Oral History of Kenneth Kocienda and Richard Williamson

Interviewed by:
Hansen Hsu
Marc Weber

Recorded October 12, 2017
Mountain View, CA

CHM Reference number: X8367.2018

Hsu: The day is October 12th, 2017. I am Hansen Hsu. I am joined here by Marc Weber, a colleague. And we are here with Richard Williamson and Ken Kocienda. Okay. And so, we generally start these with some biographical background. Because there's two of you, we'll go over that a little more quickly than usual, but I'll start with asking where did each of you grow up.

Williamson: Well, I grew up, until I was eleven, in England, in Stafford. And after that, my family moved to Phoenix, Arizona. My dad worked for Honeywell. At the time, they were making big mainframe computers. And they had a factory in Phoenix. So, he moved out to work there. Then I went to college on the East Coast, Swarthmore College. And in college, I actually met Steve Jobs. And after my sophomore year, I started working at NeXT, which was his company after he left Apple. And then I worked there on and off. And I actually went back to school and finished school. And I came back to NeXT. And after NeXT, I did a couple of ventures on my own. Then I joined Apple in 2001 and worked there for several years. And after Apple, I worked at Facebook for a few years. So, that's my five-minute-- one-minute synopsis.

Hsu: Very quick. Ken?

Kocienda: I grew up on the East Coast on Long Island in New York. I lived there with my family up until I went away to college. And I went to Yale and studied history. I didn't have any idea I'd be in high tech and wound up dong a bunch of different things after that. I went to motorcycle mechanics school right after college, also in Phoenix, Arizona, interestingly enough. I moved back home for a while, went to Japan, taught English there for a while. Then I went to graduate school to try to get a master of fine arts degree in photography, fine art photography. And it was during that program that I saw-- somebody showed me the World Wide Web the first time. And that just started me off, got into programming and making websites and high tech. And through a bunch of company startups, I wound up at Apple in 2001 as well, just about six weeks before Richard did, as I recall and worked there up until May of this year, 2017.

Hsu: So, what were each of your first experiences with computers?

Williamson: I'll go first.

Kocienda: Yeah, go ahead.

Williamson: So, I think the first time I used a computer was a TRS-80. Remember those things with the cassette tape recorder for storage? And I was part of a group called Special Projects in elementary school. And we were given a couple of hours every week to work on whatever we wanted. And I worked on this TRS-80. And at the same time, too, my dad brought home an old teletype terminal that I could connect with an acoustic modem to a big Multics mainframe. And so, I had one of the very first PCs with the TRS-80 but also access to a big mainframe that was running Multics operating system, which was developed around the same time as UNIX. It had a lot of very modern characteristics. And I learned PL/I,

which was a pretty modern programming language. And that was when I was eleven/twelve, so a long time ago. And I was hooked the moment I started programming at that age. I was a geek pretty much from that point forward.

Hsu: What year was that?

Williamson: That was like late ’70s, must have been, yeah. So, I was born in ’66, so ’77-’78, yeah.

Weber: Okay?

Hsu: Go.

Weber: On the PC, you were doing-- what different things were you doing on the PC and the--

Williamson: Mostly video games. So, there were two things. I wrote some very crude video games. I remember writing a snake program where you had to chase a snake around the screen, and ASCII art. So, we had a-- one of the old printers with a hundred and thirty-two column paper. But the printer could overstrike. So, you could get different shades of gray by overstriking ASCII characters. So, we wrote a program that could actually print out ASCII art on this old printer, very, very crude, but you know, at the time--

Kocienda: Did you have the ASCII art already or did you have to process an image in order to generate some ASCII art?

Williamson: No, we had to process an image.

Kocienda: Yeah, yeah, yeah.

Williamson: It was pretty sophisticated stuff. You know you had to figure out how to get a corresponding--

Kocienda: I mean it’s almost like a rite of passage because I did that, too. Of course, I did. Of course, I did. My uncle had access to mainframes. And yeah, I ended up doing that kind of stuff, sure.

Williamson: And the mainframes back then, they were massive.

Kocienda: Yeah.

Williamson: But computationally, the power they had was less than we have in our iPhones now.

Kocienda: Sure, yeah much less. Sure.

Williamson: I wrote an adventure game, too, in PL/I. Like-- remember the early text adventures like Zork?

Kocienda: Oh, yeah, yeah, yeah, Zork. I played that a lot, yeah.

Hsu: And Ken, so what was your first experience?

Kocienda: You know I had lots of little experiences with computers here and there. I had a Commodore VIC-20 and, within the first week, wrote a program that exceeded the bounds of its memory. It was only about 3K. So, it was just this-- wrote this BASIC program that exceeded the bounds of memory. It was a card game for-- a table game. It was for boxing. So, I just tried to program all the data. And that just wound up taking too much space. I did some programming in high school. They had some-- they had PCs in high school, oh, actually before that. We actually before-- the year before we got PCs, IBM PCs, in our high school, we had access to a time-sharing computer. Who knows where? I don't know where it was. But we actually had cards. So, some of my first programming was on Hollerith cards. And we didn't even have a card punch. So, we were just coloring in with a number two pencil Hollerith cards for Fortran. So, Fortran was one of the first programming languages I used. We got IBM PCs the year after that. And the teacher who ran the course was this real martinet. I mean he was really-- I think he was really wonderful in retrospect. But at the time, he seemed like he was really remarkably strict. And so, it was a first brand new-- first version IBM PCs. And so, we'd write these BASIC programs. And you would have to go-- once you got your program, you wanted to compile it, you-- it wasn't
interpreted. It was compiled. So, you'd have to go up to his desk and get the compiler disk. So, you'd put the compiler floppy disk in, this five and a quarter compiler disk, and do that. And then you'd have to go back up and then get the library disk. So, second-- there's your disk, library disk, turning in the first one of course because you couldn't have two disks at once. And then the third was a linker. So, just-- to do anything was just this laborious process of just trying to get your little ten-line program to do goodness knows what. But this was some of my earliest experiences with computers. I mean, obviously, just primitive and not interactive. The VIC-20 was interactive, but most of my other experiences with anything more sophisticated certainly wasn't.

Weber: But you were clearly, given what you did later with motorcycles, you were interested in technical things, mechanical things, as well.

Kocienda: Well, I'm interested in how things work.

Weber: So--

Kocienda: I mean I have to hold back the urge to just ask about all this camera gear that's pointed at us because I'm interested in it. I want to know how it works. I'm just interested in how things work.

Hsu: Well let's go to the-- your college days. So, you were a history major, but you did a thesis on the history of photography?

Kocienda: Yeah. Yeah, I didn't know what I wanted to do in college. And my sister is two years older than me. She went to college. But nobody in the earlier generations in my family did. And so, I didn't really know what do you do when you-- they were all blue-collar workers. And so, it's like what do you do with an education. They wanted me to get an education. But they really didn't have an idea really what that meant.

Hsu: What's your family's background?

Kocienda: My family's background? Well, my dad was a conductor on the Long Island Railroad, commuter railroad. He did that for thirty years, his whole career. My mom did a bunch of odd jobs. She took care of us when we were young. But she always had projects going on. She was the leader of a Girl Scout troop for a while. She officiated sports. We had a darkroom in our house. She did embroidery. And then, I mean, it's just all kinds of-- she always had projects going on. And, yeah.

Hsu: And-- excuse me. What's your family's ethnic background?

Kocienda: It's all Polish.

Hsu: Polish, okay.

Kocienda: Of course, they go back far enough there was no such thing as Poland. But ethnically-- ethnically Polish.

Hsu: Okay.

Kocienda: All on both sides, yeah. And then my family came to this country a little over a hundred years ago.

Williamson: How many generations is it?

Kocienda: So, it's my-- on my mother's side, my grandparents were born here. So, it's great-grandparents on my mother's side would have come from the old country. And then on my father's side, my-- his parents were both born in the country. So, second generation on my father's side and third on my-- on my mom's.

Williamson: Like everybody else in America, we're all immigrants.

Kocienda: All immigrants. They came to this country with-- they didn't have two nickels to rub together. And how it worked, of course, was why did I grow up in New York? It's because well that's where the boat dropped them off. And they moved to the place where people spoke Polish. My grandfather really never spoke English, on my father's side. And my grandfather on my mother's side, he went to a Polish speaking school. He could read and write Polish because that's what he learned in school here in this country. It was really like a little Poland in Greenpoint in Brooklyn, New York.

Hsu: So, how did you go from being a history major to doing the web?

Kocienda: Well, it's just one thing led to another. I followed my nose. I had spent a lot of time-- you asked me earlier about writing this senior thesis on the history of photography. I got very, very interested in photography when I was at Yale. They had a wonderful art department, wonderful art and architecture library, which really was-- that was my education, really more than anything else, spending time in that art and architectural library just going and pulling books off the shelves and looking at them. And I took courses in fine art photography. And that was the best relationships with professors that I had built up. But by the time I really-- I realized I wanted to do that, it was too late for me to switch to an art degree because I had already taken too many courses in history. And they were in the same distribution group. So, we had this thing-- so, I tried to-- and there was a very, very extensive history of art department as well. But, again, I couldn't do that either. So, there was this notion of regionalism, which one of the professors at school, a guy named-- a fellow named Howard Lamar, who actually wound up later in his career becoming the dean of Yale. I took a course with him. And it was this historical idea that well, the history of a place is really tied to the place. And so, I did this compare and contrast of photography from the East Coast to the West Coast. Certainly in 19th Century when photography-- soon after photography was invented, the photography done on the East Coast of the United States was very much looked toward Europe and the artistic trends going on in Europe, whereas a lot of the early photography of the West was holy cow look at this landscape and was very much about geology, and geography, and Manifest Destiny. It was kind of-- it was really more American ideas than European ideas. And so, that's what I wrote about. And so, there was this real-- I had this real interest and again, going from the beginning, from when my mom had the darkroom in our house, with imagery. And so, it was-- through several steps, I wound up in graduate school for this Master of Fine Arts degree. And this was just not long after, certainly within a year, of Mosaic being developed. And so, I saw this idea, this notion, that hey, maybe we could use this Internet thing to show people pictures. Now, I don't know how it was for you when you first saw the Internet. I won't-- you all-- when I was first shown it, I didn't get it. What's this thing for? It's like okay, this email thing. Okay, well I've got AOL. What do I want this Internet thing for? What's

it good for? I did not get it until just in an elective one night for one of these courses I was taking in graduate school for this fine arts program, the professor said, “Here, I’m going to take you down to the lab. And I’m going to show you the World Wide Web, this thing called the World Wide Web,” which I had no idea about. And yeah, he showed it, and that was it. That’s what then got me interested in the technology. I figured I’ll make the pictures, get into that digital imagery and whatever. And then it was just one thing after another after that.

Weber: But when you saw the web, you got it immediately. You didn’t get necessarily email, or--

Kocienda: I got it immediately. I went home that night and told my wife, “Sweetie, I just saw the most amazing thing. It’s going to change the world.” I got it immediately. And so, it was just the light bulb clicked on. I did not get the Internet before that, and then I saw this thing. And then I did.

Weber: And when-- the original Mac, did that make an impression on you?

Kocienda: Oh, it made a huge impression. I really should have mentioned, I went to school to-- as an undergraduate. I was a freshman in the fall of 1984. So, the Mac had just come out in January of that year. And I was broke. All of my money-- our money went to pay for the Yale bill. So, I couldn’t afford a Mac. But one of my roommates had one. And I was amazed by it. I mean truly, one of-- the first time you saw it, you said, “Wow, this is what I want.” There was no way I could afford it. But he let me use his sometimes. It was just-- yeah, it was remarkable from the first time I saw it.

Hsu: And then for the next several years, you were just doing various web jobs?

Kocienda: Yeah, making websites.

Hsu: Making websites, just like webmaster--

Kocienda: On the-- yeah. Yeah, I changed over from that Master of Fine Arts program into-- luckily, I went to a school, the Rochester Institute of Technology, which was not an art school. It was a technology school that just happened to have a fine arts program in photography. Lucky for me, it wasn’t an art school exclusively. So, I could transfer out of that into a more technology-focused program and did that. But after a year, I knew enough to go get a job. And so, that’s what I did. I didn’t get a degree or any kind of certification or anything from then. I just went and got a job at a small consulting-- it’s actually-- jumping ahead, I actually got a job at a printing company first that wanted to help people make their first websites using their print customers to say, “Hey, look, yeah. We’ll do-- we’ll print this brochure for you. But hey, we’ll put this brochure online for you, as well.” So, making some of the first websites for a lot of companies. And then went into a small consulting shop in Virginia making websites, starting to do CGI programs and little database access that sort of thing, started getting more into the programming side.

Weber: And even though the Mac had made an impression on you, before that, you hadn't thought of computing as a career? It was the web--

Kocienda: No, I had no idea that that was even possible. It's so funny that I never-- the thought never hit me that this wonderful computer that I was looking at was made by people and that you could go and be one of those people. I didn't get it.

Weber: Yeah, that's tr-- East Coast is also very different.

Kocienda: I just didn't get it. I didn't get that you could have a job doing that sort of thing. Again, in my family it's like, "Well, grow up. Be a doctor or lawyer." So, I mean I had some vague notion of being a lawyer. But I almost feel silly saying that because I had no idea what that meant or what you could do-- I just-- so--

Hsu: Richard, I want to go back to you now. So, were you a computer science major in college?

Williamson: No, I studied philosophy.

Hsu: Oh, analytical philosophy?

Williamson: Actually, my focus was on the philosophy of language. And at the time I went to Swarthmore College, they were just setting up a computer science program. And so, it wasn't really an option. But they did have a concentration. So, you could take a few courses. And so, I did that. But I remember Charles Kelemen was the professor. And not to be too modest, but I think I knew a lot more about computers at that point than he did. So, I really helped set up the curriculum and the computer lab. And we had Sun workstations. And I just spent all of my time in the computer lab setting up these computers and working on the computers. But yeah, I studied philosophy not computer science. And really my focus was trying to understand what it is that makes language work. And a lot of the fundamentals of computer science are applicable to philosophy. And you look at somebody like Noam Chomsky. He thinks there's a universal grammar to language and that we can codify all language in code, which I think is completely wrong. I don't think we can do that. But he-- so, there's overlap there. And things like Boolean arithmetic, all of the basics of computer science come out of philosophy. So-- but it took me six years to get through school because I took two years out of school to work with NeXT. Unlike Ken, I pretty much knew that I wanted to work with computers. I didn't think of it as a career. I didn't want to build a career around computing or get a job. It's just what I loved to do. And I kind of just fell into it. And it happened to be something I could make money doing, as well. But in college, I started a company with a guy called Rick Ross called Discovery Software. That was my first two years in college. And we wrote software for the Commodore Amiga. And we had a couple of successful applications. One was called Marauder like the pirate. And it was used to make archival backups of copy-protected software. Back in the day, you'd get software on floppy disks, and you couldn't make a copy of the floppy disks. So, if the disk was damaged, you were out of luck. Also, if you wanted to have more than one copy, for whatever reason, you were out

of luck. So, we wrote a program to circumvent that copy-protection scheme. And looking back on it, we were doing some really sophisticated things with in-circuit emulators to track exactly what the processor was doing with hardware breakpoints. And when it would hit, the registers that were involved with controlling the disk drive, we could capture that. And we'd patch the software to bypass the copy-protection schemes. That was our first successful product. Then we wrote a video game called Arkanoid, which was-- actually it was a port from the arcade game, the upright arcade game. And we got a license to port the ROM. So, the Commodore Amiga had a 68000 processor. And the same code ran in the arcade machine. So, it was pretty easy to take that same code and run it on a Commodore Amiga. And the Commodore Amiga, at the time, was very advanced in terms of its CPU and GPU. Having a co-processor, which was a GPU, was pretty revolutionary at the time. So, it was more than capable of running Arkanoid. And that was pretty successful. So, when-- I read about Steve starting NeXT in *BYTE* magazine. There was a big article, a big profile on him. I said, "I want to do that. It looks really cool." So, I just cold-called NeXT, and they said, "Well, why don't you talk to this guy John Anderson?" And so, John Anderson was one of the early employees at NeXT. And so, I did. And John-- we got to talking. And John said, "We'll fly you out to California. And why don't you come talk to some more people?" So, I flew out. And I showed-- it was-- Bud Tribble was there. Steve Jobs was there. Dan'l Lewin, John was there, Mike Hawley. And the software that we designed for Marauder was highly graphical. It had a picture of the disk drives and the computer. And it would animate as things were happening. In those days, most applications were not graphical at all. And Steve was like, "Oh, that's cool." And they said, "If you'd like to work with us, we'd love to have you do that." So, I worked with Mike Hawley at the time who was-- his vision was to make a digital library. And the idea was to have virtual books that were better than the physical book. And the first project was a dictionary, Webster's. And we got a hold of the-- we had a partnership with Webster. And we got a hold of the original tapes that were used to-- by the printers, I guess, to print the book. And we were able to transcribe those and create a digital edition complete with all of the illustrations. And topography was key to make it look just perfect. And then this was right around the same time that Tim Berners-Lee was using the NeXT to make the early prototypes of Mosaic. So--

Weber: Not Mosaic, but yeah.

Williamson: Not Mosaic, yeah. And he-- so, he was around the NeXT offices a bunch. And it kind of just seemed like a natural thing to do the web. It felt like okay, of course. This is the next obvious step. You want to be able to publish things and make them readily accessible. So, yeah, and then NeXT was a very small company at the time. We had a couple hundred software engineers and very ambitious goals to create a new operating system. And so, I quickly branched out to work on many other things from there, mostly the system applications, things like the preferences application, the equivalent of the Finder, which we called Workspace, and a slew of other utility applications. And then we came out with a laser printer because we wanted to have-- it was very-- the NeXT computer was very expensive. I think it was ten grand when it first came out.

Kocienda: Ten thousand, ten thousand dollars.

Williamson: Yeah, it was a big-- the idea was these would sit at college dorms. And you'd have your NeXT computer. And you'd have a printer to print out your homework or your papers or whatnot. So, we had a laser printer that was really innovative in that the laser printer itself was pretty dumb. And all of the rendering would be done on the CPU. So, we could lower the cost, the manufacturing cost of the laser printer, but still sell it at a comparable price to like what HP was selling at the time. And so-- so, another one of my tasks was to figure out how to make this all work. The hardware was in place, but how do you actually get this to work? So, Leo Horowitz, he was the PostScript guru at the time.

Kocienda: Display PostScript?

Williamson: Display PostScript, yeah. He pretty much implemented that. And so, he was at NeXT. He came to NeXT. And so, we used Display PostScript to render the pages. But then I built all of the other infrastructure around it for spooling. And we added voice alerts to this. It was pretty innovative. So, the computer would actually tell you the printer's out of paper or that it needed attention. And yeah, I can keep talking about the other projects at NeXT. I don't know if this is interesting.

Hsu: No, no, it's good.

Williamson: So, kind of the next big thing we tried to do, which kind of ties into--

Hsu: By this time, were you already an employee? Or were you still an intern?

Williamson: Yeah, I was an employee.

Hsu: Okay.

Williamson: So, I was never really an intern. I was an employee.

Hsu: Okay. So, you were an employee for two years, and then you went back to finish your degree and then went back?

Williamson: A little bit more complicated than that. So, I took a year out and worked full-time in California, came back to the East Coast for a year. And I was actually working on the printer technology full-time from the East Coast, telecommuting back in the-- now that I think about it, it was telecommuting. But I didn't really think about it like that back then. And so, I was spending all my time, full-time, working to get my degree, my undergraduate degree but also working full-time for NeXT. This was my junior year. Then I came out for another year. And then we were working on-- I came back to California. We were working on-- I forget the exact timing of this, but we wanted to use ISDN, which was an old standard where you could get digital 64K connections over a dial-up channel. And you could even have two of

them. And you combine them together and get 128K, which was huge compared to what you had with the old dial-up board mo-- acoustic modems. So, again, like the printer, we wanted to have some pretty dumb hardware but use the main CPU to run the protocol stack and leverage the DSP that we had in the NeXT computers to-- the digital signal processor. So, I did that. And we wanted to work with an outside vendor to make the little adapter for ISDN. So, I was like-- I was-- what was I-- I was twenty-two at this time. And I remember meeting Dennis Hayes of Hayes Microcomputer. And Steve said, "Go and see if you can get Dennis to make this for us." Okay. So, I met with Dennis Hayes. He had an office here in San Francisco. And he was in love with the whole idea of NeXT. He thought it was a great idea. He said, "Yeah, sure. We'll make it for you." So, then we got that agreement in place. And Steve said, "Well, that's great. But why don't you talk to Canon, too," because Canon had invested a bunch of money in NeXT. So, I flew out to Japan, to Tokyo, by myself, and twenty-two, kind of I'm a software developer not a business guy at all, and met with a whole cadre of Canon engineers and business folk. And my dress is jeans and shirt, always has been, probably always will be. But they were all in suits and ties sitting in a conference room with them. And they said, "Well, why don't you use a SCSI port to make this device?" And I'm like, "But we've got this DSP port. And it's perfect for this. And it'd be much cheaper." And they said, "No, we think SCSI is the better way to go." And so, there wasn't really a good meeting of the minds. And it didn't work out with Canon. But it did work out with Dennis. So, Dennis made this ISDN adapter for us. And it was really inexpensive. And I think maybe it was fifty bucks or so. But the market wasn't really there because I think, NeXT, we sold several thousand computers at most. So, selling a peripheral for a computer that is such low volume doesn't make a lot of sense. So, then after that project, I came back, finished college, and then came back to California. And then we started working on the NRW, the NeXT RISC Workstation. And-- fantastic computer. It was-- I don't know if you guys have ever saw one. It was--

Weber: The pizza box?

Williamson: Yeah, pizza box slab design we called it. And--

Hsu: But that was never-- the RISC workstation was never released.

Williamson: Never shipped, yeah. So, but it had-- the ISDN concept was actually built in. So, we had ISDN built into the NRW, the pizza box. And you could plug your ISDN connection directly in the back.

Weber: But sorry, how common was ISDN in this country at the time?

Williamson: It was not that common. And part of the problem with it is that it was a massively complex protocol designed by the telecom companies, way more complicated than it needed to be. And there were two competing vendors, I guess Northern Telecom and-- was it AT&T? And they had slightly different variations of the standards. And so, it was a nightmare trying to get any ISDN device to work. We spent hours and hours trying to get our compatibility to work with both vendors. So, I think it was kind of

doomed to fail. But it was the first purely digital all the way to the consumer, potentially, standard that we could have had. And, obviously, many other things have leapfrogged it now. But, at the time, it was--

Kocienda: Do you think it's interesting that your work, your job, was to take this horrible mess of technology and just make it easy so that if a person did have it, if they had NeXT computer okay, and they were then the subset of the people who NeXT computer that also had an ISDN connection, that they would just take the thing and plug it in the back, and it all could work.

Williamson: That's pretty much been the motivation for all of the technology that we-- or I've developed with NeXT and Apple is to hide the massive complexity of this technology to make it super simple. With ISDN, we created something called PhoneKit so developers could incorporate a digital connection directly with PhoneKit. And you could see that going all the way through to AppKit, and then ultimately UIKit on the iPhone, the idea for consumers making it simple, but also for developers to hide that massive complexity. And now with things like ARKit, hugely complex technology but made very simple for developers. So, yeah, and you know I think that is one of the things that differentiates Apple from other companies, and NeXT at the time, is-- and this comes from Steve, this hyper-focus on making it so simple that anybody can use it. There's no learning curve. There's no manual. And there's a lot of things we can tell you when we get to it about the iPhone that were inspired by that same kind of idea.

Kocienda: Yeah, I mean I think it's really-- it's really interesting that you say both for end users and for developers. It's like knowing who your audience is and then providing that ease of use for who that audience is going to be, that you're going to meet them with the product, and give that target audience the tools to get what they want done easily. For end users, obviously, that needs to go a lot further and the end result is going to be different than it is for developers. But Apple, and NeXT as well, did so well of understanding who the target audience is for a product and then making the product--

Williamson: That's right.

Kocienda: Fit what the person might want to do with it.

Williamson: When we look at kind of the history of how the software that we've developed has evolved, it's all about adding more and more layers of abstraction. So, when we started out, we were dealing with assembly, and toggling bits and bytes, and hardware address registers. All of that now is so far removed from developers and certainly from consumers because of the additional layers of abstraction. And if we look at some of the things that we did at NeXT, which was to introduce object-oriented programming into a mainstream operating system and write these frameworks that added layers of abstraction, those things have lived on all the way through to the iPhone. And it's because we can create really good abstractions that let us build layer upon layer like the foundations of a building and floors of a building that we could be so productive. And I think it's one of the keys to success of NeXT and Apple.

Kocienda: Absolutely, I couldn't agree more.

Hsu: Yeah, you mention AppKit. Did you work closely with the people in the AppKit like Ali Ozer?

Williamson: Yes.

Hsu: Bertrand Serlet?

Williamson: So, we were-- so, I was on the same hallway. Ali Ozer was there and Bertrand Serlet, Blaine Garst, and--

Kocienda: Was Trey around?

Williamson: Trey Matteson was there, yeah. And it was a very small group of people. So, we had the building, actually pretty close to here, over on Redwood Shores. And we had two buildings. One of the buildings was engineering. The other was for everything else. And the second floor of it was software. The first floor was hardware, and-- beautiful building on the inside. I remember Steve spent a fortune on the floors, like he likes to do. And in fact, the first time they put the flooring in he said it wasn't good enough. They ripped it out and put new floors in, hardwood floors. And the staircase from the first floor to the second floor, there's no elevator, was-- no elevator that we could use, was designed by I.M. Pei, the same guy that did the pyramid at the Louvre, and-- beautiful floating staircases. So, the building was very nice. That was the second building. The first building we were in was over in Deer Creek, which is now occupied by Tesla, or Tesla has a building on the opposite side of the road in that same complex. Kind of strange that these two companies both got their start in the same office complex in Deer Creek. So, yeah, so AppKit was-- we were kind of organized where we sat based on the work that we were doing. So, at the other end of the building was Avie Tevanian, who was responsible for the OS. He came out of Carnegie Mellon. And NeXTSTEP was-- the core kernel was based on Mach, which Avie had developed at CMU. He also-- he liked video games. He wrote a version of Missile Command, I remember, just a random bit of information.

Hsu: Yeah, for the Alto, which we have.

Williamson: Oh really? Okay.

Hsu: We talked to him about that. We asked him.

Williamson: So-- Yeah, so Ali Ozer was one of my compadres on the Appkit. So, I worked on, in addition to the applications that shipped with the computer, everything from Chess, to System Preferences, to the Workspace, I worked on the Foundation, the framework, which has kind of now evolved to be

incorporated directly in Objective-C, and a little bit in AppKit. So, I wasn't-- Ali and Trey were really the primary drivers of AppKit. I was a consumer of AppKit. So, often as I was developing an app, we would take some of the technology that I developed and push it down into AppKit so it could be used by other applications, very similar to what ended up happening at Apple, and UIKit for the iPhone. But I remember-- Bertrand Serlet is a brilliant computer scientist. I was so lucky to be able to learn from him because I hadn't really studied hardcore CS in school. I'd just taken a few courses and Bertrand was a great mentor. And we'll come back to him later.

Hsu: So, what years were you at NeXT?

Williamson: So, let's see-- ever I'm always-- I need to have like my resume in front of me while I'm here. So, I think I was there until just prior to the Apple acquisition, which I forget what year that was.

Weber: '96, end of '95?

Hsu: '97 is when--

Weber: End of '96?

Hsu: End of '96, end of '96, yeah.

Williamson: So, I probably left '94, '93-'94.

Hsu: And you had started in?

Williamson: I'd started in '86 or '87.

Hsu: Oh wow, very early.

Williamson: Yeah, right at the very beginning, so I was employee number thirty. And when I joined, they were still setting up the office. Everything was still in boxes. And it was very informal. I remember the main office, we had-- Steve was friendly with the folks that made Bosendorfers. So, there was a Bosendorfer piano, these computer controlled grand pianos. And that was in the main sitting area where we all had lunch. And we would play around with the piano. We had Chris-- what was his last name? One of the other early developers there had a cage with a large snake. And every-- the snake would eat like once every week or two. So-- but it was kind of tradition for all of us to get around and watch the snake being fed a live mouse. It was-- oh and that-- yeah, we-- the early development of NeXT was done on

Sun workstations. And I remember Steve didn't want to buy too many of these workstations because we knew eventually they'd be replaced with [NeXT] prototypes. So, I, being one of the junior guys, didn't get my own workstation. Instead I had a CRT remote terminal that I'd telnet into to the Sun workstation. I had to do all my coding in Emacs on this CRT terminal. But yeah, we kind of came full circle with that because eventually NeXTSTEP turned into OpenStep and was shipping on Sun workstations. But it originally developed-- we developed it on those workstations.

Kocienda: Was this just-- it had like the same CPU? It's like 68020, something like that?

Williamson: Yeah, it was the old 68000 series.

Kocienda: 68000.

Williamson: So, we went from the twenty, and I think the thirty. I forget what the NRW had, but it was one of the last generations of the 68000 series. So, it was really easy to develop, the operating system. And the guy that developed Objective-C was there, too. So, Chris--

Hsu: Steve Naroff?

Williamson: Steve Naroff, yeah. And yeah-- there was another guy besides Steve though. Steve--

Hsu: Brad Cox had created it as Stepstone. And Steve Naroff was the guy at Stepstone that got hired into NeXT.

Williamson: But we had another guy too from Stepstone.

Hsu: Oh, really?

Williamson: Because there was six of us in the office. And Steve Naroff was in-- I shared an office with him, and Chris, and--

Hsu: Are you thinking of Kevin Enderby, who--

Williamson: No, no, Kevin Enderby, he's Mr. Linker, right?

Hsu: Yeah.

Williamson: And--

Hsu: I didn't know somebody else from Stepstone had joined NeXT.

Williamson: He wasn't there for a long time. He left for various reasons. But I'm just trying to visualize the room where we were. If you can imagine six guys in a-- all of us were working crazy hours just like we did on the iPhone, and other projects at Apple. It was just like twenty-four hours a day, seven days a week, we were thinking about NeXTSTEP. And it was kind of similar to the hallway where we worked on the iPhone. Yeah.

Hsu: You mentioned the RISC workstation. NeXT went through this period where obviously the machines weren't selling. And then the whole company had to pivot towards software. And all the hardware was laid off and everything. What was that period like?

Williamson: It was tough because we'd all put our hearts and soul into it. And I think most of the people there were like me. We weren't doing this because it was a career, and we were getting a paycheck. We were doing it because we loved it. And we knew that this was going to be the future of computing, graphical user interfaces, taking everything that had been done at Apple with the Mac, and before that Xerox, and really taking it to the next level. It was just the natural way forward. So, it was really disheartening that we weren't succeeding. People weren't buying our computers. And it wasn't because of the software. It was because of the price. College students can't afford ten grand for a computer, certainly not back then, even now. So-- and Steve was scrambling to keep the company alive. And I think-- I wasn't involved in any of the discussions with Sun, but I think Avie Tevanian and Bertrand were really trying to keep it alive. And so, my-- at that time, there was a big push to see if we could monetize things in other ways and maybe shift the focus to the enterprise, not college students. So, my contribution was to work on something called Enterprise Objects Framework, EOF. And the idea was to make it really easy to bind relational databases to front ends. And we developed technology to map relational databases onto objects that could then be easily bound to the user interface. And it was pretty nifty technology. Eventually, after I left, turned into WebObjects framework, which had some success. But we managed to convince a few clients to use Enterprise Objects Framework to build some applications. And that was reasonably successful but not enough to keep the company going. So--

Weber: Sorry, is that the period when-- I know that there were some trading companies on Wall Street that used--

Williamson: Yeah so, Swiss Bank used it.

Weber: Yeah, right, for derivatives often.

Hsu: UBS right?

Williamson: Yes, and then MCI used it. And I think there's a realtor company that used it too.

Hsu: Oh, the local one, Alain Pinel--

Williamson: Yeah, yeah.

Weber: And then CATS software, didn't they also-- they make computer aided trading systems. I thought they used NeXT for a while.

Williamson: Maybe.

Weber: Or maybe I was their clients that they were-- anyway.

Williamson: So, yeah that work that I did actually helped me after I left NeXT because I got a few very good contracts working with Enterprise Objects Framework, and those few clients that did actually use it--

Hsu: So, you left NeXT in?

Weber: You said '96, I think.

Williamson: Yeah. It was '94-'96, somewhere in that timeframe, it was before the Apple acquisition.

Hsu: Right.

Williamson: So, things were pretty-- it was pretty clear that things weren't working out. And I have a friend who was pushing me really hard to start a company with him. And at that point, Java was in ascendance. And James Gosling, I guess, had just released Java. And we all thought okay, this is going to be great. We're going to be able to write web applications that really are interactive and feel kind of like desktop applications. And you can play it over the browser, and write once, run anywhere, you know the big dream. And so, I kind of bought into that idea. And my friend and I started a company called Infoscape, which-- we developed Java development tools. And leveraging the work I'd done on EOF, the focus was to make it really easy to build database entry forms that you could deploy on the web and-- using Java and something very similar to EOF. And-- but to bootstrap that, I did several pretty significant contracts. The biggest one was with MCI. And they-- do you remember Systems House? They were one of the contractors that worked with NeXT. So, they had--

Hsu: So, you're writing code for-- EOF code for these contracts?

Williamson: Yeah. So--

Hsu: In Objective-C?

Williamson: In Objective-C. I can tell you about this MCI contract. It's kind of interesting. So, it turns out that traditional billing systems are incredibly error prone, or they were at that time. And the error rate was something significant like ten percent of all bills that were sent out by MCI had an error of some kind in them. And they-- so the challenge was to try and write an automated auditing system to catch the errors in these billing systems. And those billing systems are written with COBOL code that is ancient and is just accreted, layer upon layer of cruft. And--

Weber: Abstraction in the less good way, right?

Williamson: Yeah, abstraction is not a word that COBOL developers really ever really understood. So, it was pretty obvious to me that there's no way that can be fixed. And in fact, any change that was made to the COBOL code ran into a problem. And the machines that it ran on were pretty much at capacity. So, you had to worry about the execution time of the COBOL code to-- horrible. So, what I decided to do is we're going to run a job every night to pull the data out of this COBOL database and put it into a modern relational database running on a modern HP workstation. And this modern HP workstation running this relational database could do everything that this massive old mainframe COBOL system could do. And then what I would do is I would-- I wrote a frontend on the NeXT computer with EOF that would display a representation of the bill. And then I could run-- there were several errors that were pretty common. I could run a check on the data and then highlight, in this visual representation, the errors in the bill. And then they still wanted human auditors. So, the human auditors would then look at this visual representation and say, "Yes, that is an error." Then they would have to go back and find a corresponding printout that matched that error and verify that there was an error. So, it was still a tedious process. But compared to what they were doing, it was a vast improvement. And it was a testament to the power of object-oriented programming, relational databases, visual representation of data. So, that contract enabled me to bootstrap the funding of Infoscape. And then we went down, and I had my first experience raising money. And we raised a couple million dollars and I had-- it was-- I can tell you a lot more about the history of Infoscape if you're interested but--

Weber: Well, yes, but one question on NeXT. So, like when you were doing the ISDN, which was pretty early, I mean Tim Berners-Lee did do a prototype web browser and server in almost 1990, end of '89. But prior to that, what were you guys thinking you would do online with the NeXT? What was the vision? There were already-- there was a lot of excitement already about the Internet, but there were also various other paths to go.

Williamson: So, this vision really came from Mike Hawley. He-- do you guys know Mike? He's now-- I don't know if he still is, but he was for a long time a professor at MIT Media Lab, and-- but his vision was that all of the world's information, and he was mostly thinking about books, would be accessible online

and that NeXT would make the first few kind of best examples of what an online book would look like. But then the rest of the world would come along and do the same kind of thing. So, we were really thinking that that was going to be the case and that-- most colleges have Ethernet and high-speed connectivity. You didn't really need something like ISDN for that. And so, that concept really is a precursor to the web. But there was kind of the focus on academia and bringing knowledge and books online. But it's kind of a natural next step to think about making it really easy to publish anything. And so--

Weber: So, he would be the person who had the vision. And you were thinking of using TCP/IP, or--?

Williamson: Of course. TCP/IP is the obvious way to go. Other people are-- telecoms are trying to create other crazy standards, but TCP/IP is the way to go for everything. Big fan of TCP/IP.

Weber: So, the NeXT would have essentially had clients that ran-- you would have-- where would the infrastructure have been to serve up the books? What would the server be? Or that was all blue sky to be worked out?

Williamson: Blue sky, yeah. The Webster's-- the Webster appl-- is a standalone application. All the data was on the NeXT computer, didn't download data. There were other applications like Sherlock and things that came along too that were precursors as well. But NeXT, it was-- the digital library concept I think that was leading into the web, the idea of the web.

Hsu: I'm a little concerned with time. So, maybe talk-- like quickly summarize the Infoscape--

Williamson: Okay, we did Java development tools. Java didn't work out. And we tried to shift direction and move up the value chain. We wanted to make project management software. That didn't work out, closed down Infoscape.

Hsu: So, that was-- how many years was that?

Williamson: It was maybe three years. Then I spent a year at Resonate. And getting back to TCP/IP, Resonate made software load balancers. So, this was-- the web was starting to really become a thing. And there was a problem making servers scale. So, Resonate introduced the idea of virtual IPs so you could hit one IP and then load balance out to many different IPs on the back end. So, Schwab and Excite, which was one of the early search engines, they used our technology to load balance their servers. And Resonate went public, and I left, took some time off, got married. And then, when I came back, we move on to Apple.

Hsu: Okay. I want to turn back to Ken really quick. And--

Williamson: Sorry Ken.

Kocienda: No, no, no. It's great. It's wonderful stories. I'm enjoying listening.

Hsu: So, you worked at Eazel-- how did you get to Eazel? I'm interested in hearing about that.

Kocienda: Oh gosh, how did I get to Eazel? Right, well I was working on the East Coast. I mentioned that I was working at this printing company doing some websites. And then I worked at this professional services company just doing consulting contract work for websites. And this is, again, about the time that Java was getting to be the thing. And I was pretty interested in applets in the browser. And we actually-- it was a manager that I was working with. He said-- first person I ever heard say, "Oh no, this Java thing is not going to be big on clients. It's going to be big on the server. And so, he said, "We're going to write server software." So, we did. We wrote this little application suite. And it was-- so, we had our own templating language. So, the idea was you'd run your software on the server. You hit it with a web browser. The software would run on the server, generate a web page for you, and that would then be delivered to the browser. But all the smarts was on the server. And so, we developed this templating language and database connections, object relational mapping. We did all-- caching, it was really pretty sophisticated to the extent that we were actually featured on Sun's website in like '97 or something like that. I remember that we had one day, just like almost for like a joke, we had WebLogic--

Williamson: WebLogic.

Kocienda: Come in and show us their stuff. And so, they set up as if they were going to be selling-- because we were a professional services company. So, we were kind of like sitting in the back saying, "What are these guys. Are these guys any good?" And we went-- the small group that was developing our server software, we-- I literally remember, we left the meeting laughing with them-- laughing at them how simple their technology was as compared to ours. Well, we, of course, know how well that worked out. They kept improving theirs, and we didn't really keep improving ours because the management of our company just didn't get what we had. So, we could have been a billion-dollar company. But they really wanted to keep doing their little contracts, one-off programming work. So, eventually decided to sell this software. And so, they sold it to a company in Sausalito. I don't even want to mention their name. So, I actually moved with the software, a group of us. Three or four of us went and moved with the software out to this company. And we only lasted five months there because the guy who was running the company had absolutely no idea what he was doing, made his money in a different industry, was going to try to get into high tech, and just didn't have really the slightest idea what it was to run a technology company. And so, he bought this software, spent a couple million dollars on it and then shut it down. It was five months after we got there. So, I was living in San Rafael, commuting to Sausalito every day. The job in Sausalito went away, and so I needed to figure out what to do. And so, it was right around that time that I was surfing around and found this company Eazel online. And just was-- just yeah. There was this little interim period where I tried to get this company going with another fellow, but it didn't really get off the ground. And so, yeah just picked up and moved to Eazel.

Hsu: What was it about Eazel?

Kocienda: What was it about Eazel? It was the people who started it, Andy Hertzfeld. And--

Williamson: Bud was there, right?

Kocienda: Bud was there.

Williamson: Bud Tribble.

Hsu: Bud Tribble was there, too.

Kocienda: Bud wasn't a founder, but he was like a principal. He was like employee number one basically. Mike Boich was the CEO of the company as well and also a fellow named Bart Decrem, who is the other founder with Andy. And so, it was them. And it's kind of funny how it wor-- there was a contingency plan. I said I'm going to go and work on this Linux thing. And if it doesn't work out, it's like well-- I kind of-- by moving-- I moved from the East Coast out to this West Coast company in Sausalito. That wasn't quite Silicon Valley, right? So, you're just kind of like north of San Francisco. So, I figured well, Eazel is at least in Mountain View. So, that's closer. So, if it works, fantastic. If it doesn't work, at least hey, I'll-- connections. I'll get to know these people and whatever and that. That actually all worked out. So, I was doing Java on the server. The-- Eazel, the whole idea was to make Linux on the desktop easy to use. We were doing this in 1999. I guess we're still waiting. <laughter> And this was all based on GNOME. So, we were a GNOME shop. And we were doing the file manager. So, it was Andy Hertzfeld's vision was to make a version of the Finder for Linux. It was called Nautilus, a graphical shell, so the kind of the shell, a play on shells. So, it was called Nautilus. So, that was our shell. And so, the idea was that Nautilus would be GPL'd software. And then we would make online services that we could sell to people. So, it was cloud stuff. I mean early, early cloud stuff. So, we had a software catalog that you could-- running GNOME on the desktop and try to update your software. So, we were-- like we're going to figure out all the dependencies, give you all the libraries you need, get you out of DLL hell, kind of the Linux version of DLL hell, figure that out, give you some online disk storage as well based on WebDAV. And we had this whole client certificate thing. Pretty interesting security model, which I helped to develop.

Williamson: Cloud storage in 1999.

Kocienda: Cloud storage. So, you just-- and it really was meant to be that is that you would have a folder-- It was basically like Dropbox. You would have a folder. And you'd put stuff in that folder. And that folder would just sync. And you wouldn't even know. It would just keep locally cached copies, and it would do that. I mean basically it was Dropbox. It was pretty cool, I mean, if we could get it to work, <laughter> which was ultimately the problem is that we were trying to get this stuff working, kind of do the GPL'd

software to get people enticed to download the software. And then make the online services so compelling that people would then be willing to pay to keep the company afloat because we weren't making any money on Nautilus. But all this came crashing down in 2000 because of the dot com bust. So, we had an A series round of funding. And then the timing was just all wrong. And we ran out of money. And we had to shut the company down. It was really kind of tragic in a way how it happened, was the fellow-- I mean his name's going to come up pretty soon here, Don Melton. So, I was at Eazel with Don Melton. He was in charge of the Nautilus project. I was in charge of the-- director of engineering of the online services. And so, we were involved-- we got pulled in by the executives to the-- and were one of the first people to know in the company that we weren't going to get the funding. And so, this was really-- the timing couldn't have been more awful because we were about a week away from shipping Nautilus. Funding is gone. I mean it's gone. And so, we kept the company going for a week trying to keep the secret that we're going to lay basically everybody off in a week. And so, yeah the same day that we released Nautilus to the world, we laid off two-thirds of the company.

Weber: Wow.

Hsu: That's tough.

Kocienda: It was pretty terrible. I mean you laugh now because it's kind of gallows humor. But it was-- it was really awful. One of the worst workplace experiences I ever had because you're firing dozens of people. And so, actually as Eazel wound down, we wound up trying to make the best of it. So, when the company really did, three months after that, really, really wound down, we prevailed on Mike Boich, who was going to have these horrible, agonizing exit interviews with everybody in the company, we said, "No, Mike, don't do that." There was only about twenty of us left. So, we all got into a conference room. And we all had papers in front of us. And we filled out our names. And then we passed our papers to the right. And so, we wound up firing the person to our left.

<laughter>

Kocienda: And we like countersigned like the person's paper. So, we just had this one big firing party. Okay, so I have one more digression about Eazel, which is really, really, really, I think pretty wonderful. So, we-- there was this interim period. So, we lost the money. We shipped Nautilus and fired two-thirds of the company. And then three months later, we shut down. Well, in the middle of that, we were trying to sell the company, the husk of it. We talked to Sun. We talked to Red Hat. I actually went on the Red Hat visit to North Carolina. And so, but in the midst of all that, there was just a bunch of down time. And so, one day, we got Andy Hertzfeld telling stories. And so, I went over. And he was sitting there in his office chair with an Apple II cracked open in his lap. And so, it's like-- so, he was telling lots of old stories about the Mac. But it was eventually somebody asked him it's like, "Andy, what do you got-- what's with the Apple II?" He said, "Oh, well you know the Missile Command machine?" Missile Command, again. So, you know we actually had a console, you know, an arcade game, full-size arcade game in our break room. He said, "Well, you know how the video was-- the video hold was kind of skitzing out? So, we had

another engineer, a guy named Pavel," Pavel Cisler, who works at Apple, still works at Apple in Finder for a long, long time. He was working with us on Nautilus. And so, Pavel was educated in what's today the Czech Republic. So, he said, "When you did software back then," he said, "Well, you were just a computer engineer. So, you did hardware, too." So he knew how to-- he went back and looked at the Missile Command machine and knew how to fix it. He said, "That chip is bad." So he went and, I guess, talked to Andy about this and Andy said, "Oh, yeah, well, we had that chip on the Apple II board." So Andy went home and brought in like Apple II serial number 5. Pavel went and unsoldered this chip from the Apple II, just like the eight-pin version of the chip that the Missile Command machine uses, a six-pin version of like the same chip. So Pavel took two of the pins, shorted them out to each other, put it in the Missile Command machine and turned, flipped it back on. And fixed the video hold and it's just like perfect. It worked the first time. And so we still had our Missile Command machine to keep us busy. But yeah, so the chip off of Apple II serial number 5 fixed our Missile Command machine easily because Pavel just knew this like arcane knowledge about these old obsolete chips, are pretty something.

Hsu: So you said you knew Don Melton at Eazel. So you mentioned, again, that part of the reason to go to Eazel was to sort of network with all of these people. So clearly that network served you well.

Kocienda: Yeah, that worked. So what happened is when we were getting close to winding down-- what I think happened. I never talked to him to really corroborate this but I think Bud Tribble just picked up the phone and called Steve.

Williamson: Yeah, I'm sure that's what happened. Well, I really don't know but…

Kocienda: I mean I think this is what happened, because what happened is that at Apple we had like this Eazel job fair. <laughter> I'm serious. So in Building 2 on the Infinite Loop campus up on the fourth floor before they filled it in there was the game room. So we had a pool table and there was a couple of arcade machines. So we actually didn't-- like they had an easel, oddly enough, set up with like a print out on foam core, welcome Eazel employees. And there was just like an Easel job fair. There were a dozen or two dozen hiring managers there. And we were just all sitting around on the couches and chairs and having these conversations. And a whole bunch of people from Eazel got hired on at Apple. And just in the direct aftermath of that. Don and I as well. And so I was actually going to go work on the Systems Preferences team. Robert Bowers. Robert Bowers was the hiring manager for System Preferences. So he and I just hit it off and so I was going to take a job with him. But Don Melton was always-- he was a little bit clever. He always had—"I'm just very simple. I mean what you see is what you get." Don is always a little bit clever. He's kind of working behind the scenes. And so he just set up one day-- he didn't tell me what he had going. And he said, "I want you to meet Scott Forstall." So I went and had my interview with Scott Forstall. And I didn't really know what it was for. And Scott Forstall at this time was probably just a Director, of Platform Experience. So he was doing frameworks. And system software...

Williamson: Wasn't he working on the-- was this after the emulation stuff that he was doing?

Kocienda: Yeah, it's after that.

Hsu: After Carbon. Right?

Williamson: After Carbon. Yes.

Kocienda: Yeah, it's after Carbon.

Hsu: OS X had shipped?

Kocienda: Yeah. I mean, I guess, still in the—'cause this is 2001 so mid-2001. So it's probably, you know, Carbon was still going.

Hsu: So OS X had just shipped that March.

Kocienda: Yes, it's three months after. We're like at June of 2001. So Mac OS [X] 10.0 Cheetah had just shipped. So I had my interview with Scott but still didn't really know what it was about. And then, you know, Don and I just one day he was talking to me and he finally spilled the beans. And he was like, "So do you want to make a Web browser?" And I said, well, yeah, I guess. I don't know. And so because he had been working with Scott to start a Web browser project for Apple because, of course, in 10.0 Apple shipped Microsoft Internet Explorer for Mac as the default browser. Apple didn't have a browser of its own. So Scott hired Don to start a browser project. And then Don hired me to get it going. And then six weeks later, we turned around and hired Richard, which is how we met. So it was really the three of us who then-- I don't know if you have follow up questions or not but that's-- it was the three of us that we went and really just laid out some wet cement and then put our handprints in it.

Hsu: Yeah.

Weber: What was the logic of having an in-house browser?

Williamson: Internet Explorer sucked.

<group laughter>

Kocienda: Yeah. Well, I look at it this way…

Weber: But I mean there were other browsers you could have licensed to…

Kocienda: I mean it comes from Steve. It's Steve vision. I mean I think he realized that the Web was going to be big. And that Apple always believed that critical technology needed to be under Apple's control. And so…

Williamson: Although, at that point it wasn't clear that we were going to implement our own browser. So licensing was an option at that point. So the goal was to have a browser that we had more control over than Internet Explorer. And those first few weeks we didn't know what we were going to do.

Hsu: Richard, could you talk about how you came back to Apple or not back to Apple but back to working with Steve Jobs' company.

Williamson: Yeah. After Resonate, I took a long trip, traveled around the world, spent a lot of time in Africa. And then came back to California and said, okay I should probably get back to work. And I called up Bertrand and said I'm interested in joining Apple. It's great that you guys are launching OS X and is there anything interesting you think I could work on? And he said, "Well, come in and I'll tell you about a couple of things." So he told me about two things. The browser project, and he said, "It's just starting. We're trying to figure out what to do." And then the other was, there was an effort that was super-secret at the time to go from PowerPC to Intel. And so he took to me a couple of buildings way off campus and through a couple of secret doors and showed me some prototypes they had of stuff running on Intel which isn't very exciting because it's the same thing. It's just running on a different chip. And so of those two things the browser sounded far more interesting than working on the Intel project. So I said sure. And I didn't know anything about how to actually build a Web engine or a browser. But I thought it would be a fun project. And then Bertrand introduced me to Don. And Don is very energetic and kind of convinced me to do it too. And Avie also was 100 percent behind the project. I think he was SVP at the time. He was just about to leave and then Bertrand took over his role. And I met you [Ken]. And that's how we got started.

Hsu: All right. So let's talk about Safari and WebKit. So I have listened to Don Melton's podcast on the Internet history podcast. So he's talked about the whole process of choosing a browser engine. And so Ken you were the one who actually chose KHTML?

Kocienda: No. No. This was-- to me this was a decision that came in really two parts and it's really this guy right here [Richard] who provided the essential first part. So it's interesting because we haven't compared notes. So it'd be interesting to hear-- I'll give you sort of my brief telling and then we'll let Richard fill in. So here's how I remember it is that we were-- Richard was right in that when he said earlier that what was desired, really, was control of the technology. Not that it be an in-house developed technology. It's just that we needed the in-house control over it. And so we were perfectly open to building-- even thought that fell by the wayside pretty quickly because-- just the complexity of developing a Web engine from scratch. But that also there were sufficiently attractive buy options or download options. Since Web browsers by 2001 weren't bleeding edge anymore. And so open source software, free software did what it does is it goes and takes existing technologies and makes versions of them. So it

was kind of this build, buy or download, is really what we could do. And so Don and I were floundering around for a little bit of a while for a few weeks before Richard came on board. And Richard very, very quickly said, “Now, what are you guys doing?” And he went and he just knocked together, I mean just almost, it’s like knocked together this demo with KHTML. And we told him what we were doing and what our options were and what we would be looking at. We were looking at Mozilla. We were looking at maybe licensing from Microsoft, Internet Explorer, for Opera versus iCab versus-- There’s KDE and GNOME, these Linux desktops or whatever. And Don and I were just kind of tripping over ourselves. We didn’t really know what to do. Too many options. And so Richard goes away and just says, “How about KHTML?” And so he goes away for a couple of days. And then he calls us in for a demo and says, “Hey, guys look.” And he’s got KHTML working on a Mac. So it’s just like, well, X Windows, there’s an X Windows version for Mac even though that’s not the native graphical system, no big deal. We just take this KHTML code and you kind of slap it down in there and just kind of get all of the pieces together, duct tape and bailing wire and whatever. And there’s Konqueror, the KHTML browser running on a Mac. And Don and I were looking at this going, what planet is this guy from? How did he do this? Basically. It was basically. I mean maybe I’m romanticizing but it’s basically, you just made this like this little Linux emulation system to convince this KHTML code because there just wasn’t that much time. So he just convinced this KHTML code that yeah, yeah, yeah, you’re running on a Linux machine even though it’s a Mac. No, its X Windows. It’s fine. Don’t worry about anything else. And we were floored. And the thing was really the analysis after that was, okay well, so obviously this guy is pretty good at knocking stuff together. And KHTML can run on a Mac. Well, what does it look like? Well, Mozilla was the leading candidate at that point, the other leading candidate but it was a million-and-a-half lines of code. And KHTML was 150,000 lines of code.

Weber: Wow. Oof.

Kocienda: And there you go. It was three guys and we figured 50,000 lines of code each. But really it was just the simplicity and the convincing-- the utterly amazingly convincing demo that Richard made to get this code working on the Mac. Well, obviously, it can work on the Mac. So now we just need to strip off the parts that we don’t want, change over from X-Windows to like, CoreGraphics. And you get that little sense of hubris, how hard can it be? And there you go.

Weber: And was…

Kocienda: I don't know if you agree with that. But that’s kind of what happened.

Williamson: Pretty much!

Kocienda: <laughs> So it’s him. So it’s basically to kind of join up my description, is that there’s two parts to that decision. The first part was Richard’s amazing demo. The second part was this analysis to go, wow, what did we just see? Does this really actually make sense at all? Huh. Yeah, it’s a small code base. Much smaller than that one. We’ve got X Windows. We need to swap out the graphical system. We need to strip out these parts that we don’t need so we just have the web engine. Bring that over to the

Mac. Start hacking on it from there. That's a plan. Right? And everybody we showed [said] "yeah that's a plan." So that's what we did.

Hsu: And I mean, it's particularly interesting because Don had worked on Mozilla.

Kocienda: Yeah.

Williamson: Oh yeah, he managed that team. Right?

Kocienda: Yeah, well, he was involved, he was a manager on Mozilla. And is actually-- there's this wonderful PBS-- I don't know if they produced it. I saw it on PBS-- documentary called Code Rush which is the story of Microsoft coming after Netscape and Netscape's response of open sourcing the browser code making-- turning Navigator into Mozilla basically. And there was this hour-long documentary and Don was in it. And I seem to remember his job was to clean up the source code so that it didn't have any dirty language in it. <laughs> And that there were these deals that they did with other companies. And so, of course, it's closed source so it's like yeah, yeah, those jerks over at company X they're freaking-- they don't have any idea what they're doing. So we're putting this workaround in there because their mail system or their Web server software is crazy or whatever. So he had to purge all of that software so that the source code could be put-- made available online. And so yes, Don knew Mozilla too well to want to work with it again.

<group laughter>

Kocienda: So we had this thing that was a whole lot simpler that kind of Richard showed us the way. And so if Don and I were smart at all we were smart enough to realize that we just got shown something that was too good to be true. Really.

Weber: And before that point, Opera was never a serious?

Williamson: That's not true. So when I-- more or less I agree with everything you said. When I joined I thought there were three potentially viable options, Opera and licensing something from them, and Mozilla, and KHTML.

Weber: Sorry, and the Amaya, the W3C, there wasn't enough there going on to have? OK.

Kocienda: No, never looked at it.

Williamson: So the Opera thing we actually, I think had scheduled a trip to go out and talk to them. And so it was a pretty serious contender. I mean that browser was far more advanced than KHTML. And it could have worked. But…

Weber: So you went over and talked to…

Williamson: We didn't. We scheduled a trip but we didn't actually take that trip because this other path presented itself as being a really good path. But when I looked at the Mozilla code, I remember checking that thing out. Beast. I think an important thing in software development is that the software does what it needs to do well and doesn't do things that it doesn't need to do. And the Mozilla code base has so much stuff in there that's irrelevant for a Web engine. Interesting technology in other domains but really irrelevant in terms of building a Web engine. And in order to get your head around a piece of software you need to be able to context switch it into your head and really understand how it works. It's almost impossible with Mozilla because of all of these irrelevant subsystems. So it was really clear to me that Mozilla was out. And we didn't want to add one-and-a-half million lines of code to the operating system. And we knew that WebKit was going to be fundamental not just to the browser, but to other aspects of the system.

Kocienda: See this is the point. I'm going to interrupt you there just to say, WebKit. We say WebKit and Safari and it's like "well, we were hired to make a Web browser." Well, no, let me fill this in. We were hired to make a browser app that we could go and replace Internet Explorer as the double clickable app in the Dock. But we also from the very, very beginning we needed to make a framework. We needed to make a developer toolkit. That was part of the plan from the very, very beginning. And so WebKit was not after thought. No, no, no. It was there from the earliest, earliest direction coming down from Steve.

Williamson: And it gets back to this idea of abstractions. So WebKit takes the complexity of a web browser and distills it down into something which is really easy for developers to use as well as making a great consumer application. So WebKit was having something which was going to be a fundamental part of the system on a peer level with AppKit was really important because there were a few ideas emerging at that time. And one point idea was native applications versus Web-based applications. And it was pretty clear that there was a potential to build Web-based applications that would be almost as good as native applications. This wasn't going to be Java, write once, run anywhere. It was going to be based on Web standards, HTML, CSS and JavaScript. So I was looking at it from that perspective, you know, what can we build that's going to be the foundation for many other things moving forward? And Mozilla wasn't going to cut it.

Weber: And you couldn't cut away those parts in Mozilla. It was too intrinsic?

Williamson: I actually tried. So I spent maybe a day which doesn't sound like a lot but I can do a lot in a day.

Kocienda: For him it is. For him a day is a long time.

<group laughter>

Williamson: And it was just too…

Kocienda: Just too much stuff. I mean there's two stories I can give to [give] some perspective, perhaps a little programmery perspective on Mozilla is that [the] Mozilla source code base had 20,000 typedefs. So you want to know what is in that software, you've got to learn 20,000 names. Okay. So not only that I started-- I actually did get Mozilla running on Mac OS X which was a weeklong, which was a horrible, horrible trial because Mac OS X was so new that Mozilla didn't have a port. And this was secret. So you couldn't ask anybody. So I had to figure it all out for myself. So I got Mozilla running on Mac OS X but it basically never-- I don't think it ever rendered a webpage. It would just crash. And this is just a couple of days of work. But still starting to get into-- so after a crash I'd look at the core file. It's 100 levels deep, 100 frames, 100 stack frames deep. So how are you going to context switch this into your mind? Okay. Well, maybe things went wrong in frame 57. And eventually it crashed in frame 102. I mean forget it.

Williamson: The other good story about the early decision days was, we call this environment, or I call this environment spooge [ph?] because it was such a hack. But it was a very compelling hack and I actually made it look like it was a Mac Window even though it was running X Windows. So it looked like a native app. So I think the three of us were all convinced this could work. But we had to do a presentation to Avie. And so Don said, "Okay, we're going to do the presentation but we're not going to use PowerPoint or any other application. We're going to do the slides in this application. And then we'll project that." So that's what we did. We actually made HTML slides and presented it in the browser and it was flawless. And we didn't tell Avie until the end of the presentation [that] we had been using the application.

Weber: The whole time.

Williamson: The whole time. Yeah. And then Avie was convinced.

Kocienda: Yeah, he was convinced.

Williamson: And he gave us the greenlight and then we went forward.

Weber: And the functions you wanted this to do were where do they map or not map to conventional browsers. I mean you were looking for-- saying it could be integrated, really at a lower level.

Williamson: Yeah. So I mean the idea of incorporating a web view into any other application is great. Everything from showing the documentation snippet, to having a JavaScript engine.

Kocienda: Simple things like maybe an app wants to have in their About Box the release notes. Well, the release notes can just be online. You don't have to bake it into the program necessarily.

Williamson: The other thing is at the time the RTF and RTFD were kind of the best Apple had in terms of formatted text. So if you wanted to do anything sophisticated in terms of layout, Apple didn't really have an engine to do that. Things became a lot better when the text system advanced later on. But in terms of just simply trying to have multiple fonts and colors and underlying text, RTF was pretty crappy. And I did the RTF implementation at NeXT that carried forward into OS X. So I knew that RTF wasn't going to cut it in terms of a rich formatting system. And the Web had the potential to do that. So even just like two lines of text can be in a web view, can be like a webpage but it doesn't have to look like a webpage in a web browser from the developer's perspective.

Hsu: So for instance, having HTML in Mail…

Weber: Yeah, I that read in your…

Kocienda: Yeah, that happened later. That happened after we shipped the 1.0 version of Safari. Scott Forstall called me into his office and said, "Ken, we've got a problem. We're getting more and more HTML mail sent to us from Outlook on Windows." Outlook, they had an HTML editing engine in their mail program. And so the issue was that, well, when we detected a mail message coming in that was HTML we could switch out the RTF view for a Web view so we could render it. But then when you hit the reply button there was nothing to do. I mean what do you do? It's like, well, we could have this embedded webpage message that you received as an attachment on RTF. It just wouldn't be editable. It would be this nugget of content that was basically just one character in an RTF document. It was just at an offset. And so Scott said this is no good. And so we need to be able to edit HTML. And so could you add editing to WebKit and then that was…

Williamson: Just add editing to WebKit.

Kocienda: Just add editing to WebKit. That was two years running, two years of my life.

Weber: So that was not in the beginning?

Williamson: No, that was not.

Kocienda: That was what?

Weber: So the editing was not something you thought of in the original spec for…

Kocienda: No, that wasn't. That came later.

Weber: Got it. So once you had-- well, I don't want to skip. I didn't realize it was that far ahead.

Kocienda: It was that far ahead.

Weber: We'll go sequentially.

Hsu: Yeah, let's maybe go back to just talking about what work it took to actually ship the first version of Safari? I mean it came out first for 10.2, for Jaguar? But it wasn't shipped with the operating system until 10.3.

Kocienda: Yeah, we had a beta.

<overlapping conversation>

Williamson: With a button [ph?] right on the tool bar.

Weber: Yeah, I think I remember that.

Kocienda: Yeah. We had a six-month beta. It came out in-- because it was eighteen months. it was January 2003. So it was like a MacWorld, January 2003. And yeah we had like a six-month beta. I don't know what version that matches up with in OS X history but that's it.

Williamson: But I think the work that we had to do wasn't really in the engine itself which it turned out was lacking a lot of areas. But it was just getting it integrated into the system and performance. Performance, I think, was our number problem just optimizing the hell out of it.

Kocienda: Yeah, performance and correctness, things like its performance, Richard did just absolutely brilliant work with trying to get text rendering quickly. So work with CoreGraphics and you went and talked with…

Williamson: John Harper [ph?].

Kocienda: …John Harper to try to get-- somebody else who came from Eazel-- to just get characters blasted onto the screen as quickly as possible.

Williamson: I had to learn about Arabic text shaping.

Kocienda: Yeah, we had. And we had a fast path and a slow path because some character sets, some script systems, we didn't quite have a fast path working but Richard just, again, just took all of this stuff and put it under this level that you didn't need to think about it and the text would render as fast as we could. And then we could go and change that and improve it over the time. But you didn't need to worry about, as a developer or naturally not as an end user. It's just that you would get this text and it would just render as quickly as possible. Of course, this, all that work wound up having really important, like performance, really, really important implications down the line.

Williamson: For iPhone.

Kocienda: For the iPhone.

Weber: Right. And the small size.

Kocienda: Small size. Yeah. Because we had all of that performance groundwork done. And so that when we-- because we're skipping ahead now but when we back to even a less powerful computer, it's like we could still be blasting text on the screen no problem.

Weber: And you talked about Arabic script. Were you dealing at all with the standards with W3C or any standards stuff? Or you were…

Williamson: Not at that point. I mean it really for the first version we were just focusing on trying to make what was there work well and work correctly and getting the performance to be there. Obviously, later, we got very involved in W3C. But for 1.0 no. How many people were [there] for 1.0? We were like seven maybe. It was a tiny, tiny group. Especially if you compare it to Internet Explorer or Mozilla. It was a pretty herculean effort.

Kocienda: Yeah, but it just was people who just were committed. And then we had people like Dave Hyatt who probably knows more about Web engines than anybody on the planet.

Williamson: Well, he didn't come on until later, though.

Kocienda: He didn't come-- well, he was there for 1.0.

Williamson: He was? Okay.

Kocienda: I'm pretty sure he was there for 1.0. But you're right, he didn't come on until maybe like a year into it.

Williamson: But Dave is a god in terms of Web standards and W3C. It's in his genes almost.

Kocienda: Yeah, well, no it's really more the other things. It's the Web is his genes.

Williamson: That's right.

<group laughter>

Kocienda: Yeah, that's kind of how that works. But I mean look we had a whole lot of really, really talented smart people, Darin Adler, John Sullivan, Maciej Stachowiak, Chris Blumenberg. This guy over here.

Williamson: It was a really good group of people. Incredible team. Yeah, they always say that one good engineer is as good as ten average engineers or one really great engineer is. And I think we had a whole team full of just stellar engineers. And it's the kind of group that just knows how to do the right thing. And we worked together so much we just executed.

Kocienda: Well, you know, it's funny because we would-- I mean I know that we would pick up like in a posse and just go down to the café, you know, it was on the Infinite Loop campus, eat lunch there and then just pick up and then just go back to the office. It's kind of like this little massive half-a-dozen geeks just spending all of their time together. And I remember that even at lunch there was really only two topics that I can really ever remember talking about at lunch. We would talk about the Web browser or we'd talk about World of Warcraft. <laughs>

<group laughter>

Kocienda: Because right at this time-- because it was early-- because now Darin Adler had before he-- I mean and this is somebody you should to talk to but he was working in Southern California for a while. And he was doing contracts and whatever. And so he knew some people at Blizzard. And so he got us into the World of Warcraft beta, I don't know, a year, eighteen months before they shipped. So we were still-- where you could run behind walls and you could get stuck behind the scenery and all of this. But we were just playing. So we would spend eight or ten, eleven, twelve hours a day at work. And then we'd go home and we'd log into World of Warcraft and be playing until 3:00 A.M.

Williamson: I think we say we probably lost about six months of productivity. Playing that beta.

<group laughter>

Kocienda: I mean I can just remember I mean it was like why did we do this? I can just remember the best thing that I discovered in World of Warcraft, the best feature that they put in was follow. So you could just take your character and just follow. And I'd be like over there at 2:00 A.M. just like dozing off just on follow as we're running across some zone to go and get five silver rings and whatever I don't know even know why. Five, we need five, good. Yeah, let's go and losing all of our sleep over this. But that's basically what we did. But we were just a very close-knit team. I mean there's small team with high team cohesion and they're really outstanding individuals. Well, you can get a lot of stuff done in a short period of time and we did.

Weber: And for support of different alphabets, so Unicode was already pretty well settled on the Web.

Kocienda: Yeah.

Weber: So you were supporting which language character sets?

Kocienda: All of them.

Weber: Okay. Immediately.

Kocienda: Yeah, all of them.

Williamson: So I mean double byte characters as well as single byte characters. So UTF-8 is the standard for pretty much Roman languages. But then UTF-16 is for double byte characters.

Kocienda: Yeah, but we had-- the Web is a…

Williamson: Global thing. World Wide Web.

Kocienda: Yeah. So I mean there was JIS for Japanese and Big 5 for Chinese.

Williamson: Emoji, though, that didn't come until after 1 point--

Kocienda: Yeah, that was way after.

Williamson: That was iPhone. But we extended-- we're skipping ahead but we ended the-- so ultimately the iPhone used the WebKit text rendering system to render all text, not AppKit's text rendering system, because the CPU and GPU weren't powerful enough yet. But we had to do emoji. I'm skipping ahead.

Kocienda: You're skipping ahead.

Williamson: Okay. Is that okay? We'll wait.

Weber: Well, we can go sequentially. We won't forget emoji.

Hsu: So do you want to finish up talking about the editing, the HTML editing?

Weber: Well, that comes later. Right?

Kocienda: Well, that comes after Safari 1.0.

Hsu: So we shipped Safari 1.0 and then-- so the next version what was-- what did you need to work on for the next version?

Williamson: So, my perspective is editing. But also, it was pretty clear that, relative to other browsers, we weren't feature capable. So, this is where Dave Hyatt really comes in. And--

Weber: What were you missing?

Williamson: A lot of the CSS extensions. So, CSS is really fundamental to how HTML works. People think that it's kind of a thing on the side. But really, all of HTML and the DOM really is defined by CSS. And the CSS implementation in KHTML was not great. And Dave was responsible for pretty much re-architecting that implementation. So-- and then performance was still an issue.

Kocienda: It's performance and just compatibility. You have to QA the whole web. I mean it's just like what is the proper rendering for a webpage? It's whatever the developer thinks. Whatever the developer says, "Oh yeah, I put this code, and style, and script--"

Williamson: Or the other answer to that question is what the other browsers do.

Kocienda: Yeah, what do the other browsers do?

Williamson: So, even if the other browsers do it incorrectly, there's an expectation--

Kocienda: That's it.

Williamson: That's how it should be.

Kocienda: That's it because it really is is a-- what does that person see in their browser when they hit the button to go from their staging server to their production server. So, whatever crazy browser they're using, I mean whatever broken version of Internet Explorer 6 on Windows, whatever, that's the standard.

Williamson: Right. And we'd unleashed this new browser onto the world. Developers weren't looking at our browser and designing their sites to work with our browser. They were designing it to work with IE on Windows or Mozilla. So, we were always playing catch up.

Kocienda: Yeah.

Williamson: Until much-- I don't know what version, it kind of shifted. And people started implementing stuff for Safari.

Kocienda: Yeah.

Williamson: But for those first few releases, it was--

Kocienda: Banking sites were horrible. Just people couldn't log in to look at their bank account. So--

Williamson: Forms.

Weber: Oh, just not working, okay. Forms were-- they can be tricky, yeah.

Kocienda: Yeah.

Hsu: What did you do to support Dashboard?

Williamson: Yes so, this is interesting. The-- so, getting back to this idea that the web potentially could support what looked like native applications as well as native applications, Dashboard was kind of the first manifestation of that idea on OS X. And there were a few things that were missing, that I thought were missing. And John Louch-- so, John Louch was the lead on the Dashboard project. But he and I both

thought that we could use WebKit to build these little widgets. But there were-- the critical missing piece that we both thought was absent was the ability to dynamically render content in the same way that you could with something like CoreGraphics. So, we added the Context object, which lets you render 2D, and now 3D, elements inside a DOM component. So, you can get pretty much exactly the same performance you can out of native rendering on a webpage using the Context object. And so, the widgets took real advantage of that. That's why they looked really great, and they were dynamically rendered. So, that's that project. It kind of gave us a glimmer of hope that yeah, maybe this is possible that we could actually add enough to WebKit to make native applications. But it was just a kind of a first stepping stone. And one of the-- my personal struggles with the web is that, getting back to this story of COBOL and accretion of crap on top of crap, web standards are so-- they've evolved so crazily, not with a single architecture in mind, but just over time, that it's possible to make really great things with web technology, but it's also very easy to make crappy things. And there isn't really a clear path that guides you in the right direction. Nowadays, there's lots of JavaScript frameworks that try to do this, that give you the best of the web and hide the worst. But still, it's very easy to make something that's fragile. And so, with the Dashboard, we tried to prove that you could make really good applications using just web technologies. And I think we accomplished it. But the widgets, they didn't have menus. They didn't have most of the services you expect from a native application. So-- but still it gave us a glimmer of what could be. And that kind of led to what we did when we first released the iPhone and the developer story there, which is another story.

Kocienda: But this is, you know, an interesting point. I'll just throw this in here, maybe-- I don't know if we're there or not. This idea of you're making good frameworks, providing the right levels of abstraction so that people can get the jobs done that they want done, developers in particular, and this idea that-- again, something that I think Apple is very, very good at is putting out toolkits, frameworks, that are hard to use wrong, hard to use wrong. And the web simply has not done that over its history. You just get this mess of standards that doesn't really give you that path to get through to the good stuff. If you're clever enough to cobble together the right pieces in the right way, you can make wonderful things. But again, what Apple has done is provided that point of view, or that curation, that experience. One of the best things that came out of WebKit, why WebKit I think is as successful as it was as a tool kit for browsers, is that we used it ourselves to make the browser. We were the first clients. And we were going to be the most particular clients with the highest needs for performance, needs for highest performance, highest fidelity. We were a web browser. So, if you were going to be using WebKit to maybe show an occasional webpage in your app, or in your About Box, or for a form, or something like that, you were going to be fine if you needed to load a couple of URLs here and there because we were loading dozens and dozens of URLs for every webpage typically.

Williamson: I remember the-- one of the keynotes. We did a demo where in like two minutes using Interface Builder, I built a web browser. It was fully functional, using WebKit. <laughter>

Kocienda: Yeah, yeah, yeah, yeah. Yeah, yeah, yeah. Yeah, on that side of the-- on that side of the stage and--

Williamson: I remember I was very stressed because I had put together the Interface Builder palette to make that work like a couple of days beforehand. And I wasn't sure if it was going to work. And I remember at the demo table, there were two tables that abutted each other. And the mouse pad was over the crack between the two tables. And there was a slightly different height difference in the two tables. So, as I was doing the demo, I was trying to move the mouse. And it kept hitting this little kink.

Kocienda: Man, it's always something, right?

Williamson: It's always something.

Hsu: I don't know if I want to keep going, but I guess I'll just-- I'll just ask a question that--

Kocienda: Yeah, we'll just keep going. You're going to wind up with tons of material anyway. You'll have whatever you need.

Hsu: Was there-- I mean so, WebKit was open source from the beginning, right?

Williamson: No, not from the beginning.

Kocienda: Well--

Hsu: Oh, not from the beginning? Or was it intended to be open source from the beginning?

Williamson: Go ahead. You can probably tell this better than I can.

Kocienda: Okay, well, it had to-- while we were developing it, we were developing it basically for eighteen months in secrecy. But because of the license, because of the license on KHTML, we needed to publish our source code when we made a product available to people. When we made the end product available, we needed to publish the source. But basically, what we did was we just balled up our software and threw it over the wall. There was no source code repository.

Williamson: Quite literally, it was a tar ball.

Kocienda: It was a tar ball. We just put our tar-- so, we conformed with the conditions of the license because we had to. That was the conditions of the license. But there was no developer community around-- there-- It wasn't an open source project, as such, in the beginning. Of course, it followed on and eventually became that, and that now all of the check-ins are available for view all the time. And there's

people all over the world that contribute to WebKit. But at the beginning, we just conformed with just the letter of the license, which is the source code for our changes needed to be available whenever we made a product available. So, we did.

Hsu: So, when did you actually start to set up that infrastructure, or--

Kocienda: Well, we got quite a bit of negative feedback from-- it was interesting. I mean it was mixed because on the one hand, I think some KHTML developers said, “Wow, boy, Apple’s really investing in this technology. That’s great.” But on the other hand, they said, “Well, you didn’t really make it easy for us to incorporate your changes.” And so, yeah, there was this really “wow, that’s great/gosh, this is terrible” right from the beginning. And so, I think it was maybe a year or two after that. I really wasn’t involved in the open sourcing effort. I don’t think you were.

Williamson: No, that was Darin mostly, I think.

Kocienda: Darin and Maciej, I think mostly.

Weber: So, you talked about adding editing to Mail. But did you-- so, and then you actually implemented it.

Williamson: No, no, that was all Ken.

Weber: Okay, sorry. And did you guys think about adding authoring to the browser at all?

Kocienda: No. No, to make a webpage authoring app?

Weber: Okay.

Kocienda: No, we never considered that in any significant way. No, this was all about solving the problem of HTML mail. That was the idea behind it.

Hsu: There was the iWeb app, later on, right?

Williamson: Yeah, that’s right. There was, kind of around the same time, some discussions, not within the core Safari team, but with other teams, about what it would take to build a webpage authoring tool. And that ended up being done by a different team.

Kocienda: Yeah, which is-- it actually even slipped my mind. I mean that's how far-- I mean I remember it now that you mention it. But it was just so far out of our core effort in the engine and on the Safari WebKit team that, yeah--

Williamson: Well, yeah.

Kocienda: I had nothing to do with it.

Williamson: Yeah. We did meet quite a bit with the other team.

Kocienda: You did. I didn't.

Williamson: Yeah. But it was-- as far as the HTML editing worked, that was not really applicable to web authoring. It was really-- I want to-- you know actually, kind of the recursive idea of using text editing in a form when you render-- use little instance of the WebKit engine inside the form, it's kind of very geeky, but a testament to how powerful the editing implementation is. So, when you edit text inside a form, it's actually like a little HTML view, a web view, inside the form.

Kocienda: Well, you see, but this idea then came out later is that like the Notes app on the phone, the first iPhone, the Notes app was a little webpage.

Weber: Yeah, that's what-- I mean it seems like doing--

Kocienda: It was a little webpage with an insertion point blinking in it. It was-- so--

Weber: Right, I mean once you have the general capability, Mail is what you did it for, but I guess I'm wondering both a genuine like browser editor, but that's kind of a particular use, but also as an across the operating system way to edit things in all sorts of applications.

Williamson: But the HTML editing that is core to the engine really isn't about rearranging DOM elements so much as text editing.

Kocienda: Yeah.

Weber: Okay.

Williamson: That was kind of the critical piece was the text editing.

Kocienda: The critical piece was to make it work like a text editor such that a person running the old version of RTF-based Mail, when the OS updated, and they changed over to the HTML mail engine, they wouldn't be able to tell the difference. It would just work. If you never received or sent HTML mail, you could just go and type out your mail messages, highlight the occasional word, make it bold, send it off, and you'd never know that it was a webpage that was doing that. It was-- yeah.

Hsu: Shall we move on to the phone?

Williamson: Okay.

Hsu: All right, so--

Williamson: In between this and the phone actually, it was a lot of tedious work just to refine the engine, get it better, add more compatibility. And then Dashboard actually was one nice diversion. But pretty much it was just the grind after the first couple of releases to try and catch up with IE, and--

Kocienda: Trying to QA the web.

Williamson: QA the web, yeah. So, then we're at the phone.

Hsu: Okay. So, then how did you guys get recruited to the phone project?

Williamson: Should I go first?

Kocienda: Go ahead.

Williamson: So, as I said, it got to be kind of a grind working on Safari after we got the core engine up and running. The browser was pretty decent. Performance was pretty decent. So, I was thinking of moving on to something else and actually was very close to leaving Apple and going to one of the other big technology companies. And so, I told Don. And then Don told Scott. And so--

Weber: Why were you thinking?

Williamson: Why what?

Weber: Why did you want to leave?

Williamson: Because it was kind of a grind working on Safari. It had become-- it wasn't real exciting anymore.

Weber: But you could have moved to another project within Apple.

Williamson: Could have done, yeah. Well so, it was time for me to move on. I'd been there for several years. And the other prospect was actually pretty darn good. And--

Hsu: Did you actually have an offer?

Williamson: Yeah. So, it would have been very interesting how my life would have been different if I'd gone down that path.

Hsu: Was this Google?

Williamson: Yeah. So-- so, Steve-- so, Scott talked to Steve. And then Steve called me into his office. And I also-- were you in the meeting, too? Oh, sorry. Me and three other-- I think three other people on the Safari team were also thinking about leaving.

Hsu: Wow.

Williamson: So, it was kind of a crisis point for the team as far as the engineers. And Steve sat us down and--

Hsu: All of you together?

Williamson: Yeah.

Weber: But all to Chrome, or?

Williamson: I don't want to talk about the specifics.

Weber: Okay.

Williamson: So, Steve said, “I really don’t want you guys to leave. I’ve got this great project that I’m working on. And I want you guys to work on it.” And he said, “It’s a phone.” And he said, “I want it to be a touch screen phone. And it’s going to be connected.” And I said, “Well, the display technology’s not there. The compute capacity is not there. The battery is not there. This thing-- it’s not going to work.” And he said, “Yeah, it will. Don’t worry about the cellular stuff. That’s going to be solved. The display technology is great. Trust me.” And he said, “I want to show you a demo.” And so, it wasn’t at that meeting we got the demo. But a few days later, I got a demo from Bas, actually, Bas Ording, of some of the work he was doing on a very fast computer with a big display, of some of the touch interfaces that he was prototyping. And it was pretty darn awesome. And it was really just a scrolling list at the time, but kind of the physicality of it and the fluidity, it just looked like something underneath the screen was really moving.

Weber: But this was on the projection table?

Williamson: No, no, this was--

Weber: Or after that.

Williamson: This was after that.

Hsu: So, there was already a tablet prototype at this point?

Williamson: So, we had tethered units that were basically like a second display. But they were-- at that point, they were super crude. And so-- I was convinced, though, after seeing that and decided to, okay, yeah I’ll do this. So-- but we were wrapping up, I forget which release, six, five or six. We were wrapping up a release.

Kocienda: 2005.

Hsu: 2005? Okay, that’s 10.4, Tiger.

Kocienda: 10.4 four, yeah.

Williamson: So, I stayed on working on OS X and WebKit up until that point and then, when we shipped, switched over to working with Bas and Scott Hertz to see if we could, in my classic fashion, hack something together using whatever we could find to get something which was a better proof of concept. And we actually used these prototype ARM boards, little chip boards, off-the-shelf stuff. And we were just hacking away and kind of getting whatever pieces of hardware we could find to try and get something to run. And we had Linux up and running. And we did something on top of Linux. And it was pushing the

envelope for what was possible. But there was the hope that we could do something. So, we eventually got a demo running that was good enough to convince Steve and Scott to say, “Okay, let’s build out more of the team.” And we brought on-- you came on. And we brought Chris on. And we started fleshing out that group. But then there were a whole series of other kind of meta-arguments and discussions going on in addition to the hacking on the prototypes. A lot of this is already public. We-- I and Scott, we fundamentally believed we could take OS X and strip it down and get it running on a very low-end ARM processor. And Tony Fadell had a whole other team in parallel working on something to basically evolve the iPod software into something that could run a phone. And we believed, and I still do, that the phone is not really about the phone. The phone is a computer. It’s just a much smaller Mac in a smaller package. And the phone is kind of an irrelevant side aspect. It’s just a cellular modem, which is off-the-shelf, you can plug it in. So, we had a bunch of fierce arguments, though, about whether we should go OS X versus the iPod stack. I could spend a lot of time talking about that. We also had fierce arguments about web versus native.

Hsu: Right.

Williamson: So, there were-- we can go in different directions.

Hsu: I’m actually interested in both of those meta decisions because I’m really interested in all the possible paths that this could have taken and, ultimately, why the decisions were made to go the way that they did. So, yeah.

Williamson: So, the first big decision was, is this going to be OS X or is this going to be based on the iPod software? The iPod software is reminiscent of going back and looking at Mozilla, you know, or that big stack of COBOL code. It’s just a bunch of cruft. So, the iPod ran a single task. And there were like subtasks that could be run with it, not larger tasks. But it wasn’t a modern operating system at all. And for each individual functionality, there was an assigned range of memory. So, you couldn’t go outside of that. It was very primitive. So, if the vision was we’re going to build a computer, iPod was not the way to go. It was- they were pushing the limits just to get what functionality they had in the somewhat functionally-limited iPod.

Hsu: It’s almost like going back to the original Mac architecture.

Williamson: Yeah, kind of like going back to pre-OS X but worse. So, but their vision really wasn’t to build a general-purpose computing device. Their vision really was, let’s add a phone to an iPod, and-- which is kind of what Steve wanted, too. But Steve also, I think, saw the potential for something beyond that. And what I saw was a web device. I want to run the full-on web on a phone and have it with me all the time. So, eventually, we had a couple of bake-offs. And we did a demo, which ultimately was compelling enough to prove to Scott and Steve that yes, this is the direction to go. We had a meeting with Bertrand, and Scott, and Tony, and they gave us-- Tony backed off. And we got the greenlight to go ahead. And it was-- in hindsight, it was pretty clear the right choice. So, then we ramped up the team

significantly. And one of the-- one of our big tasks was, can we get the web running on this? And that's when we built out the Safari-- or the iPhone WebKit team. And so, we had-- the organization really was split into two pieces under Henri. Nitin Ganatra managed Mail and the phone application and contacts. And I managed the Safari team and the other kind of utility applications.

Hsu: What was the reason for that breakdown?

Williamson: Just too much work to do. So, we-- you know, I was-- I had a good background in web technologies and WebKit. And we knew that Safari was going to be important. And we knew Mail was going to be important. And obviously, the phone application-- Nitin came from the OS X team. He managed the Mail application, I think, on OS X.

Kocienda: Yeah.

Williamson: And so, it was a pretty obvious division of labor. And I'd done-- I'd helped John with Dashboard and a lot of the widgets on the phone, like Stocks and Weather and whatnot, were kind of like widgets in Dashboard. So, there was a big scramble to get everything ready. So, transitioning to the decisions about WebKit versus native--

Weber: Sorry, one last question on the-- the-- by bringing in people from the Safari team for iPhone, it's not a surprise that you would have had a vision that you wanted a browser on a phone. Was there any purpose to that? Hiring people from the Safari team into that just guaranteed they were going to try to make a more general-purpose phone, right?

Williamson: Yeah so, the--

Hsu: And that was Scott Forstall's--

Weber: I mean do you think they set it up, in a way, to see--

Williamson: Well, we set it up that way.

Weber: Okay.

Williamson: I mean we really wanted a web browser on the phone. And so, as far as the actual hiring, we couldn't really get people from the outside because--

Weber: But I'm saying by bringing you in, that's almost, you're putting someone who you know is going to go for a browser on the phone.

Williamson: That's right. But you know, I'm kind of a general-purpose hacker, too. So, I'm good at kind of getting stuff up and running. But also, the browsing is fundamentally important to the phone. It was. But kind of this clear-cut vision that Steve presented when he launched the product of three things, that wasn't always the case. It kind of evolved as we realized what was capable-- what we were capable of doing. And so, it was-- Jony Ive always talked about technology catching up with vision. And we were in that kind of process of, can we make this technology catch up with a vision that we have of a general-purpose computing device. So, yeah, it's--

Kocienda: See, now-- Add a little bit of color from my perspective on what Richard said and backing up even a second, too. So, Richard described how, when it came time to decide whether to move on, maybe go do something else, someone like him, I put myself in this category too, general-purpose-- We're not trained computer scientists. We don't have areas of research that we spent our whole career doing. We're hackers. We get stuff together. You want something like that? I say well, let's grab some of these pieces and knock it all together and see if we can make something out of it. And that-- Richard is, well, one of the best people in computing history to do that. I think the results speak for themselves. I don't need to say more about it. But, you see--

Williamson: Thanks.

Kocienda: So, the-- so, that's an aspect of it, is you get to version 5 of the browser, it's like okay. I mean it's kind of a grind. And so, this opportunity to go do something new, I think that fit his skill set and his interests more. But then he also had the vision. And so, Richard's vision is part of what wound up being what the iPhone became, how it-- how Steve went up on stage and said, "Hey, we've got three things here. We've got a music player, and a phone, and an Internet communications device." The Internet communications-- he had the vision to say, "We can shrink this web browser down and put it on a phone. And people can walk around with it wherever they are. Because you can have cellphone connections, they can have web with them wherever they are." And so, yeah, it was then that-- you wish you could say that you see it clearly from day one. But that's not really how visions typically are. You kind of have this gosh, maybe we can do this thing. Gosh, maybe I don't know. Is the technology going to be good enough? Well, let's keep pushing on this area and see if you can-- yeah, let's kind of cobble something together. And then, you know, the vision solidifies, sometimes really, really, really pretty quickly. But I just don't think it's that eureka moment. It's a slightly slower process than that. But it does take that willingness to kind of keep pushing in a direction to say, "Yeah, I think there's something there. I'm going to keep going this way. Rather than that way over there, I'm going to keep going this way." And that's what Richard did for making the web-- shrink it down so you can put it on the phone.

Williamson: The state or the art at the time was WAP. And I called it the baby web. And people were trying to create a dumbed down set of web standards to render these horrible webpages that were nothing like the real web. So, actually taking Mozilla, million and a half lines of code, putting that on a

phone, people thought it wasn't even possible. So, the idea of actually rendering something with the same fidelity on a small screen and making it navigable was really, at the time, kind of considered crazy.

Weber: So, had you looked at i-mode in Japan at all?

Williamson: So, i-mode is not the full web.

Weber: No, compact HTML, but it's a lot more than WAP.

Williamson: Yeah, but it's still not the real web. So, if you want exactly the same content, you want the New York Times homepage on your phone, you're not going to get that with i-mode. Or France had-- what was their system? There were a bunch of other systems that were dumbed down technologies. And the idea behind these technologies is, let's take as a given the technology we have and make things fit into that. Not, let's take what we want, and make the technology fit that. So, it's a different perspective.

Weber: And the best Opera browsers for Symbian or whatever were still not--

Williamson: Oh yeah, rubbish.

Weber: Yeah or ACCESS also made browsers for-- so, they were not rendering proper HTML?

Williamson: The other thing that is something that's often overlooked about mobile devices at the time is that the carriers actively transcode content, or did until the iPhone. So, if-- they would look at the pipe, see what's coming across the pipe, transform it into what they thought was appropriate for the phone, and then deliver it to the phone. So, when we negotiated the contracts with the carriers, with AT&T first, we said, "Look, we want a dumb pipe. We don't want you to transcode anything. Just deliver the bits straight to the phone," which was a huge, huge deal. And no other cellphone manufacturer had gotten any carrier to agree to that because, of course, the carrier's concern is bandwidth, bandwidth, bandwidth. We've got to do this. And now, they've reintroduced this with high definition streaming. So, they've kind of turned on transcoding. But they're kind of marketing it as something that's a positive thing. So-- but again, that's saying this is where we want to go. Let's change the technology or change the world to fit that direction. And Kim Vorrath actually was-- pushed really hard on getting a clear pipe. And we all thought that they weren't going to go for it. But they did.

Kocienda: They did.

Williamson: Yeah. And now, I'm not at Apple anymore, but at the time, all of our new carrier contracts had that clause that we had to have a clear pipe with no transcoding.

Hsu: Let's go back to the web versus native decision.

Williamson: Yeah. So, UIKit existed pretty much from the get go. We used that to develop our applications internally. And, just as we had with AppKit, the development of Safari and of Mail drove the development of UIKit. And we intentionally wanted to share components across applications. And the app developers fed into UIKit. But nobody thought it was ready for primetime. And Steve knew about Dashboard. And there was like an SDK around Dashboard that let third-party developers build widgets. So-- and there was a strong desire to have a developer story at launch. So, this isn't about the philosophical division of native versus web. It's about what actually happened. So, there wasn't a discussion about what was the right thing to do when we launched the iPhone as far as a developer story. Literally, days before the keynote--

Hsu: The January keynote or the WWDC keynote?

Williamson: Whenever it was announced that we were going to have this great developer story. That was WWDC I guess.

Kocienda: Actually, I think it was-- yeah, I don't remember.

Hsu: I think Steve announced at WWDC that there was going to be a web developer story.

Williamson: So, that announcement was not the end result of a process of discussion among the engineering team and Steve. That was Steve saying, "This is what we're going to do." So, it was a surprise to all of us that we had this great developer story. And so, we-- not because we didn't necessarily think that it was a good idea, but there's lots of differences of opinion of what the right thing to do is. So, we had to scramble, after the great development story pitch came out, to make the web a viable set of technologies. And it got really complicated because WebKit was open source, but we wanted to add proprietary extensions that were specific to the iPhone. But at that point, everything we did had to be open source. So, we did our best to actually make it work. But I think all of us thought that, on a powerful desktop platform, web technology was just not there to build applications. But on an impoverished platform with very little memory, very little CPU, and GPU, it was going to be tough. So, it was a challenge. But we did things like we added touch extensions, so you could have a touch GUI. But pretty quickly, it was obvious that it wasn't a great story. It wasn't a great developer story.

Hsu: By pretty quickly, like how many months?

Williamson: Oh, I don't know.

Kocienda: Not long.

Williamson: Not long at all. And as far as-- you asked the story about web authoring. As far as providing really good tools to generate web applications for the iPhone, we didn't have all those great tools. It wasn't like you could use Xcode and do something great. So, a great developer story has all of that. It has the frameworks. It has the IDE. And so, it was a pretty weak developer story. And I think, in hindsight, if there'd been more discussion, we probably would have held off in making that announcement. But there was a-- I think after we launched-- between the time we launched and WWDC, there was a lot of outcry-- or not outcry but request for a developer story. And we had to just come up with something very quickly.

Kocienda: Yeah, I mean during the original-- during like the pre-1.0, from inception to 1.0, we were so busy just trying to get the iPhone done that we were not thinking about a developer story. We just weren't. We were frantically just trying like-- going as quickly as we could to get everything done, not only to get everything done but to figure everything out. How does a touch screen user interface work? There's no mouse pointer. How does that work? There's no cursor. Well, how does-- how does that-- do we have-- do we show your touches, or is that just invisible? It's like all these contin-- What about multiple touches? What if you've got-- if you think about event routing or whatever, what if you've got two views on the system, and you land two fingers? Then there are two different views. How do you route those touches? All of these kinds of questions were things that we just-- we had to figure out. We didn't know.

Williamson: Yeah, we kind of skipped over that whole period of developing those technologies. But the period of time between launching-- demoing the phone and WWDC, we still had a ton of work to do because the phone that was demoed was overclocked. It was tethered. And Internet was provided through the tether, not through Wi-Fi. It was some smoke and mirrors in a way that Apple typically doesn't do that. But this device was so jury-rigged. It just barely made it. And to go from that to something which we were going to ship took a ton of work.

Kocienda: Yeah, it's kind of interesting you say because the tricks were pretty severe. But they were also pretty limited.

Williamson: Yeah.

Kocienda: In that-- it's like when you bring-- Steve brought-- in that original iPhone demo, when he brought up Safari. Safari was just running okay. It was getting its bits through the tether. But that was the real Safari. I mean it's-- yeah, it still needed some work. There were some bugs. I seem to remember there was a crashing bug in the thing. You know, that one out of every ten or twenty times or whatever, when he loaded the New York Times, it crashed. And so, I-- was it going to crash-- I don't know. So, it was like rolling the dice. Of course, it didn't crash, and it was fine. But the-- there were tricks. But I don't want to have it go that far. A lot of the software was--

Williamson: Yeah, it was real.

Kocienda: The real thing. It wasn't completely smoke and mirrors. But there were some pretty big--

Williamson: Mostly around the hardware, we had to juice the hardware to get it to work. But prior to that, I think there's a lot of interesting stories about the development of UIKit, and touch, and the keyboards.

Kocienda: Sure, yeah.

Williamson: And--

Hsu: Yeah, I want to get to that. I have-- or I want to get to that. But I did want to-- what I did want to ask about, like both Scott Hertz and Nitin had told me that the native versus web decision, that there was actually a decision to be made about doing the internal apps all in WebKit.

Williamson: Yes.

Hsu: Versus native.

Williamson: That was much earlier.

Hsu: And so, that-- I wanted to ask about that--

Williamson: Yes.

Hsu: Inflection point.

Williamson: Yeah, so really Scott kind of set us up. Me and-- Scott Forstall set Scott Hertz and I up as kind of the two advocates of the different positions. And obviously, I was advocating the web approach. And Scott Hertz was advocating the native approach. And I tried to make an argument for that position and-- because we were coming off the Dashboard experience. And so, we implemented a lot of the components in both. The simple components like Weather and the scrolling address list, and-- I actually got it working really well using web technology. So, the performance was pretty good. But I think it-- I eventually backed off and acquiesced and said, "Yeah, I think native's a better approach." But the reason for that wasn't because the underlying technology wouldn't have worked. It was kind of what we talked about earlier in that it's possible to create really good performing web applications. But it's also possible to make poorly performing web applications that are very fragile. And, as Ken said, a lot of what we do at Apple is create frameworks to make it easy to do the right thing and hard to do the wrong thing. So, ultimately, that's what the decision came down to. We had a small team of people. And we had to do an

awful lot of work very quickly. And using a traditional kind of Objective-C framework similar to AppKit could get us there much more quickly. So, I did a best effort to try and prove that it could be done. But ultimately, I agreed that UIKit internally was the best way to go.

Hsu: So, it wasn't just an issue of the web solution would have been not performant enough, like too much memory or--

Williamson: No.

Hsu: It could have been possible to do it in a high-performance way.

Williamson: Right, and so, a lot of the work that we actually did there to make it perform well actually translated into optimization of the web engine in general. So, things like hardware accelerated transforms on layers ultimately made it into the CSS implementation on desktop. So, John Harper, again, we've mentioned him before, was brilliant at trying to provide the right abstraction so we could use the GPU to composite layers and get the kind of performance we needed. So, ultimately, when-- like the scrolling list implementation. When we scrolled the list on the version of WebKit on iPhone, we were using the GPU to move that up and down. So, the performance was there. But we had to do a lot of top to bottom kind of optimizations to make it work, which paid off, ultimately. But that wasn't the key. It wasn't about performance. It was about developer productivity and how rapidly you could develop an application.

Hsu: There is one more thing that I want to try to clear up, which is that Nitin told me that the various utilities like the Stocks and the Weather had been written in web technology but that, before the phone shipped, your team rewrote them in native?

Williamson: Yes. Yes.

Kocienda: Before we'd shipped? No.

Williamson: Oh.

Kocienda: Web version shipped--

Williamson: Yeah, maybe.

Kocienda: Web version shipped. They did.

Hsu: Oh, so they actually did ship.

Kocienda: They did ship. I think for-- I think they were maybe rewritten for the third version, maybe the second version, but no, they shipped.

Hsu: Oh, the later version, in iOS 2 or 3.

Kocienda: They absolutely did, yeah.

Weber: But so, which ones shipped as web?

Kocienda: So, it was Weather, Stocks, Clock,

Williamson: Not Clock.

Kocienda: Not Clock?

Williamson: I don't remember, but--

Kocienda: Weather and Stocks definitely.

Hsu: But the ones that your team owned?

Williamson: Yes. So, but, again, that was-- we did that as an exercise of proof of concept that we could do it. And I think ultimately, it was the right decision to switch to UIKit. But like I said, we pushed as hard as we possibly could to get those things to perform well. And I don't know what version we switched.

Kocienda: Yeah, I don't recall exactly, but I'm--

Hsu: But the switch to native was-- to UIKit wasn't until later versions of iOS?

Kocienda: Later versions-- I am pretty sure that the web versions of Stocks, and Weather, and maybe one or two more which I just don't recall. But they shipped as web apps on the first phone, yeah.

Weber: And how much did the internal decision for what you were producing affect the later decision for the developer community? I mean obviously--

Williamson: Oh, a lot, clearly a lot. I mean it was-- if we couldn't be productive with these technologies, and we know the engine intimately, then how could external developers be productive? So, yeah. Now, having said that, I still have a lot of hope for web technology. And I think, with a lot of these new JavaScript frameworks-- not even new, but JavaScript frameworks, it's possible to write good web applications. If you look at some of the work that-- for the iCloud apps on the web, they're pretty darn good. But you have to have really good developers that understand the technologies really well.

Kocienda: When Apple works at its best, it's because it owns the whole stack. And when we have a vision for what we want to do, how we want something to work, we can go down as far in the stack as we want to make sure that that experience, that vision, can be delivered.

Williamson: That's the stack from the silicon all the way to the bits on the screen.

Kocienda: Certainly, the silicon-- In the old days, I would think that we could go all the way down to the kernel, right, when we were on the Mac, and when we were on the iPhone. Hey, we can go all the way down to the kernel. We could change. We can add a system call to the kernel if we need to. Not that we ever really did, but that we could if we needed to. And all throughout the period of Safari, and WebKit, and the iPhone, it was so often making the terrific experiences, was going down, dropping down a level, sometimes two, to make sure that the supporting technology was there so that the high-level feature was really, really good, delivered on the promise, delivered on the vision, and that this is-- that link-up with the vision and the technology, that's what makes Apple Apple, is the willingness to do that. Not the political arguments, who owns what, I want to own this, I want to own that, this is my domain. No, no, no, it's all us. We're all together. We're all trying to make your best experience for the person out there who's going to go to the store and buy the thing. That's what the vision is. That's what we need to keep our eye on. And as technologists, we're the ones who can keep both of those things in sight, the vision and the supporting technology to do it, to make it happen. And so, that's, to me, the secret sauce. If there's a magic, it's that willingness to and that ability to actually develop a vision and then develop the software to deliver on that vision.

Williamson: Right. When Apple works well, that's how Apple works. It doesn't always work well, though.

Kocienda: Yeah, it doesn't always work well. And-- yeah.

Williamson: But I think when Apple works well, it really is like a collegial kind of atmosphere. And there aren't boundaries between, say, the camera team and the apps team or even the GPU guys. It's--

Kocienda: I think you and I have had those experiences where we've gone to people and said "we really need your part of the stack to do this thing." It's like iOS 7 blurs. Blurs on iOS 7. I went to John Harper

and said, “John, you know.” I’d say, “You really need to do blurs.” And he’s like, “No, it can’t be done. No, it can’t be done.” And of course, then it’s John Harper. He goes away for a day or two and comes back and says, “Yeah, it’s finished.” But I mean-- and so, that--

Williamson: That reminds me of a story that-- this came up often with the development of the iPhone. We were-- Steve would say something that he wanted, or Scott would say something he wanted, or the UI team-- the design team would say something they wanted, and we would say, “No, you can’t bend the law of physics. You can’t bend the laws of physics.” And Scott would say, “I never want to hear that phrase again. Bend the laws of physics.”

Kocienda: Bend the laws of physics.

<laughter>

Kocienda: Just do it. So many of these things are just not taking no for an answer. We’re going to make this vision. We’re going to deliver it. We’re going to make it happen. And we’re going to go down as far, as many levels as we need to. We’re going to go, and we’re going to ask people working at those levels to help us out. And again, when Apple worked really well, you could just go to another building and another team that worked under another VP or whoever it was in whatever organization. And they’d go, “Oh yeah, you really need that thing? Oh, okay well, I never really thought about that. You really need that thing? That will really help you? Wow, okay.” And there you go. You’re off and running.

Hsu: Well, let’s go back and talk about developing some of that stuff. So, we could-- there’s a lot of places we could start. We could start with the keyboard. That’s one of the big things that you guys did. Could you talk about that and--?

Williamson: I can talk about the process. Maybe you can talk about the-- whatever happened.

Kocienda: Sure.

Williamson: So, you might have heard this from Nitin and Scott too, but it became one of the critical things that we had to solve and solve well. And there were really a couple different schools of thought among software developer geek kind of guys. We thought that the most important thing was efficiency of the keyboard. And if you had to learn a little bit, okay. The other school of thought was you don’t need to-- we want to make a keyboard where you have to learn nothing. You just pick it up and know how to use it. But it was pretty clear that the screen was too small to have a good keyboard that looked like a traditional keyboard. So, pretty late into the project, we decided to-- as a team, we were going to take everybody off what they were working on and just focus on making keyboards to try and make-- come up with some creative ideas to do something radically different to make a keyboard that would work.

Hsu: By pretty late, like what timeframe?

Williamson: Always with the time. I--

Kocienda: This is burned into my brain, so I can actually-- this was the end of 2005.

Williamson: Okay.

Kocienda: So, it was just a little bit more than a year out of Steve heading out on stage saying here's the product.

Williamson: So-- and we had a couple of kind of basic precepts that we wanted to follow when we designed the original phone for the original screen, which was the hit region had to be thirty pixels on that screen. Otherwise, it was too small to hit. And you just couldn't fit enough keys on a traditional keyboard to make that work. So, the team went off, and we came up with all kinds of radically crazy keyboards. We had cord keyboards, and bubble keyboards, and all kinds of crazy ideas that all ended up being patented I think. But Steve saw them and said, "No, no. You need to be able to go into the store, pick up the phone, and start using it immediately with no learning curve at all." So, that means a QWERTY keyboard. So, then the challenge was okay, it's got to be QWERTY, how do we make it work. And this is where Ken came in and made it work.

Kocienda: Yeah. It's--

Hsu: So, there was a contest, right? You won the contest.

Kocienda: Yeah, there was a contest, yeah. So, like Richard said, there was a just a bunch of people that went off and started making keyboards, trying to come up with something that would work well. And so, well yeah, I came up with the best option. So, I won the contest. And then I had to make it work.

Hsu: What was it about--

Weber: Sorry, there was a contest which could be non-QWERTY keyboards, as well.

Kocienda: Yeah.

Weber: But then there was--

Kocienda: There was-- it was really-- there were a couple of threads, as Richard was saying. There was clearly pressure from Steve to say we need to be able to pick up this thing. And people need to know how to use it. But we didn't have that. We didn't know. We didn't know how we could possibly get that done. And so, it's actually, my winning version was a keyboard
that had the letters laid out as Qwerty, but letters were joined up on multiple keys. So, you could actually go and look at it and type like you were typing on a Qwerty keyboard. But the keys were ganged up. So, the Q-W-E letters were--

Weber: Invisibly, you mean.

Kocienda: Visibly. There was actually a Q-W-E key. There was an R-T--

Weber: Like an old telephone, like a nine-key or something--

Kocienda: Yeah, there was a Y-U-I key.

Williamson: We actually looked at the-- what do they call the nine--

Weber: The nine-key telephone, yeah.

Williamson: T9 keyboards, we looked at T9 keyboards.

Kocienda: So, the-- so, yeah-- so, then the-- so, but you were-- it was really pretty subtle. So, you could try to ignore that there were these large keys that were easy to hit. And when you hit the Y-U-I key, those three letters just went into a bucket, right? Instead of just one letter, you got three letters. And then you'd wind up with this stack of multiple letters. And then the software went and tried to figure out well, if I took a letter from this group, and a letter from this group, and a letter from this group, what words could you make with that. And that was the first--

Weber: And it would suggest the way it does today.

Kocienda: It would suggest, yeah. There was a suggestion bar. But from the very, very early on, when you hit space, it would choose the best suggestion. If you wanted to choose a second or third selection, there was a suggestion bar much like there is-- I call it a suggestion bar now. It's a QuickType bar.

Williamson: It's really interesting how many of those ideas have come back.

Kocienda: Have come back, yeah, amazing. I'll go on the record as saying this, I think that QuickType, I don't think it works.

Williamson: Now even like the versions of the bubble keys are back, so you can pull down on a key to get the alternate letter.

Kocienda: Oh, yeah really?

Williamson: Yeah. It's iOS 11.

Hsu: QuickType is a feature in--

Kocienda: QuickType is basically the suggestion bar, the bar above the keyboard that suggests words.

Hsu: Oh, right, for autocorrect.

Kocienda: Yeah.

Hsu: For auto-- I mean auto--

Kocienda: Auto-completion.

Weber: Predictive auto-completion.

Kocienda: Yeah, auto-completion, yeah people don't use it. I don't think people use it. Anyway, so--

Weber: Right, yeah you get three at the top, something like that.

Kocienda: Yeah, three at the top, yeah.

Williamson: We had that in 1.0.

Kocienda: We had that-- well, in pre-1.0 but then took it out.

Williamson: Right, right.

Kocienda: Pretty late in the game because Scott went to me and challenged me and said, “Yeah, this autocorrection thing is pretty good. You need to make it better so that the best autocorrection is always the one people want. So, go make that work.” And so, again, that’s one of those things where it’s just the vision came. And it wasn’t my vision. It was Scott’s vision, but then I just went and figured it out because that’s-- we had to make the technology rise up to match the vision. So, that’s why we wound up with the little in-line bubble right under your typing.

Weber: But then when did the ganged keys go away?

Kocienda: Well, there are some problems with ganging up letters on keys. How do you type something that’s not in the dictionary? How do you type a specific key? What if I want to type a nonsense word? What if I have a friend from some foreign country who has a last name that is not in the onboard dictionary? How do you type that? And so, well, you can’t because you can’t choose a specific letter. And so, eventually, we-- we made the decision, it’s like okay, well look, we’ve got to divide up the keys. But then I had the idea to say well, okay, well why don’t I make just dynamic groupings of all of the letters. So, instead of Q-W-E being a locked set, Q will be with W-E and actually A underneath it. And W will be with Q and E, and, what, like D underneath it. And E will be with W and R, right? So, dynamic. But make that invisible to the user, just visible to the autocorrection software. To the user, they were just individual keys. You didn’t know that still, behind the scenes, when you tapped one of those original keys on the first iPhone, that a bucket-- it was the same thing as-- a letter, a bunch of letters went into a bucket for each space in the word. And then the autocorrection algorithm would go through and try to figure out what was the best word, what could you have meant, rather than what you did.

Weber: But how would you then still-- you still needed a way that the software knew you actually had hit “A.”

Kocienda: Yeah.

Weber: Okay.

Kocienda: So, what happened was there was-- I worked with Bas Ording on this quite a bit. And I worked with Richard on the keyboard. His office was right next to mine. So, every-- he was the first demo guinea pig, all of these keyboard-- basically, all of the keyboard ideas, I saw them first because I developed it. And Richard saw it second because he was right next door. And so, and then I started working with Bas, Bas Ording, a lot on look and feel issues and getting to the question of like what you’re saying about what key did you hit. We-- I decided, basically, that the letters that got popped up were pretty important. So, we came up with this idea that when you hit the letter, since the keys were so small, that they should pop up because this was one of the initial problems of the iPhone. The whole-- one of the big problems of the

whole user interface system was you would move your finger down to the display, and right as you got close, you're blocking up the thing you're trying tap, especially something really, really small like an individual key on the darn keyboard. But that had problems like with the back button, as well, or links on a webpage or whatever. Right as you're getting close, you're blocking the thing you're trying to touch. So, ways to work around that, really that physical limitation of, your fingers aren't transparent, the way we solved it for the keyboard was to make the keys pop up, so you could see them. And so, that added a biasing to the algorithm that I would try to give you the letters that you popped up because it's the-- it was part of the feedback system that you were getting. You're standing over there nodding your head right now. You're giving me feedback that you're hearing me and understanding what I'm saying. That's what key-- that's what the popped-up keys did. It let you know what-- that the keyboard was following along, what it saw as you were talking at it. And so, that biasing went into figuring out what path of letters got chosen from the ones that were in the neighborhood of the tap areas that you tap, biased a little bit by the actual keys that got popped up, biased by word frequencies in English, biased by words that you've tapped before, biased by words that are in your contact-- your address book, so on and so on. And so, that's-- you put all of those things together, and you wind up with an autocorrection algorithm.

Hsu: And you said that the thing-- the actual-- the QuickType thing that you had already prototyped got pulled out before the thing was released?

Kocienda: Yeah. because we realized that the best way-- this is something that Richard tested with me as well, as I recall, but that the best way to type was to just keep going, just keep going, tap, tap, tap, tap, tap, tap, tap, tap, tap, just keep going, and that if you st-- if you tap and then stopping and looking, no, the suggestion isn't there to-- tapping and looking at the suggestion to begin-- it's actually way slower than just keep going, just keep typing, and that the autocorrection algorithm needed to be good enough that it would just clean up all your mistakes as you went.

Hsu: Were you aware of the-- I mean this has been published in the recent book, but that Phil Schiller was against the virtual keyboard and wanted a physical keyboard? Was that-- did that figure into any of this at all?

Williamson: I never heard anything like that.

Kocienda: I never heard anything like that. I met Phil once and gave him a demo of the keyboard, and it was-- there was nothing like that.

Hsu: Okay.

Williamson: I don't know anybody that-- I mean I guess Tony said that. But I don't know anybody who could verify that. It seems to come out of left field.

Hsu: Okay.

Kocienda: Yeah, I mean it's-- yeah.

Williamson: Go ahead.

Kocienda: Go ahead, go ahead.

Williamson: I mean as far as input in any of the design, Phil-- he wouldn't take it on himself to do that. I mean that wasn't his role. So, he was certainly cognizant of what we were doing. But I've never had any experience where he would interject his opinion about a design issue.

Hsu: Hmm.

Kocienda: You know, to me, his role is like as a prospective customer and to somebody who needs to understand the technology. Look, he understands technology pretty darn well because he needs to be one of the main people to explain it to everybody out in the world. And so, one of the things that-- these projects were so small, the number of people who actually knew about these things was so limited, that the people like Phil who weren't involved, yeah, it was important to get their input on-- to verify whether this was an idea that might work well out for people in the world. But I agree with Richard, when it gets down to real content, never in my experience was it-- was my work changed in a--

Williamson: But having said that, there were fierce debates about the actual key-- I mean I don't think there was ever a debate about not having a virtual keyboard. But the debates were about what the virtual keyboard was going to be. That was a struggle right up until the very end.

Kocienda: The whole-- yeah, the whole idea of it was that the-- the whole screen was software.

Williamson: Right. It's very much like, you look at a Mac. You don't have a bunch of dedicated buttons on a Mac. Sure, you have a keyboard. But everything's virtualized. And the kind of the larger vision is ultimately, you just have a piece of glass. I mean the idea of the home button finally going away on the X, we were talking about that with the original version. So, putting a physical keyboard on was never really debated among the iPhone team. That was never any discussion I was a part of.

Kocienda: No, no. I mean it's like the keyboard needs to be there when it needs to be there. And then when it doesn't, it should-- it must go away.

Weber: Given that sounds like it was fixed, but in terms of whether it needed to be QWERTY and familiar, I'm wondering, the Palm was-- with Graffiti, was an example of how you could actually train people to learn a weird system.

Williamson: Could you? Is the Palm still around anymore?

Weber: Yeah. No, but the fact it succeeded at all.

Williamson: Well, so we talked about--

Weber: What I'm saying is--

Williamson: We talked about that a lot.

Weber: How did you think about-- because that was a strange example.

Williamson: We talked about the Palm. And we also talked about the Newton, and both of which you could argue were failed products. But I don't know how many times we'd say we don't want to build another Newton.

Kocienda: Yeah, a lot.

Weber: I don't mean specifically the text recognition, but the idea that you could get people to learn a bit to input more factors.

Williamson: It's just-- we-- I'm sure we all could. But you want grandma going into the store, picking up an iPhone. She's seen a QWERTY keyboard her whole life. She knows immediately how to use it. There was zero learning curve. And this is really Steve. I mean he just drove this and drove this. It's got to be something so simple anybody, anybody can use it.

Kocienda: Yeah. Yeah, that was the challenge is that the work should explain itself. It shouldn't have to be this thing that you need to learn. Look, in order to use the iPhone, to go from whatever phone you had or no phone to the iPhone, you were going to need to learn things. And so, we didn't want to overburden people with the number of things that they were going to need to learn. They were going to need to now overcome-- because I mentioned it before, you-- we forget about it now because it's just-- we just bang away on our touchscreen smartphones all the time, that apprehension that you had about putting your finger down to hit a target that is going to be blocked by your finger was something we were dealing with as technologists, right, back in the beginning when we had those prototypes. And so, there were-- we

were going to be imposing on you. So, when and where we could take away something that wasn't going to be adding to that list of impositions, we did. We tried that. We went for that. We went for simple for work that would explain itself.

Williamson: When we finally got to carry phones, I brought one home. And I had my three-year-old daughter play with it. And she just picked it up, and she knew the swipe to unlock it. And she just intuitively could figure out how to click on things and scroll things. And at that point, I-- it was obvious that we had hit a home run. And she didn't really use the keyboard, but she could operate the phone. So, the phone's kind of not that way anymore. It's gotten a lot more complex.

Kocienda: See, I--

Williamson: But it's a natural evolution. You know, I think you layer on more functionality. But it's unfortunate. It's lost its original simplicity.

Kocienda: See, I think that the-- there are-- it's almost like the forces of entropy are at work. And-- but in an opposite way is that technology tends toward complexity. Right?

Weber: Creeping feature-itis.

Kocienda: Yeah. And so, making things simpler requires an absolutely constant, constant, constant effort. I mean this is something that Steve was absolutely brilliant at as an editor of-- the final editor of-- and the final arbiter on what was in the product and what wasn't. And he would just always, always, always drive us to make things simpler. And so, that was obviously a reflection of his vision for the company, the kind of products they were going to make. And people like Richard and me, that we see it that way, too, that we're going to try to make things simpler and to try to have that mind of a technologist, but then also the mind of the user, always that double view of how to make the product. Make the technology work, but then also make it simple enough that people don't need to-- have a PhD in computer science to figure out how to make it work.

Hsu: Can we talk about collaborating with Google on Maps and YouTube?

Weber: Do you want to do the-- we already talked about the text input, though?

Hsu: Okay. Maybe we'll hit that first. Ken, do you want to talk about the text input system on the iPhone? It was based on the WebKit/KHTML implementation?

Kocienda: Yeah, they're little webpages. So the text system on the iPhone, it's all based on the WebKit work to start building up from the initial work that Richard did to make really, really fast text rendering and

the work that I did to make editing work. We could only afford one text system on the iPhone. We couldn't afford to bring over the text system from AppKit and the Mac. So WebKit editing was it. If you saw a piece of styled text on-- well, if you saw a piece of text on the iPhone it was rendered with WebKit. It wasn't necessarily done with my editing system but it was all done with Richard's character blitting system basically. Once it got to editing, well there's two big parts. There's, well, three parts perhaps. There's single line widgets in UIKit. There's multiline widgets in UIKit. And then there's WebKit form widgets. And, of course, all of those are based on WebKit. And the two UIKit widget-- yeah.

Hsu: So UITextView, UITextField…

Kocienda: Are implemented-- they are little webpages. They were, up through, I think, through iOS 7.

Williamson: There's just one thing I want to interject here.

Kocienda: Sure.

Williamson: Coming back to this idea of abstractions. So all text used to be rendered with the WebKit text rendering engine and all text used to edited with WebKit. But because we had the right abstractions, that was completely hidden from the end developer. So we can shift out the whole engine underneath and applications that use those APIs don't have to change.

Kocienda: And that happened in iOS 7.

Williamson: Because the phones got more powerful. The capabilities were there to actually use a much more complex and feature rich engine than we needed originally.

Kocienda: Yeah.

Hsu: So did you actually take the AppKit text engine and put it underneath?

Kocienda: What in iOS 7?

Hsu: Yeah.

Kocienda: I didn't. Other people did.

Hsu: Right. But that is what happened?

Kocienda: But basically yeah.

Williamson: CoreText was ported from OS X…

Kocienda: Yeah, it was Dean Baird [ph?] and Aki Inoue were basically the ones who did it in iOS 7 but I wasn't involved in text or keyboarding at that point.

Williamson: Because there were features like, you know, which we thought were not critically important like ligature support and kerning and a lot of advanced features for really advanced topography. Nobody really missed them for the first seven versions of the phone.

Kocienda: You couldn't even select text in the first phone.

<group laughter>

Williamson: Right. You couldn't copy and paste.

Kocienda: You couldn't copy and paste. But yeah, so the text editing-- text entry system on the iPhone was just a bunch of little webpages made with some UIKit APIs that were very much text focused, very much based on what we had on AppKit so that people would be familiar with that and yeah.

Williamson: Getting back to this example of taking this kind of bucket of this technology and using what we can, that's just sufficient enough to get the job done.

Kocienda: Just sufficient enough. But there were a couple of little niceties. Because there were different kinds of keyboards so that from the very beginning I came up with this idea of text traits so that you could, on the field when you programmed it, you could say well, when the insertion point goes in that field, alter the keyboard in this way. You could turn off auto correction. Or you could make all caps like when you're entering in the state abbreviation in an address book field. So you get all caps. Or you could call for the-- when you put the insertion point in a field, you could call for the numbers keyboard to type in a zip code. So there were some innovations there to accommodate the platform. And, again, trying to think of things that people will want but then also developers will find useful. But yeah.

Hsu: Shall we go on to talking about the Maps app?

Williamson: Okay. So where should we begin? So very late in the development of the phone Steve had a carry unit. And I wasn't at this meeting but I heard about it. He showed it to Larry Page. And they had

some discussion about, wouldn't this be great if we had maps on the phone, which in hindsight it's obviously a good idea. And we had zero technology at Apple that was applicable to this. So the kind of the command comes down that okay we've got to figure out how to get maps on the phone. And go talk to the guys at Google and they'll help you. So I remember Henri and I went to meet with the guys at Google. And I think it was around Halloween so it must have been October 2004-- close to launch.

Kocienda: Six. 2006. I mean it was late, late, late. It was super late.

Hsu: Wow.

Kocienda: No, this was late.

Williamson: And we were literally a few weeks away from having to launch. And so it was like one of these, I've got to pull a rabbit out of the hat kind of moments. So we went to Google. And Jerry Morrison [ph?] was there. He was the lead for GMM, Google Mobile Maps. And I remember he was dressed in like a nun's outfit for Halloween.

<group laughter>

Williamson: And so we're kind of all sitting around the conference table and we were like "we have this thing. We can't really tell him that much about it but Larry and Steve want us to work together. And we'd really love to take a look at the GMM code." And Jerry said, "Oh, okay." And then we talked a little bit more about how GMM worked which was really designed for feature phones with tiny little screens. And it delivered these little tiny bitmap tiles. And we said, what if we had to actually pull down quite a lot more tiles than that and make the tiles a little bit bigger because the screen, even though it seems tiny now for a phone, it was pretty darn big. And he said, "Well, we could probably do that. But let's talk about it some more. So why don't you take the code and have a look at it." So we went back and immediately-- I hadn't been coding for a while but Chris Blumenberg and I said, okay, we've got to do this. We've got to build a map app. And we've got to do it in like a few days. And we have this crappy code that Google gave us that's in C++. And we've got to make it work on the phone. So we did. And it was similar to the original Safari Web KHTML kind of rush. And we got the client up and running hitting their servers. And it worked pretty well but it wasn't great. And so we went back and we met with Google again. And they said, "Well, we've been kind of monitoring what you've been doing from some devices and you're pulling down a heck of a lot of tiles, you know, what are you doing?" And we said "we can't really tell you." But can you make the tiles bigger? And can you support this load on your servers? And they said, "Oh, well, okay we'll look at that." And they did. They were really, really helpful. And this was all done without a contract. It was just a handshake. And so we were able to get something ready and it was-- they shipped in the first version.

Kocienda: Oh yeah.

Williamson: Yeah, it shipped in the first version so-- but it was another kind of herculean effort to build this client. And we in the first months after launch we just thrashed Google servers because they were used to delivering one small tile very slowly over pretty crappy pipe. And we were just saturating their servers. But they cranked up the capacity and it worked pretty well. And we had also a good partnership with them on YouTube. And John Harding helped us get a client running very quickly. We developed the client just like we did with the Maps app because we couldn't let them know what we were working on.

Weber: And they're all Web-based.

Williamson: Yeah. So-- well, no the Maps app and the YouTube app were not Web-based. They were native apps.

Weber: Oh, they were native. Even from the start.

Kocienda: Yeah, but pulling the content over HTTP.

Weber: Yeah. But I mean they were not WebKit. They were native.

Kocienda: The apps themselves, right. But all of the content to fill the app.

Williamson: Yeah, the original maps client that Google had, their state of the art was delivering images over HTTP. It was not sophisticated at all. It was a bunch of tiles that you stitched together. So yeah that's how the Map app happened very quickly. Then there's a lot more interesting stories. <laughs>

Hsu: Later on we'll get to that. Probably not today.

Kocienda: But, again, this is just-- this is how things happened at that time. You get this tiny, tiny collection of people together to make something that just goes out and becomes something that everybody just expects to be there and works. And is just this little step down what gets to be this long, long road.

Williamson: Yeah, it was a good time too. Google collaborated with us on the search engine for integration into Safari. And it was a happy time then. Apple and Google were best friends.

<group laughter>

Williamson: It didn't last.

Hsu: Yeah. And YouTube came in after the keynote. Right?

Williamson: Again, with these dates. I think it came in after.

Weber: I think. Yeah.

Williamson: Maps and YouTube came after. Yeah.

Weber: But maps had been-- I mean obviously it didn't get around to doing it until late but had there been talk of maps for a long time or not?

Williamson: No, no. It was very late. The whole idea was late. It's kind of shocking that it wasn't part of the earlier planning.

Kocienda: And I remember it because I wasn't involved with it. I had my hands completely full with keyboard and text but I just remember because, again, [his] offices were right next to mine. It's just like well wait a minute, we're adding an app? You're doing maps? You're what? When? Have you guys looked at the calendar?

<group laughter>

Kocienda: But yeah, it's just like oh yeah, we're just going to get it done. Okay.

Weber: But why not add it after launch? Because it was such a cool feature for the launch?

Williamson: Exactly. It was such a cool feature for the launch. It could have been a fourth thing.

Weber: Yeah, moving maps are a big deal.

Williamson: If we had had it when the thing was announced it could have been a fourth thing.

Hsu: This is something that I actually forgot to ask earlier but you were the manager of this team. Right? Was this your first manager position?

Williamson: Yes. Well, no, that's not true.

Hsu: Oh, you had started a company.

Williamson: I had started a company and so I had a company with 35 engineers reporting to me. And then I was CTO or CTO in training at Resonate. So, but then when I first came back to Apple I was an individual contributor.

Hsu: But was it because of that previous experience that you were tapped to be the manager of that group?

Williamson: I kind of fell into it. It's like okay, I need to get some more people on this. I pulled people in. And then…

Hsu: And you were just sort of the leader so then you became.

Williamson: Yeah. And then things just kept adding, you know. Yeah. It's strange, though, I never really-- there wasn't really an idea of career development and you're going to move into a management track now. And it was just like, what do we need to do to get this done? Okay. We need more people. Let's go and get some more people. Okay. What do these people need to do? They need to do this. So actually it wasn't until pretty late that things were formalized in terms of the management structure. I think there were a lot of fungible components as far as people. So the division between Nitin and I eventually-- there was still even fluidity among people-- like Scott Hertz worked on keyboards a lot initially.

Kocienda: Sure.

Williamson: But he was really mostly responsible for…

Kocienda: Springboard.

Williamson: …Springboard. Yeah. Springboard is the Finder app on the iPhone.

Kocienda: Home screen button.

Williamson: Home screen.

Hsu: Talk a little bit more about that process of finding people, recruiting people, building out the team.

Williamson: So we were very lucky. Steve gave us a mandate to, you know, whatever resource you need across the company, go get the people but get the best people. And we were lucky enough to find some really talented folks within Apple. I don't think we hired anybody from outside of the company for quite some time. And we had folks like Wayne Westerman who is just mind-blowingly brilliant in terms of math. And even though we didn't even up using much of his code in the keyboards, he was an advisor in that process.

Kocienda: Yeah, and of course, we used basically all of his code under a certain level to do all of the touch processing.

Williamson: Yeah. So he came from FingerWorks. And then kind of the people that I look for in terms of coding are the people that kind of are driven to code. There's-- I don't want to get in any kind of trouble with HR but I think some people have a gene for coding. And other people don't. So it's not necessarily something you can learn. You either have it or you don't. And I could be wrong but that's just what I found that there's kind of an OCD aspect to really good developers. They want to code. And they can sit down in front of a keyboard and eight hours later oh my gosh I've been coding for eight hours. So I think we were a whole team of people like that. We all had a little OCD.

Kocienda: A lot.

<group laughter>

Williamson: But that's what-- it's kind of necessary to have that when you're dealing with very complex systems. You need to-- we talked about context shifting or context loading to understand the system and be able to manipulate it. You need that kind of mentality and not everybody has it. So I looked for people like that at Apple. And we got some really, really great developers.

Hsu: What was the camaraderie like within the team? I mean you said there wasn't really much of a division between your group and Nitin's group. You were sort of one big kind of…

Williamson: So, you know, when you talk about camaraderie you might think about we all go out and play basketball together or go and have a beer together. It wasn't like that because we were working so intensely. So the camaraderie was really about connection around the work. We all understood the code base. We had arguments about "do it this way or do it that way." But the camaraderie was about a shared intellectual endeavor. And it's very different what I imagine-- I'm not a sports guy but very different than camaraderie around a football team. It's really the shared mental space.

Kocienda: The shared mental space. I think there's also this aspect to it of-- that you get together and you have a discussion that-- it's like we're here and we need to be here. It's like okay I'm going to go away and I'm going to do this part and you're going to go away and you do that part. It's like well, while

I'm doing my part over here I'm not going to be looking at you doing your part. It's just that when I'm done you better be there because that's what we agreed upon. And we both need-- I need this part that you just agreed to go do. And it's like when that happens, when it works whether that's two hours or two days or two weeks to get that level and you then sync up with that other person and that other person has delivered, it's like you build this trust, this sense that yeah we are all in this together. It's like yeah, you're not my best friend. It's just like look when I want love I go home and pet my cat. I have dinner with my wife. This is not friends. It's not fun. It's about getting a job done, and getting the work done.

Williamson: You know, and I think kind of the way that trust is built, this might be an over simplification but there is at Apple, unlike other places that I've worked, a huge emphasis on API. API is kind of the contract you have with your coworkers. You adhere to this API. You design an API. And that API has to mesh with whatever you're working on. I mean I don't know if they do this anymore but we used to have an API Review Board. And any API that was going to become public or might become public had to be scrutinized by a team of very senior engineers. And if the API wasn't good enough they had to go back and be fixed. But because there was so much attention on getting those APIs right and consistent with the rest of the system that, you know, you really could build these massive complex systems and have it all work well together.

Kocienda: I mean it's just another kind of trust. Right? So you just have the trust in the software. It's like I call this API it's going to do what it says it's going to do, what it seems like the contract is. It's actually going to fulfill that contract. And so yeah, you just-- it's like one of those ways that you wind up building momentum and building speed. You're not wasting time waiting for somebody to-- it's like hey look I'm done. It's like what, you've not even started yet? Gosh. It's like what can I even do until we get back up here because we both need to then go to the next level together. That back in this time that we're talking about, the Safari WebKit time, these teams were very, very small, highly, highly skilled, dedicated people. It just all happened. It just all came together. Good things happened because of this trust and because of these systems and this focus on getting the work done right.

Hsu: Ken, you mentioned that you collaborated with Bas and the design team a lot on the keyboard and other things. Can you talk about that process, like the back and forth? And how much input did you actually have on the design?

Kocienda: On like specifically about the keyboard?

Hsu: Yeah but other things as well.

Kocienda: Just generally working with-- so the Human Interface team at Apple, we called it HI, Human Interface. So the HI team was the folks who were software designers, graphic designers. They were not necessarily technical. They were-- in some ways they were advocates for people because some people who were really, really good at code don't really get or understand people. Again, this-- if you have a brain that really can mesh well with computers it may not necessarily mesh so well with people. I put

myself as guilty as charged in that camp certainly. And so it's this effort of trying to develop a process where you've got engineering and look and feel going on, working at the same time. It's not this thing-- I don't think you can successfully develop a product by engineering it and then slapping on look and feel later as so many high tech companies, consumer electronic companies out there in the world try to do. And so it's this process [of] working with these designers very, very early on. And so where I worked with Bas on the keyboard for the original iPhone, we would get together and talk. He would be coming at it from the standpoint of well, this thing needs to look good. What might the colors be? What might the fonts be? What are the size of the fonts? If we're going to have a popup show up under your finger, how big should that be? Should there be a little animation? Or should it just pop and then go away in one frame? Should there be a little bit of a f-- it popup and fade away? Or should it just popup and pop-down? If we're going to have just QWERTY letters on the first keyboard layout, where do all of the other letters go? What does that look like? And so we would have these conversations. And, again, he would be looking at it more from this look and feel and designer aspect. And I would be looking at it more from the engineer and coding aspect. But the key is, is that we didn't-- I didn't come up with the engineering from here and he didn't come up with the look and feel from here. We absolutely overlapped because for someone like Bas he could absolutely get in and he had his own little demo environment. He used Macromedia but then later Adobe Director to make demos. So he would make little demos and write little bits of code to communicate his designs as well. And so it was just this, again, I really boil it down. It's this process of trying to integrate the engineering and the look and feel so that there's as much of an overlap as early on in the process as possible. So that those two strains are running through the product right from the beginning and then eventually added to the hands of people who are going to be using them.

Williamson: And I think Ken's description of his experience with Bas and the keyboard is true across the whole product.

Kocienda: Yeah.

Williamson: You know, we were lucky to work with not just brilliant engineers, but brilliant HI designers. And it worked best when there was that overlap.

Kocienda: Yeah. Later, slightly later on in the process the lawyers would come because Apple patented all of these things. And so one of the things that the lawyers always said was well when we put the list of inventors on the patent we need to know what you did. Because if you didn't do something that-- actually through the patent editing process they may go and say well, we're not going to actually put that claim on the patent. So they needed to know which inventors were on which claims because, again, the editing process just might mean that then an inventor's name might come off the patent because they might file or do a separate filing. Or decide that that's not patentable or for whatever reason. So the inventors, they needed to know who the inventors were. And when it came to the keyboard, like look and feel for the keyboard, Bas and I were looking at each other and it's like, "Ken did you think of that?" It's like, "Bas, I thought that was you" because we didn't know because it was just this process of being so deeply in the work and standing in front of a white board or looking at a display or trying something out that you just

don't remember who was actually the first one to have the idea because somebody throws something out that's sort of it. And then the next person finishes the idea, finishes the sentence. And so absolutely that's what it was like working with Bas. I absolutely loved working with Bas whenever I got the opportunity to do it.

Hsu: So then the thing that you mentioned the letters popping up, you don't remember which one of you actually came up with that?

Kocienda: No. I mean I can't.

Hsu: It was one of you, both of you?

Kocienda: I came up with *an* idea for it in some form. And then Bas supplemented it and made it better. And then I implemented it. And then he tried it. And he said, "How about you make it work a little bit like that." And then I'd go away and I'd try the font size he suggested but I thought maybe a point or two might be bigger. And that's how you just wind up with this…

Weber: Iterative.

Williamson: Exactly. Iterative development is something that we did a lot on the iPhone team. So it's not like there's a product specification. You enable the specification and you're done. That's not how we developed the iPhone. We had a rolling prototype. And it was constant iteration on design tweaks and coding tweaks and performance tweaks.

Kocienda: Everything. Everything was always open for debate. And whether it was an engineer going from one office to another. I mean if you think about it this way it's like "I need to use the Contact." We try to live on this one. I need to use the Contact app. And so I might got to the Contact app engineer and say, "Well, maybe if you change it around it would work a little bit better." But then the Contacts engineer needed my keyboard and it's just like "hey, this keyboard, how about I have this little change, it would actually make it easier to fill out the form." That's one of the ways we came up with saying that you can get an all caps keyboard to fill in the state abbreviation. So these ideas, who came up with that idea? I don't know. It was the whole team and the whole process came up with the ideas. Everyone's collective investment into the whole effort, that's what came up with the idea. I guess the lawyers need a name for the patents but it really is the whole group, the whole outlook, the whole approach.

Hsu: Was it similar working across with the hardware group, with Fadel's organization as well? I mean was it that same level of collaboration? Or was there barriers?

<section off the record>

Williamson: So further aspects of the hardware is like, what can we do in silicon to make the software go faster? Is there anything we can do, anything from video decoding, to compression to what can we put in silicon because it's going to be faster than doing it in software? And although huge debates about the amount of RAM. You know, I was a huge advocate for more RAM and the hardware guys went no. And we're talking about pennies. But they insisted on living with the RAM that we had. Or Jetsam, a whole other discussion about Jetsam.

Hsu: How did you come up with that?

Williamson: So well it's pretty clear that we had to do something because the frameworks on OS X are written with the assumption that you have unlimited memory because you have virtual memory. So if you run out of physical memory you swap. And so almost no software is written with malloc, assuming...

Kocienda: Malloc can't fail.

Williamson: Yeah, malloc can't fail. So Malloc is the memory allocation routine that's kind of the lowest level memory allocation function on OS X. And nobody checks whether or not they get a pointer back, whether they got a return value. So we weren't going to rewrite a lot of the software that we imported from OS X onto the iPhone. So it's pretty clear early on we had to do something. And so, the solution was, use a gun and shoot a process in the head if it's misbehaved. So the idea was, in the kernel we were monitoring memory allocations. And if memory levels were getting low or there was an application that was abusing memory we'd just kill it and give precedence to the foremost application. So when you dismiss an application on the phone it doesn't necessary exit. It's given a bunch of opportunities to do things before it has to exit. And if it doesn't exit then we kill it with Jetsam. So this was actually, in the very early days, something we had to get right to take software frameworks that were designed for a desktop operating system and run it on a system that didn't have paging. Didn't have virtual memory. So Jetsam was the solution. So we met with the kernel guys and pushed this idea forward. And they were like, you know, you're crazy. This is ridiculous. What's going to happen? How are things going to get cleaned up? But eventually they came around. And we implemented it. So it's remarkably seamless, how memory management works on the iPhone. You know, you really don't notice when applications exit and start. So yeah.

Kocienda: It's one of those things that worked out so well. I think it was eventually like Matt Watson [ph?] was the engineer…

Williamson: Yeah, so in terms of implementing it, it was the kernel team. And I didn't implement it. But I actually had the original idea.

Kocienda: Right. Right. But eventually it was that somebody who was going to come along and say, yeah I can make this thing work, who was sitting in the right place in _____.

Williamson: And also Eric what's his name? Eric-- I forget his last name. But Eric and Matt both did an incredible job working on it because it got far more refined over time. And it's a real operating system behind the iPhone and multiple processes. And one of the things that I was very harsh about is limiting the amount of processes that run. And everybody wants to spawn off a process to do something. You know, you look at a modern Mac, just you boot the darn thing and it's like 20 or 30 processes running. And it's okay because you've got virtual memory. But doing that on an iPhone, disastrous. So every time somebody wanted to introduce a new process they had to justify why they needed that new process. Because even if it just takes 2 or 300K it all adds up. And on a modern operating system you don't really worry about that, but on the phone we had to. So it's kind of like API review committee but for process review committee. And now it's really different. There's a lot more memory. But in those days we needed every K. Sorry, I kind of went on a tangent there.

Hsu: No, no, no, that was good because I was actually planning to ask about that.

Kocienda: It's really good. I think Jetsam is one of the incredible unsung heroes of the whole system. This system would not have worked otherwise. You would start an app and you would do something, it would crash. That's what would have happened because that's what would happen if you call malloc and you don't get a pointer back and then you try to de-reference that pointer, it's going to go boom. The operating system will shut you down immediately. And so to always make sure that there was this buffer and that Jetsam was working silently in the background. It's just this huge success story, that the whole product would not have worked without it.

Hsu: And Ken you mentioned you had something that you did want to say on the record.

Kocienda: Oh. I had a fine experience working with one of the iPod engineers. There's, you know, the-- of course, when we were working on the iPhone we didn't call it the iPhone because we didn't know that's what it was going to be called. I actually found out in the hall when Steve actually said iPhone. I don't know if you knew ahead of time. Anyway, I had no idea up until the morning that Steve announced the product. So we called it Purple or P2. So our phone, the original iPhone, the code name was P2. So there was a P1. It never shipped. But the idea was to make a phone that was based on an iPod. And it was going to be with some click-wheel like keyboard. And so I worked with one of the iPod engineers to make sure that whatever code that I was writing for the keyboard, the software and text autocorrection system would work on P1.

Hsu: So you were actually-- both teams, even though they were competing were-- you were writing code that could have been shared with them?

Kocienda: Yeah. Yeah. So I worked for several weeks and that's actually what made me first decide to starting writing some of the keyboard code in C++ so it could be shared with the P1 team.

Williamson: Really? I didn't know that.

Kocienda: Yeah. That was really it. That turned out to have a huge, huge benefit because of the features. Some of the features of C++ just made it possible to implement the keyboard.

Williamson: I mean, I didn't know that P1 was still live at that point.

Kocienda: Yeah, it was. I worked with what was it Stephen Wang [ph?] for several weeks. But yeah, then going to C++ was just really made the autocorrection software possible. So it's like you have this-- P1 never shipped. We cut off that project a couple of weeks-- you know, several weeks I would say into it. But then it had this long term positive impact on the code that could never have been anticipated. So It's just like one of those things that happened and had a pleasant side effect.

Hsu: And, again, the timeframe for that was?

Kocienda: Gosh, well, so I won the great privilege of making my hair go gray at the same time and doing the keyboard in October of 2005. So this is November, December 2005ish.

Hsu: And by your memory, how long was the P1 project…

Kocienda: Yeah, that I was working on [it] like I said a few weeks. Not more. It certainly wasn't as long as three months. Maybe as long as two but I doubt it.

END OF THE INTERVIEW

