If it weren't for that opportunity to do something audacious, he likely would have left Google, so by continuing to think big and push the boundaries of technology we got to hang on to at least one great smart creative.

And doing big things that matter, as Larry often puts it, inspires people even if they aren't the Mike Cassidys who get to be directly involved in them. We often hear people around Google talk about "10Xing" their job, even though most of their jobs are far removed from the audacious projects the company has become famous for. These are salespeople, lawyers, financial folks, all of whom are inspired by the company's prevalent attitude to shoot for the moon. Thinking big is not only a very powerful recruiting and retention tool, it's contagious.

Set (almost) unattainable goals

The well-worn corporate manager has mastered many skills, and high on that list is the setting of annual and quarterly goals. This requires a certain finesse. Set the objectives too low and you are obviously trying to make yourself look good by "miraculously" exceeding them at the end of the quarter.[172] But set them too high and you run the risk of failure. The trick is to find the sweet spot by creating objectives that look difficult but are actually easily doable. The perfect scorecard at the end of the quarter and year is one that is full of 100 percent marks.

In late 1999, John Doerr gave a presentation at Google that changed the company, because it created a simple tool that let the founders institutionalize their "think big" ethos. John sat on our board, and his firm, Kleiner Perkins, had recently invested in the company. The topic was a form of management by objectives called OKRs (to which we referred in the previous chapter), which John had learned from former Intel CEO Andy Grove.[173] There are several characteristics that set OKRs apart from their typical underpromise-and-overdeliver corporate-objective brethren.

First, a good OKR marries the big-picture objective with a highly measurable key result. It's easy to set some amorphous strategic goal (make usability better… improve team morale… get in better shape) as an objective and then, at quarter end, declare victory. But when the strategic goal is measured against a concrete goal (increase usage of features by X percent… raise employee satisfaction scores by Y percent… run a half marathon in under two hours), then things get interesting. For example, one of our platform team's recent OKRs was to have "new WW systems serving significant traffic for XX large services with latency < YY microseconds @ ZZ% on Jupiter."[174] (Jupiter is a code name, not the location of Google's newest data center.) There is no ambiguity with this OKR; it is very easy to measure whether or not it is accomplished. Other OKRs will call for rolling out a product across a specific number of countries, or set objectives for usage (e.g., one of the Google+ team's recent OKRs was about the daily number of messages users would post in hangouts) or performance (e.g., median watch latency on YouTube videos).

Second—and here is where thinking big comes in—a good OKR should be a stretch to achieve, and hitting 100 percent on all OKRs should be practically unattainable. If your OKRs are all green, you aren't setting them high enough. The best OKRs are aggressive, but realistic. Under this strange arithmetic, a score of 70 percent on a well-constructed OKR is often better than 100 percent on a lesser one.

Third, most everyone does them. Remember, you need everyone thinking in your venture, regardless of their position.

Fourth, they are scored, but this scoring isn't used for anything and isn't even tracked. This lets people judge their performance honestly.

Fifth, OKRs are not comprehensive; they are reserved for areas that need special focus and objectives that won't be reached without some extra oomph. Business-as-usual stuff doesn't need OKRs.

As your venture grows, the most important OKRs shift from individuals to teams. In a small company, an individual can achieve incredible things on her own, but as the company grows it becomes harder to accomplish stretch goals without teammates. This doesn't mean that individuals should stop doing OKRs, but rather that team OKRs become the more important means to maintain focus on the big tasks.

And there's one final benefit of an OKR-driven culture: It helps keep people from chasing competitors. Competitors are everywhere in the Internet Century, and chasing them (as we noted earlier) is the fastest path to mediocrity. If employees are focused on a well-

conceived set of OKRs, then this isn't a problem. They know where they need to go and don't have time to worry about the competition.

70/20/10

When someone pitches you a new idea, are you inclined to say yes or no? If you have spent too much time in the wrong organization, your knee-jerk reaction is likely to say no. Or "NO!" Organizations have a way of breeding antibodies whose sole purpose is to preach from the gospel of "Thou shalt not." Saying no lets managers avoid risk and reserve their resources (by "resources," we mean headcount, or for those of you who speak human, "people") for projects that are more likely to succeed. Do I really want to dedicate precious smart creatives to that crazy project? What if it fails? They might pull my headcount next year! I think I'll just say no and let my team keep working on streamlining the widgets.

As much as people might preach about creating a culture of yes (including some very smart, handsome people who like to write books), it is very difficult to do without a structural framework to enable it. And when your most precious resource is your people—which is almost always the case—then developing a smart system to allocate those resources is a critical element to success.

In 2002, we still managed resource allocation and our project portfolio by maintaining the top-100 list, but as the company grew we all became concerned that this simple system would not scale well enough. We worried that a creeping culture of no might take hold. So one afternoon, Sergey examined the top-100 list and put the projects in three different buckets. About 70 percent of the projects were related to the core businesses of search and search advertising, about 20 percent were related to emerging products that had achieved some early success, and about 10 percent involved completely new things that had a high risk of failure but a big payoff if successful. That started a lengthy discussion, the end result of which was that 70/20/10 became our rule for resource allocation: 70 percent of resources dedicated to the core business, 20 percent on emerging, and 10 percent on new.

While the 70/20/10 rule ensured that our core business would always get the bulk of the resources and that the promising, up-and-coming areas would also get investment, it also ensured that the crazy ideas got some support too, and were protected from the inevitable budget cuts. And 10 percent isn't a lot of resources, which is fine, because overinvesting in a new concept is just as problematic as underinvesting, since it can make it much harder to admit failure later on. Million-dollar ideas are a lot harder to kill than thousand-dollar ones, so overinvestment can create a situation wherein willful confirmation bias—the tendency to see only the good things in projects in which a lot has been invested—obscures sound decision-making.

Jonathan saw this happen when he was at Apple and worked on the infamous Newton. (For those of you too young to remember, or who worked at Apple and suppressed the memory, the Newton was a digital notebook that was a precursor to today's tablets, except for the fact that it was an #epicfail. A fact that may be interesting only to us: The Newton was manufactured in part by Motorola.) The company had poured a lot of resources into the product and therefore chose to overlook a major shortcoming. One of the Newton's major features was handwriting recognition—you could write whatever you wanted on the screen and it would recognize what you were saying… theoretically. The problem was that, for most people, it didn't work. In fact, about the only people whose handwriting it could read well were the people who had developed and tested the product, and even they had to adjust their handwriting to achieve their results. Nevertheless, a lot of money had been invested in the

project, and the success of the handwriting recognition feature with this small, pliant sample group provided Apple with the answer it wanted to hear. The company plowed ahead with launch plans, and the rest, as the Newton itself might transcribe, is hamstery.

Ten percent also works because creativity loves constraints.[175] It's why pictures have frames and sonnets have fourteen lines. It's why Henry Ford set pricing for his cars so low, because he knew that "We make more discoveries concerning manufacturing and selling under this forced method than by any method of leisurely investigation".[176] A lack of resources forces ingenuity.

In 2002, Larry Page began wondering if it was possible to make every book ever published searchable online—not just the most popular titles or some other subset, but every single book. (We later calculated there were precisely 129,864,880 different book titles in the world.)[177] When every book ever published was available online, he reasoned, *and* when universal translation became available, then all of the world's knowledge would be accessible to every person.

As the cofounder, Larry could have assigned a team of engineers to the problem and given them a nice budget. Instead he got a digital camera, rigged it to a tripod, and set the contraption up on a table in his office. He pointed the camera down at the table, turned on a metronome to pace his movements, and started snapping pictures while Marissa Mayer turned the pages. Based on this crude prototype, they were able to estimate what it took to digitize a book, and made some calculations that the audacious project was indeed feasible. Google Books was born. (Sergey later employed a similar approach to see if Google's Street View project was feasible. He took a drive around town with a camera and snapped a photo every few seconds. He showed off the pictures in Eric's next staff meeting to rally support for what is now called Street View. Today Street View covers over five million miles of roads.)

It's possible that Google Books would have happened if Google had funded it with a coterie of engineers and a healthy budget. But it's also possible that being well funded might have hobbled the project before it could get started. Larry's scrappy digitizing system, built from parts purchased at Fry's,[178] proved to be a lot more cost efficient than the more advanced systems he might have purchased if he had allotted the time and budget. When you want to spur innovation, the worst thing you can do is overfund it. As Frank Lloyd Wright once observed, "The human race built most nobly when limitations were greatest."[179]

20 percent time

In the summer of 2004, a Google engineer named Kevin Gibbs came up with an idea. He described it as a system to "perform real-time completion against all URLs in our repository and all historical Google search queries, with results sorted by their overall popularity." In English, this means that Google would try to anticipate what your query was and suggest ways to complete it. Working in his spare time, Kevin developed a prototype and sent a description of his incipient project to an email list for people who wanted to share new ideas.[180] The note included a link to his prototype, where people could enter queries into Google search and watch the system perform the real-time completion.

The prototype drew the interest of several other engineers, who joined Kevin's project. (Derek Sivers would call these engineers Kevin's first followers.) This feature, now called Google Suggest, is why, when you type "we", Google suggests that you are looking for the weather forecast and provides you with a drop-down menu to click on the full query without needing to type the whole thing out yourself. Google Suggest shaves seconds off search times and helps users get exactly what they need even more quickly. One guy, from idea to launch

to global availability to "how did we ever live without it?" for billions of people, in just a few years.

This is the power of 20 percent time,[181] the Google program whereby engineers can spend 20 percent of their time working on whatever they choose. Twenty percent time has spawned a host of great products—Google Now, Google News, transit information on Google Maps, and many more—but it is generally misunderstood. It's not about time, it's about freedom.[182] The program doesn't mean that the campus turns into summer camp every Friday, with all the engineers goofing off in (hopefully) creative ways. In fact, 20 percent time is more like 120 percent time, since it often occurs on nights and weekends. But it can also be stored up and used all at once—Jonathan had one product manager take a summer to work on a 20 percent project. Regardless of when you take your 20 percent time, assuming it doesn't get in the way of doing your regular job, no one can stop you from doing it. Twenty percent time is a check and balance on imperial managers, a way to give people permission to work on stuff they aren't supposed to work on. It helps bring to life the Steve Jobs maxim that "you have to be run by ideas, not hierarchy."[183] And we have found that when you trust people with freedom, they generally do not waste it on extravagant pies in the sky. You don't get software engineers writing operas—they write code.[184]

The Street View trike, which lets us capture ground-level photos along streets and paths too narrow for cars, got its start when Street View cars engineer Dan Ratner went on a trip to Spain. When Dan had to walk the last stretch to his Barcelona hotel through narrow alleyways where his cab couldn't drive, he realized there was a lot of great stuff that the Street View cars couldn't access. When he returned home, he initiated a 20 percent project building a tricycle that could navigate these places, and the Street View trike was born. It has since been adapted to snowmobiles (to chronicle the ski runs at the Vancouver Olympics, for example) and pushcarts (to walk the halls of some of the world's great museums). Up next: Street View skateboards?

There's no doubt that when you give people a lot of freedom, they can be very difficult to rein in. A stubborn smart creative sometimes won't take no for an answer. When is this OK? There is no absolute answer to this question—like all aspects of leadership, judgment matters—but it sure does help if the employee turns out to be right.

When Paul Buchheit decided that email could be a lot better, he started a 20 percent project he called Caribou. At some point, he decided that his new product, which is now called Gmail and has hundreds of millions of users, should generate revenue, so he suggested placing ads next to emails based on the content of the note. We didn't initially agree, and told him to just concentrate on making Gmail great. We'd worry about revenue later.

But Gmail was Paul's baby, and he ignored us. He hacked the internal system to talk to the AdWords ad server (Gmail + AdWords = combinatorial innovation), and one morning we came in to work to see ads alongside our emails. At first people were angry, but then we started noticing that the ads were actually pretty useful. At the time, Jonathan happened to be trading emails with his siblings about what to get their parents for their fiftieth wedding anniversary, and a Gmail ad for Williams-Sonoma appeared in Jonathan's browser next to the note. His sister had brought up their mom's love of gardening, and the ad helpfully suggested a garden bench. Jonathan proposed the idea to his brothers and sister, and not only did his parents end up with a nice gift, but Jonathan got credit for being sensitive and thoughtful. (This is atypical.)

Gmail launched a few months later. Its ads did not generate much revenue, but the technology that Paul developed to match ads with emails was later refined to improve our

AdSense product, which is now a multibillion-dollar business. Needless to say, Paul was not punished for his insubordination.

That example notwithstanding, this doesn't mean that if you work for us and you have an idea we hate, you can just plow ahead with it unhindered by our boorish, wrongheaded, unenlightened managerial opinions. The first step toward bringing a good idea to fruition is to build a team of people who are committed to it, and while *we* may be clueless, your peers probably are not. Our constant advice to anyone who wants to launch a 20 percent project is to start by building a prototype, because that's how you get people excited about the project. Coming up with an idea is pretty easy. Getting a few of your colleagues to join your project and add their 20 percent time to your 20 percent time is a lot harder. This is where the Darwinian process begins.

Finding collaborators can be difficult in a nonhierarchical organization, especially for newcomers, because it's much harder to figure out where to go to get things done. Relationships become highly critical, and since they take time to build (and this isn't everyone's natural skill set), you can end up with a lot of great ideas dying on the vine.

One of our search teams tries to circumvent this through a program called "demo days." The concept is pretty simple: A team spends a week building prototypes of new ideas that they will be expected to demo by the end of the week. Before the week starts, the engineers clear their calendars of all meetings and launches, no exceptions. This not only makes the demo days logistically possible, it serves as a forcing function to get everyone to commit. Since some people may be expected to work in unfamiliar areas, they can get training in the systems they may want to use. All systems are set up and ready to go, so there's no time wasted. They need to recruit at least one other person to their project—no teams of one are allowed—and are encouraged to collaborate with people they don't normally work with on a daily basis. Then the week starts and they get on it.

The result at the end of the week is a set of prototypes, generally shared in a sort of science-fair open house on a Friday afternoon. Most of those prototypes won't progress any further—"I have a lot of ideas and throw away the bad ones," said Linus Pauling[185]—so the teams also learn that it's OK to fail.

Steve Jobs once told Eric that he did something similar when he was running NeXT. Every six months or so, the engineering team would stop what they were doing and dedicate themselves to creating applications for the NeXT platform. This was a critical tactic to building their ecosystem, but it also gave everyone fresh insights into the work they were doing in their "day jobs."

The most valuable result of 20 percent time isn't the products and features that get created, it's the things that people learn when they try something new. Most 20 percent projects require people to practice or develop skills outside pursued. Talent exits, and projects fail more often, not less."[169]

That's why one of Eric and Larry's challenges to engineers and product managers in Google product reviews was always "you aren't thinking big enough." In the Internet Century, with infinite information, reach, and computing power, global scale is available to just about everyone. But too many people are stuck in the old, limited mindset. "You aren't thinking big enough"—later replaced by the Larry Page directive to "think 10X"—helps fix that. It encompasses the art of the possible… and the impossible.

The obvious benefit of thinking big is that it gives smart creatives much more freedom. It removes constraints and spurs creativity. Astro Teller, the head of Google[x], notes that if you want to create a car that gets 10 percent better mileage, you just have to tweak the current design, but if you want to get one that gets five hundred miles per gallon, you need to start

over. Just the thought process—How would I start over?—can spur ideas that were previously not considered.

There are other, more subtle benefits to thinking big. Big bets often have a greater chance for success by virtue of their size: The company can't afford to fail. On the other hand, when you make a bunch of smaller bets, none of which are life threatening, you can end up with mediocrity. We see this all the time in business: the company with a large line of not-great products. When Google bought Motorola and Jonathan started helping new CEO Dennis Woodside, he discovered that the company had dozens of different phones, with each device targeted at a particular, market-research-defined segment, such as millennials, Gen X, boomers, and soccer moms. There was some logic behind this—different carriers wanted their own unique models—but it led to a sprawl of mediocrity. Each phone had its own set of product people who would work hard to make their product great, but they also knew that if their product was only good, the company would survive. (Dennis largely fixed these problems and created a much more user-focused Motorola before Google sold the company to Lenovo in 2014.)

On the other hand, the iPhone is a popular product precisely because it is the only phone Apple makes. If there's a problem in the development of the next-generation iPhone, no one on that team goes home until there's a plan to fix it. It's no coincidence that Apple has only a few product lines. None of them can afford to fail.

It can also be easier to take on big problems because bigger challenges attract big talent. There is a symbiotic relationship between big challenges and highly smart, skilled people: The challenges get solved and the people get happy. Give the wrong people a big challenge, and you'll induce anxiety. But give it to the right people, and you'll induce joy.[170] They enjoy rising to the challenge for the sake of it, but also, as sociologist and management guru Rosabeth Moss Kanter has pointed out, for the very real benefits it can bring: new skills, new connections with colleagues in the field, enhanced reputation—what economists would describe as investing in their human capital.[171] For these reasons, thinking big is actually a very powerful tool for attracting and retaining smart creatives. Put it this way: Let's assume you are a brilliant smart creative just graduating from college. You have two competing job offers, which are virtually identical except for one difference. One of the companies tells you that they like to try to make things 10X better, while the other settles for 10 percent improvement. Which one will you choose?

Our friend Mike Cassidy is a great example of the power of 10X thinking in retaining smart creatives. A cofounder of a company called Ruba, Mike joined Google in 2010 after we acquired Ruba's intellectual property and hired its team. Mike is a serial entrepreneur—Ruba was his fourth company; his second one was a search engine called Direct Hit that briefly competed with Google before it was sold to Ask Jeeves. So we figured that it was only a matter of time before Mike would leave the Google mother ship to start something new. Over time we lost track of what Mike was working on, but we would occasionally see him around campus so we knew he was still a Googler. Then, in June 2013, Google announced Project Loon, the aforementioned Google[x] project that aims to use helium balloons to bring broadband Internet access to the five billion people who don't yet have it. We soon learned that Mike, who has a degree in aerospace engineering, was one of the Loon project leads and had been working on it for over a year. If it weren't for that opportunity to do something audacious, he likely would have left Google, so by continuing to think big and push the boundaries of technology we got to hang on to at least one great smart creative.

And doing big things that matter, as Larry often puts it, inspires people even if they aren't the Mike Cassidys who get to be directly involved in them. We often hear people around

Google talk about “10Xing” their job, even though most of their jobs are far removed from the audacious projects the company has become famous for. These are salespeople, lawyers, financial folks, all of whom are inspired by the company’s prevalent attitude to shoot for the moon. Thinking big is not only a very powerful recruiting and retention tool, it’s contagious.

Set (almost) unattainable goals

The well-worn corporate manager has mastered many skills, and high on that list is the setting of annual and quarterly goals. This requires a certain finesse. Set the objectives too low and you are obviously trying to make yourself look good by “miraculously” exceeding them at the end of the quarter.[172] But set them too high and you run the risk of failure. The trick is to find the sweet spot by creating objectives that look difficult but are actually easily doable. The pursued. Talent exits, and projects fail more often, not less.”[169]

That’s why one of Eric and Larry’s challenges to engineers and product managers in Google product reviews was always “you aren’t thinking big enough.” In the Internet Century, with infinite information, reach, and computing power, global scale is available to just about everyone. But too many people are stuck in the old, limited mindset. “You aren’t thinking big enough”—later replaced by the Larry Page directive to “think 10X”—helps fix that. It encompasses the art of the possible… and the impossible.

The obvious benefit of thinking big is that it gives smart creatives much more freedom. It removes constraints and spurs creativity. Astro Teller, the head of Google[x], notes that if you want to create a car that gets 10 percent better mileage, you just have to tweak the current design, but if you want to get one that gets five hundred miles per gallon, you need to start over. Just the thought process—How would I start over?—can spur ideas that were previously not considered.

There are other, more subtle benefits to thinking big. Big bets often have a greater chance for success by virtue of their size: The company can’t afford to fail. On the other hand, when you make a bunch of smaller bets, none of which are life threatening, you can end up with mediocrity. We see this all the time in business: the company with a large line of not-great products. When Google bought Motorola and Jonathan started helping new CEO Dennis Woodside, he discovered that the company had dozens of different phones, with each device targeted at a particular, market-research-defined segment, such as millennials, Gen X, boomers, and soccer moms. There was some logic behind this—different carriers wanted their own unique models—but it led to a sprawl of mediocrity. Each phone had its own set of product people who would work hard to make their product great, but they also knew that if their product was only good, the company would survive. (Dennis largely fixed these problems and created a much more user-focused Motorola before Google sold the company to Lenovo in 2014.)

On the other hand, the iPhone is a popular product precisely because it is the only phone Apple makes. If there’s a problem in the development of the next-generation iPhone, no one on that team goes home until there’s a plan to fix it. It’s no coincidence that Apple has only a few product lines. None of them can afford to fail.

It can also be easier to take on big problems because bigger challenges attract big talent. There is a symbiotic relationship between big challenges and highly smart, skilled people: The challenges get solved and the people get happy. Give the wrong people a big challenge, and you’ll induce anxiety. But give it to the right people, and you’ll induce joy.[170] They enjoy rising to the challenge for the sake of it, but also, as sociologist and management guru Rosabeth Moss Kanter has pointed out, for the very real benefits it can bring: new skills, new

connections with colleagues in the field, enhanced reputation—what economists would describe as investing in their human capital.[171] For these reasons, thinking big is actually a very powerful tool for attracting and retaining smart creatives. Put it this way: Let's assume you are a brilliant smart creative just graduating from college. You have two competing job offers, which are virtually identical except for one difference. One of the companies tells you that they like to try to make things 10X better, while the other settles for 10 percent improvement. Which one will you choose?

Our friend Mike Cassidy is a great example of the power of 10X thinking in retaining smart creatives. A cofounder of a company called Ruba, Mike joined Google in 2010 after we acquired Ruba's intellectual property and hired its team. Mike is a serial entrepreneur—Ruba was his fourth company; his second one was a search engine called Direct Hit that briefly competed with Google before it was sold to Ask Jeeves. So we figured that it was only a matter of time before Mike would leave the Google mother ship to start something new. Over time we lost track of what Mike was working on, but we would occasionally see him around campus so we knew he was still a Googler. Then, in June 2013, Google announced Project Loon, the aforementioned Google[x] project that aims to use helium balloons to bring broadband Internet access to the five billion people who don't yet have it. We soon learned that Mike, who has a degree in aerospace engineering, was one of the Loon project leads and had been working on it for over a year. If it weren't for that opportunity to do something audacious, he likely would have left Google, so by continuing to think big and push the boundaries of technology we got to hang on to at least one great smart creative.

And doing big things that matter, as Larry often puts it, inspires people even if they aren't the Mike Cassidys who get to be directly involved in them. We often hear people around Google talk about "10Xing" their job, even though most of their jobs are far removed from the audacious projects the company has become famous for. These are salespeople, lawyers, financial folks, all of whom are inspired by the company's prevalent attitude to shoot for the moon. Thinking big is not only a very powerful recruiting and retention tool, it's contagious.

Set (almost) unattainable goals

The well-worn corporate manager has mastered many skills, and high on that list is the setting of annual and quarterly goals. This requires a certain finesse. Set the objectives too low and you are obviously trying to make yourself look good by "miraculously" exceeding them at the end of the quarter.[172] But set them too high and you run the risk of failure. The trick is to find the sweet spot by creating objectives that look difficult but are actually easily doable. The failure later on. Million-dollar ideas are a lot harder to kill than thousand-dollar ones, so overinvestment can create a situation wherein willful confirmation bias—the tendency to see only the good things in projects in which a lot has been invested—obscures sound decision-making.

Jonathan saw this happen when he was at Apple and worked on the infamous Newton. (For those of you too young to remember, or who worked at Apple and suppressed the memory, the Newton was a digital notebook that was a precursor to today's tablets, except for the fact that it was an #epicfail. A fact that may be interesting only to us: The Newton was manufactured in part by Motorola.) The company had poured a lot of resources into the product and therefore chose to overlook a major shortcoming. One of the Newton's major features was handwriting recognition—you could write whatever you wanted on the screen and it would recognize what you were saying… theoretically. The problem was that, for most people, it didn't work. In fact, about the only people whose handwriting it could read well

were the people who had developed and tested the product, and even they had to adjust their handwriting to achieve their results. Nevertheless, a lot of money had been invested in the project, and the success of the handwriting recognition feature with this small, pliant sample group provided Apple with the answer it wanted to hear. The company plowed ahead with launch plans, and the rest, as the Newton itself might transcribe, is hamstery.

Ten percent also works because creativity loves constraints.[175] It's why pictures have frames and sonnets have fourteen lines. It's why Henry Ford set pricing for his cars so low, because he knew that "We make more discoveries concerning manufacturing and selling under this forced method than by any method of leisurely investigation".[176] A lack of resources forces ingenuity.

In 2002, Larry Page began wondering if it was possible to make every book ever published searchable online—not just the most popular titles or some other subset, but every single book. (We later calculated there were precisely 129,864,880 different book titles in the world.)[177] When every book ever published was available online, he reasoned, *and* when universal translation became available, then all of the world's knowledge would be accessible to every person.

As the cofounder, Larry could have assigned a team of engineers to the problem and given them a nice budget. Instead he got a digital camera, rigged it to a tripod, and set the contraption up on a table in his office. He pointed the camera down at the table, turned on a metronome to pace his movements, and started snapping pictures while Marissa Mayer turned the pages. Based on this crude prototype, they were able to estimate what it took to digitize a book, and made some calculations that the audacious project was indeed feasible. Google Books was born. (Sergey later employed a similar approach to see if Google's Street View project was feasible. He took a drive around town with a camera and snapped a photo every few seconds. He showed off the pictures in Eric's next staff meeting to rally support for what is now called Street View. Today Street View covers over five million miles of roads.)

It's possible that Google Books would have happened if Google had funded it with a coterie of engineers and a healthy budget. But it's also possible that being well funded might have hobbled the project before it could get started. Larry's scrappy digitizing system, built from parts purchased at Fry's,[178] proved to be a lot more cost efficient than the more advanced systems he might have purchased if he had allotted the time and budget. When you want to spur innovation, the worst thing you can do is overfund it. As Frank Lloyd Wright once observed, "The human race built most nobly when limitations were greatest."[179]

20 percent time

In the summer of 2004, a Google engineer named Kevin Gibbs came up with an idea. He described it as a system to "perform real-time completion against all URLs in our repository and all historical Google search queries, with results sorted by their overall popularity." In English, this means that Google would try to anticipate what your query was and suggest ways to complete it. Working in his spare time, Kevin developed a prototype and sent a description of his incipient project to an email list for people who wanted to share new ideas.[180] The note included a link to his prototype, where people could enter queries into Google search and watch the system perform the real-time completion.

The prototype drew the interest of several other engineers, who joined Kevin's project. (Derek Sivers would call these engineers Kevin's first followers.) This feature, now called Google Suggest, is why, when you type "we", Google suggests that you are looking for the weather forecast and provides you with a drop-down menu to click on the full query without

needing to type the whole thing out yourself. Google Suggest shaves seconds off search times and helps users get exactly what they need even more quickly. One guy, from idea to launch to global availability to "how did we ever live without it?" for billions of people, in just a few years.

This is the power of 20 percent time,[181] the Google program whereby engineers can spend 20 percent of their time working on whatever they choose. Twenty percent time has spawned a host of great products—Google Now, Google News, transit information on Google Maps, and many more—but it is generally misunderstood. It's not about time, it's about freedom.[182] The program doesn't mean that the campus turns into summer camp every Friday, with all the engineers goofing off in (hopefully) creative ways. In fact, 20 percent time is more like 120 percent time, since it often occurs on nights and weekends. But it can also be stored up and used all at once—Jonathan had one product manager take a summer to work on a 20 percent project. Regardless of when you take your 20 percent time, assuming it doesn't get in the way of doing your regular job, no one can stop you from doing it. Twenty percent time is a check and balance on imperial managers, a way to give people permission to work on stuff they aren't supposed to work on. It helps bring to life the Steve Jobs maxim that "you have to be run by ideas, not hierarchy."[183] And we have found that when you trust people with freedom, they generally do not waste it on extravagant pies in the sky. You don't get software engineers writing operas—they write code.[184]

The Street View trike, which lets us capture ground-level photos along streets and paths too narrow for cars, got its start when Street View cars engineer Dan Ratner went on a trip to Spain. When Dan had to walk the last stretch to his Barcelona hotel through narrow alleyways where his cab couldn't drive, he realized there was a lot of great stuff that the Street View cars couldn't access. When he returned home, he initiated a 20 percent project building a tricycle that could navigate these places, and the Street View trike was born. It has since been adapted to snowmobiles (to chronicle the ski runs at the Vancouver Olympics, for example) and pushcarts (to walk the halls of some of the world's great museums). Up next: Street View skateboards?

There's no doubt that when you give people a lot of freedom, they can be very difficult to rein in. A stubborn smart creative sometimes won't take no for an answer. When is this OK? There is no absolute answer to this question—like all aspects of leadership, judgment matters—but it sure does help if the employee turns out to be right.

When Paul Buchheit decided that email could be a lot better, he started a 20 percent project he called Caribou. At some point, he decided that his new product, which is now called Gmail and has hundreds of millions of users, should generate revenue, so he suggested placing ads next to emails based on the content of the note. We didn't initially agree, and told him to just concentrate on making Gmail great. We'd worry about revenue later.

But Gmail was Paul's baby, and he ignored us. He hacked the internal system to talk to the AdWords ad server (Gmail + AdWords = combinatorial innovation), and one morning we came in to work to see ads alongside our emails. At first people were angry, but then we started noticing that the ads were actually pretty useful. At the time, Jonathan happened to be trading emails with his siblings about what to get their parents for their fiftieth wedding anniversary, and a Gmail ad for Williams-Sonoma appeared in Jonathan's browser next to the note. His sister had brought up their mom's love of gardening, and the ad helpfully suggested a garden bench. Jonathan proposed the idea to his brothers and sister, and not only did his parents end up with a nice gift, but Jonathan got credit for being sensitive and thoughtful. (This is atypical.)

Gmail launched a few months later. Its ads did not generate much revenue, but the technology that Paul developed to match ads with emails was later refined to improve our AdSense product, which is now a multibillion-dollar business. Needless to say, Paul was not punished for his insubordination.

That example notwithstanding, this doesn't mean that if you work for us and you have an idea we hate, you can just plow ahead with it unhindered by our boorish, wrongheaded, unenlightened managerial opinions. The first step toward bringing a good idea to fruition is to build a team of people who are committed to it, and while *we* may be clueless, your peers probably are not. Our constant advice to anyone who wants to launch a 20 percent project is to start by building a prototype, because that's how you get people excited about the project. Coming up with an idea is pretty easy. Getting a few of your colleagues to join your project and add their 20 percent time to your 20 percent time is a lot harder. This is where the Darwinian process begins.

Finding collaborators can be difficult in a nonhierarchical organization, especially for newcomers, because it's much harder to figure out where to go to get things done. Relationships become highly critical, and since they take time to build (and this isn't everyone's natural skill set), you can end up with a lot of great ideas dying on the vine.

One of our search teams tries to circumvent this through a program called "demo days." The concept is pretty simple: A team spends a week building prototypes of new ideas that they will be expected to demo by the end of the week. Before the week starts, the engineers clear their calendars of all meetings and launches, no exceptions. This not only makes the demo days logistically possible, it serves as a forcing function to get everyone to commit. Since some people may be expected to work in unfamiliar areas, they can get training in the systems they may want to use. All systems are set up and ready to go, so there's no time wasted. They need to recruit at least one other person to their project—no teams of one are allowed—and are encouraged to collaborate with people they don't normally work with on a daily basis. Then the week starts and they get on it.

The result at the end of the week is a set of prototypes, generally shared in a sort of science-fair open house on a Friday afternoon. Most of those prototypes won't progress any further—"I have a lot of ideas and throw away the bad ones," said Linus Pauling[185]—so the teams also learn that it's OK to fail.

Steve Jobs once told Eric that he did something similar when he was running NeXT. Every six months or so, the engineering team would stop what they were doing and dedicate themselves to creating applications for the NeXT platform. This was a critical tactic to building their ecosystem, but it also gave everyone fresh insights into the work they were doing in their "day jobs."

The most valuable result of 20 percent time isn't the products and features that get created, it's the things that people learn when they try something new. Most 20 percent projects require people to practice or develop skills outside of those 20 percent time

to point out, 20 percent time may be the best educational program a company can have.

Jonathan's Favorite 20 Percent Project

Jonathan's favorite 20 percent project is the one that digitized and published online the artifacts of Yad Vashem, the Jerusalem-based center for remembering the Holocaust's victims and survivors. The project got its start when Jonathan visited the museum with his family in 2007. During their visit, the guide told them how he had recently learned something about a photo in the exhibit from another visitor. This got

Jonathan thinking about how the youngest survivors of the Holocaust, those who can remember the people and places in these photos, were in their seventies and eighties. Their firsthand knowledge was literally dying out.

The next day, Jonathan visited our Israeli office and talked about the museum. Team Israel took it from there, using their 20 percent time to create a partnership with the museum to build its online presence. Today there are over 140,000 images and documents digitized and online, searchable from anywhere in the world. And the real beauty of it is the survivors who have added their own knowledge and stories to these photos, via comments or videos, to form a more complete picture of this important part of history.

When project leader Yossi Matias demoed the Yad Vashem project for Larry and Sergey, their response was typical: Why only this museum? Why not all museums? Why not create a product that helps all the archives in the world digitize their content? So they did. Google's Open Gallery product, which lets any museum (or owner of cultural content) create online exhibitions, launched in 2013. And the Google Cultural Institute site has hundreds of online collections, with high-definition images of art and artifacts belonging to museums ranging from the Acropolis Museum to the Museo de Zaragoza.

Ideas come from anywhere

The last time you saw a suggestion box, what did you think? Maybe you were at an amusement park or ski area, or maybe you were at work, sipping a coffee in the breakroom. You spied the box and the sign next to it that said WE'RE LISTENING! or WE CARE WHAT YOU THINK! The sign probably inspired just the opposite feeling: You're not listening, you don't care what I think, and the slit in that box transports the paper dropped therein to a wormhole in space, emerging somewhere in the Andromeda galaxy.

How jaded we've become. When the concept of the suggestion box was first introduced to business, it was quite revolutionary. As chronicled in Alan Robinson and Sam Stern's *Corporate Creativity*, the father of the modern-day suggestion box was Scottish shipbuilder William Denny, who in 1880 distributed a pamphlet to all his employees entitled "Rules for the Awards Committee to Guide Them in Rewarding the Workmen for Inventions and Improvements." Denny offered two to fifteen pounds for employee-submitted ideas that were accepted; his workforce responded with hundreds of them over the next decade. Denny's program soon crossed the Atlantic to land at John Patterson's National Cash Register Company, where suggestions from employees peaked at over seven thousand in 1904—roughly two per employee—with a hit rate of roughly one-third.[186] That's over two thousand ideas from rank-and-file employees in one year that were deemed good enough to implement, which is a pretty good hit rate for a suggestion box.

A century later, Marissa Mayer ran meetings at Google that were like *The Gong Show*[187] for geeks. People got up to present their ideas and could keep talking until they were gonged. The better the demo, the longer the person was allowed to go on. Then Craig Nevill-Manning, who founded our New York engineering office, morphed that idea into weekly "Beer and Demos" meetings, where people would gather to sip beer and watch demos, voting with marbles for the ones they liked best. (The demos, that is. The beers they voted on by continuing to drink them.) Patterson and Denny (and Marissa and Craig) were smart enough to realize that good ideas could come from anywhere. They knew that workers could not only work, but think too. This is more true than ever today, when everyone in the organization is

armed with abundant information and great tools. In fact, the biggest danger is not the conceit that only managers have good ideas, it's the conceit that only the company's employees have them. When we say good ideas come from anywhere, we mean *anywhere*. They are just as likely to come from outside the company as from inside. When Google started to expand internationally, we quickly figured out that most California-based engineers lack the skills necessary to translate web pages into other languages. The traditional way to solve this problem would have been to hire professionals to do the translation work, which would have been both expensive and time-consuming. Instead, we let our users do the job. We published all of our text and asked for volunteers to translate it into their local languages. They did, and did a fantastic job. Similarly, when our Geo team set out to chart the world's geography, they discovered that for many areas good maps simply didn't exist. They created a product called Map Maker, which lets anyone contribute to Google Maps. Live on a street that doesn't show up on the Map? No problem: Just draw it in and we'll add it (after we check to make sure it's actually there). Thus was built a new community of grassroots citizen cartographers, who made it so that maps of entire cities were just a mouse-click away for our users. For example, they mapped over twenty-five thousand kilometers of roads in Pakistan in just two months.

Not long after we joined Google, the executive team had an off-site where we discussed opening more engineering offices around the world. When Eric asked Larry how many engineers he thought the company should have, Larry's response was: "A million." He wasn't joking, but he also didn't mean the company should have a million employees (at least, we don't think he meant that). Today, developers around the world regularly work with Android, Google App Engine, Google APIs, Google Web Toolkit, and open-source tools to which Google contributes. These are not Google employees, but if we add them all up it's likely that the total number of people using Google tools or creating cool stuff on top of Google platforms is in the millions. So it could be that we have met Larry's goal, in which case he will probably 10X it to 10 million.

Ship and iterate

So we have just the right amount of resources set aside for new ideas, we've muzzled the imperial managers, freed our geniuses to do their thing, and opened our minds to pull ideas from the masses. The innovation is flowing; good ideas are a-bubblin'. Most of them die before they see the light of day, but a precious few are good enough to reach the promised land. You cue the blog post, push the green button, and pop a cork to celebrate with the team.

Then you get back to work, because if you did the job well, the product is not yet done. Voltaire wrote, "The perfect is the enemy of the good."[188] Steve Jobs told the Macintosh team that "real artists ship."[189] New ideas are never perfect right out of the chute, and you don't have time to wait until they get there. Create a product, ship it, see how it does, design and implement improvements, and push it back out. *Ship and iterate.* The companies that are the fastest at this process will win.

When we launched our flagship product, AdWords, there was a debate about whether new ads should go live without an internal review. There was a strong internal contingent who believed that allowing ads to go live instantly would lead to a lot of bad, spammy ads. But there was another contingent who believed that if advertisers could see their ad in action right away, they could gather performance data and improve the ads much more quickly. They argued that a faster cycle time would lead to higher quality, not lower. We opted for minimal internal reviews, and the ship-and-iterate approach worked.

Ship and iterate applies in many realms. It is easiest to practice in our world of software, where our product is bits and bytes that are distributed digitally, and not physical goods. But with new technologies like 3-D printing and the ability to model many types of things online, the cost of experimentation has dropped in many industries, sometimes precipitously, making a ship-and-iterate process feasible in many more places.

The hardest part of ship and iterate is to iterate. It's easy to rally a team to ship a new product, but much harder to get them to stick around and do the hard work to make that product better. One form of motivation we found to work well was negative feedback. From Larry scrawling "These ads suck" to Marissa posting negative product reviews on the wall outside her office and scrutinizing them with the product managers and engineers, we often used criticism to inspire teams to iterate their products. There is a fine line to walk here, and we haven't always gotten it right. The right criticism is motivating, but too much has the opposite effect.

Ship and iterate doesn't always work. After launching, some products will get better and gather momentum while others will wither. The problem is, by the time a product has gone to market there has been a significant amount of resources and emotion invested in it, which can get in the way of good decisions. Forgetting sunk costs is a tough lesson to heed, so in a ship-and-iterate model, leadership's job must be to feed the winners and starve the losers, *regardless of prior investment*. Products that get better and gather momentum should be rewarded with more resources; products that stagnate should not.

To decide which efforts are winners and which are losers, use data. This has always been the case, but the difference in the Internet Century is how quickly data is available and how much of it there is. A key factor in picking the winners is to decide which data to use and to set up the systems so that it can be retrieved and analyzed quickly. Using data will muffle the sunk-costs fallacy—that irrational tendency most humans have to count the amount of resources that have already been invested in a project as one of the reasons to continue to invest in the project ("We have already invested millions, we can't stop now").[190]

Too often, the tendency is to feed the losers in hopes of making them winners. When Jonathan ran products at Excite@Home, the company's portal, Excite.com, had various sections, such as news, real estate, sports, finance, and so on. Each of those sections competed for clicks on the front page, and when a section's traffic dropped, Excite's management would try to rectify the situation by moving that section to a *better* spot on the page. Hey, finance, your

When Google started to expand internationally, we quickly figured out that most California-based engineers lack the skills necessary to translate web pages into other languages. The traditional way to solve this problem would have been to hire professionals to do the translation work, which would have been both expensive and time-consuming. Instead, we let our users do the job. We published all of our text and asked for volunteers to translate it into their local languages. They did, and did a fantastic job. Similarly, when our Geo team set out to chart the world's geography, they discovered that for many areas good maps simply didn't exist. They created a product called Map Maker, which lets anyone contribute to Google Maps. Live on a street that doesn't show up on the Map? No problem: Just draw it in and we'll add it (after we check to make sure it's actually there). Thus was built a new community of grassroots citizen cartographers, who made it so that maps of entire cities were just a mouse-click away for our users. For example, they mapped over twenty-five thousand kilometers of roads in Pakistan in just two months.

Not long after we joined Google, the executive team had an off-site where we discussed opening more engineering offices around the world. When Eric asked Larry how many

engineers he thought the company should have, Larry's response was: "A million." He wasn't joking, but he also didn't mean the company should have a million employees (at least, we don't think he meant that). Today, developers around the world regularly work with Android, Google App Engine, Google APIs, Google Web Toolkit, and open-source tools to which Google contributes. These are not Google employees, but if we add them all up it's likely that the total number of people using Google tools or creating cool stuff on top of Google platforms is in the millions. So it could be that we have met Larry's goal, in which case he will probably 10X it to 10 million.

Ship and iterate

So we have just the right amount of resources set aside for new ideas, we've muzzled the imperial managers, freed our geniuses to do their thing, and opened our minds to pull ideas from the masses. The innovation is flowing; good ideas are a-bubblin'. Most of them die before they see the light of day, but a precious few are good enough to reach the promised land. You cue the blog post, push the green button, and pop a cork to celebrate with the team.

Then you get back to work, because if you did the job well, the product is not yet done. Voltaire wrote, "The perfect is the enemy of the good."[188] Steve Jobs told the Macintosh team that "real artists ship."[189] New ideas are never perfect right out of the chute, and you don't have time to wait until they get there. Create a product, ship it, see how it does, design and implement improvements, and push it back out. *Ship and iterate.* The companies that are the fastest at this process will win.

When we launched our flagship product, AdWords, there was a debate about whether new ads should go live without an internal review. There was a strong internal contingent who believed that allowing ads to go live instantly would lead to a lot of bad, spammy ads. But there was another contingent who believed that if advertisers could see their ad in action right away, they could gather performance data and improve the ads much more quickly. They argued that a faster cycle time would lead to higher quality, not lower. We opted for minimal internal reviews, and the ship-and-iterate approach worked.

Ship and iterate applies in many realms. It is easiest to practice in our world of software, where our product is bits and bytes that are distributed digitally, and not physical goods. But with new technologies like 3-D printing and the ability to model many types of things online, the cost of experimentation has dropped in many industries, sometimes precipitously, making a ship-and-iterate process feasible in many more places.

The hardest part of ship and iterate is to iterate. It's easy to rally a team to ship a new product, but much harder to get them to stick around and do the hard work to make that product better. One form of motivation we found to work well was negative feedback. From Larry scrawling "These ads suck" to Marissa posting negative product reviews on the wall outside her office and scrutinizing them with the product managers and engineers, we often used criticism to inspire teams to iterate their products. There is a fine line to walk here, and we haven't always gotten it right. The right criticism is motivating, but too much has the opposite effect.

Ship and iterate doesn't always work. After launching, some products will get better and gather momentum while others will wither. The problem is, by the time a product has gone to market there has been a significant amount of resources and emotion invested in it, which can get in the way of good decisions. Forgetting sunk costs is a tough lesson to heed, so in a ship-and-iterate model, leadership's job must be to feed the winners and starve the losers,

regardless of prior investment. Products that get better and gather momentum should be rewarded with more resources; products that stagnate should not.

To decide which efforts are winners and which are losers, use data. This has always been the case, but the difference in the Internet Century is how quickly data is available and how much of it there is. A key factor in picking the winners is to decide which data to use and to set up the systems so that it can be retrieved and analyzed quickly. Using data will muffle the sunk-costs fallacy—that irrational tendency most humans have to count the amount of resources that have already been invested in a project as one of the reasons to continue to invest in the project ("We have already invested millions, we can't stop now").[190]

Too often, the tendency is to feed the losers in hopes of making them winners. When Jonathan ran products at Excite@Home, the company's portal, Excite.com, had various sections, such as news, real estate, sports, finance, and so on. Each of those sections competed for clicks on the front page, and when a section's traffic dropped, Excite's management would try to rectify the situation by moving that section to a *better* spot on the page. Hey, finance, your recruited within Google after the project shut down, precisely because they had worked on something that had pushed the boundaries. And it failed after having created a lot of valuable technology: Pieces of the Wave platform migrated to Google+ and Gmail. As a failure, Wave failed well.

To innovate, you must learn to fail well. Learn from your mistakes: Any failed project should yield valuable technical, user, and market insights that can help inform the next effort. Morph ideas, don't kill them: Most of the world's great innovations started out with entirely different applications, so when you end a project, look carefully at its components to see how they might be reapplied elsewhere. As Larry says, if you are thinking big enough it is very hard to fail completely. There is usually something very valuable left over. And don't stigmatize the team that failed: Make sure they land good internal jobs. The next innovators will be watching to see if the failed team is punished. Their failure shouldn't be celebrated, but it is a badge of honor of sorts. At least they tried.

Management's job is not to mitigate risks or prevent failures, but to create an environment resilient enough to take on those risks and tolerate the inevitable missteps. Author and professor Nassim Taleb writes about making systems that are "antifragile": They don't just survive failures and external shocks, they get stronger as a result.[192] Don't get us wrong: Failure is not the objective. But, if you are measuring the health of your innovation environment, you need to count the failures as well as the successes, to become more "antifragile." As Dilbert cartoonist Scott Adams says, "It helps to see failure as a road and not a wall."[193] Mulla Nasrudin, the thirteenth-century wise fool of Sufi lore, seconds the notion: "Good judgment comes from experience; experience comes from bad judgment."[194]

The timing of failure is perhaps the trickiest element to get right. A good failure is a fast one: Once you see that the project will not succeed, you want to pull the plug as quickly as possible, to avoid further wasting resources and incurring opportunity costs (those smart creatives working on a doomed project could be better deployed working on one that is a potential success). But one of the hallmarks of an innovative company is that it gives good ideas plenty of time to gestate. Projects like self-driving cars or Google Fiber, which will deliver up to 1 gigabit bandwidth to homes (about a hundred times more than the average US household has today), have the potential to be highly profitable, but it will take a very long time. As Jeff Bezos points out, "Just by lengthening the time horizon, you can engage in endeavors that you could never otherwise pursue. At Amazon we like things to work in five to seven years. We're willing to plant seeds, let them grow—and we're very stubborn. We say we're stubborn on vision and flexible on details."[195]

So fail quickly, but with a very long time horizon? Huh? How does that work? (See, we told you this was the tricky part.) The key is to iterate very quickly and to establish metrics that help you judge if, with each iteration, you are getting closer to success. Small failures should be expected and allowed, since they often can shed light on the right way to proceed. But when the failures mount and there is no apparent path to success (or, as Regina Dugan and Kaigham Gabriel put it, when achieving success requires "multiple miracles in a row"),[196] it is probably time to call it a day.

It's not about money

While we believe in paying extraordinary people extraordinarily well for extraordinary success, we *don't* pay people for successful 20 percent projects. Dan Ratner may have received very generous compensation for being part of the transformational Street View product team, but he didn't get anything directly tied to his work on trikes.[197] We don't provide any monetary incentive for 20 percent projects for the simple reason that we don't need to: It may sound corny, but the reward comes from the work itself. Several studies have shown that extrinsic rewards don't encourage creativity, and in fact hinder it, by turning an inherently rewarding endeavor into a money-earning chore.[198]

Our favorite example of the inherent rewards 20 percent projects can yield comes from August 2005, when Hurricane Katrina ravaged the US Gulf Coast. Google Earth had been in the market for only about eight weeks, and the team that developed our "geo" products—Maps and Earth—was small and overworked. But when the hurricane hit it sprang into action, launching over eight thousand up-to-the-minute satellite images (from the National Oceanic and Atmospheric Administration—NOAA) that accurately showed the scope of the disaster and provided high-resolution images of streets and neighborhoods. This helped rescue workers, who were having a hard time navigating when so many signs and signal lights had been wiped out. It also helped agencies distribute relief supplies, and later aided survivors in deciding whether or not to return to their homes.

This was a classic 20 percent project. The idea was born from within a team. No hippo told them to do it. No one suggested they camp out at the office for several nights in a row, no one asked them to reach out to the growing Google Earth community to solicit the help of volunteers, and no one told them to work with NOAA to get the images. In fact, the only executive involvement was when Eric visited their war room, looked around, and issued the wise edict to keep doing what they were doing.

Since Hurricane Katrina, that 20 percent project has turned into a full-time Crisis Response team in Google.org, the organization that leads philanthropic initiatives across Google. Aided by that team, Googlers have used our platforms to help people suffering from natural disasters ranging from the 2008 Chinese New Year snowstorms that stranded thousands of travelers to the 2011 Japan earthquake and tsunami that killed thousands and left hundreds of thousands more homeless. With each disaster, they come up with new ways to use our products to help people, building on previous experiences. Most of them don't get paid a dime for their efforts. They are motivated by the work itself.