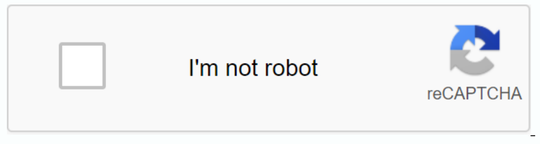

Continue

Coding decoding questions pdf

From 2005 to 2014 (or 483 weeks to be precise), London-based consultancy Berg stood quietly at the forefront of interactive product design. Encouraging the studio to search for opportunities on networks and physical things positions them about a decade ahead of their time, so it's no surprise that tech brands like Google and Twitter have hired Berg to help them invent an upcoming gift -- in which tablets are ubichi, AIs are as smart as a puppy, appliances connect to the cloud, and texting is a new interface. Matt Webb was Berg's CEO, and now splits his time between design consulting, hardware startup advisory and software. No wonder he should know how to code. Well, sort of. If anything, Webb is a living example of how the question on everyone's --- Should designers code? — has no black and white answers. The code has good and evil powers inherent in it, and it is these different possibilities that we all need to develop in our native language of understanding, says Webb. Sounds right. But how? I recently interviewed him on this topic. Here are his thoughts (condensed and edited for clarity): Why he started the course about two days in college, I had a spiritual experience with transisters. And suddenly things were different. I studied physics, and in the second year we studied electronic orbiters in semiconductors. Once the job we had to do was to do something that works like a transistor: like a channel lock or gate with lots of electrons on one side and not so much on the other, and when you poke it on one side, all the electrons rushed through. We made this model and found out it matched all our calculations, then we put it to the side and they gave us a real transistor---three black plastic piece with three legs and we tested that too. Then we got a handful and had to make ourselves a logical gate and then we threw it away and got a 6502 processor, which is an incredibly simple processor. And I vividly remember the moment we connected our 6502 to a small keyboard and a small display, and typing 2+3 and pressing a button to get it to perform. And in my head, just seeing every register of buttons and every transistor gate open and close gave me an incredible sense of dizziness from what was happening all the way to the electrons themselves. And I suddenly felt like I understood it, even though I really didn't. But at that point I was like: I don't know what I want to do next, but it has something to do with computers. Programming is not something transcendental: Mount Everest is still just a very large hill. I'm very bad at coding. I can barely build anything more than about 20 lines of code. But this introduction gave me the view that [programming] is not something transcendental: Mt. Everest is still just a really big hill. Speaking of transisters explains nothing about the code culture and the kind of thinking you need in order to do so david hears you from this world, which in the beginning is completely beyond your conceptual understanding. When treating the code like a physical material, I barely know anything about wine, but remember walking from the moment the variety seemed potentially infinite to Good, there are two colors of wine, and they taste different. Once you have this tiny handle on it, you can recognize when something doesn't taste quite right and you can do something with it. You can start to see opportunities. You start to see how to snag things together. Here's an example. I had a computer in my college room and I realized I could have a web page on my computer and from that web page I could run something on my computer that would work in my room. So I made a small web form that, when you type into it, would use a speech synthesizer to talk about it. A good friend of mine was in Australia at the time, so if he wanted to talk via email but I wasn't at my computer, he could go to that web page from an internet cafe, type something into it that would speak out loud and grab my attention from somewhere else in my apartment. It was about 10 lines of code. It's just figuring things out - take any next step that makes sense at the time. Coming up with the next step that makes sense is sometimes difficult, but the way to do it is to just spend time with it. So it feels a bit like modeling. You know, as they say, that the statue is in the rock? All you do is since the part, which is not a statue, one piece at a time. You have to be pretty down with the idea that it would be useless and embarrassing to begin with. As he learned: it's pointless to do the beats of careful studySome people need to understand what's going on on at a deep level, but learning new things for me about just going through the motions. The only language I can bother is to do anything in Python. I remember learning: I got a book about an idiomatic python, and I didn't focus very much. I just copied it. I literally cited in everything that was idiomatic. It's like learning how to drive. You're sitting in that chair with an instructor next to you and you're just doing what you're being told at the moment -- despite driving in completely the wrong direction, being dangerous in traffic, and everything else. Then 10 hours of this passes and you are a competent driver. And 20 hours pass, and you're good enough. And the feeling that you haven't really learned anything, but clearly you've learned the loads. It just bypassed your conscious mind. It's important to know what automation does for people – to know what a job is below the API, or maybe. It's the same as knowing how security agencies monitor email and use meta-data. But does everyone literally need to know how to code to do this? Perhaps not. It may just be that they will understand it culturally because someone close to them knows how to do it. Not everyone should do the same, but you need to people of people so that these things may be absorbed. On code as controlE a quote that resonates with me from [Ted Nelson's 1974 book] Computer Lib, which was similar to a Bloomberg article [What is code?], but written on a typo and self-published. He says: Whatever it is in the real world, for a computer program, it's just another device. This statement says everything we need to understand about how Uber treats the driver: the driver is just another device. It's dehumanizing. People need to understand the tools of production, so that they begin to understand their manipulative forces. But Ted Nelson also called the coin hypertext and came up with the idea that you could use a computer for creativity. So the key here is for people to come to know these manufacturing tools so that they start to understand their manipulative powers. If you understand the manipulation inherent in the medium of reality TV, watching it almost becomes a game when you're trying to explore how edits try to manipulate you. The software and code are also as follows. On what code I can't think there are a few important reasons why people shouldn't learn to code. One is that it can encourage you to take it to break it, an algorithmic approach to absolutely everything in life. The code is such a powerful approach, I think you can easily sort of steamroll over other things. Remember those kids at school who were really good at a debate lesson? When they start treating every conversation or interaction that way, you wanted to remind them that not everything is discussed. Sometimes you just banter in the park. So the coding position is not always the most appropriate way to engage, but also what does it teach? People who grew up with different video games have learned that there is always a solution to this one puzzle on the screen on the platform they are on. It can be incredibly difficult, but that everything is interpretive in its approach. Maybe it's not great because you don't know what code doesn't. It doesn't talk about feelings, and it doesn't say anything holistically. It doesn't talk about survivability, and connection, and things that are true without being reliably true. I'm not going to say that people who code always have this approach because they absolutely don't. But as a descriptive mechanism for our society, I think there are things he can leave. The fact that your mileage can varyIt is safe that I came out of 10 years of managing a design company, and it was all about embracing the intuitive. And while it's that the code was a big component, it was just one medium. In Berg, this was woven into a much richer tapestry that you never get, just using one mechanism. However, we hit places that were better because we used code as one of these mechanisms. It just has to be a thread, not a kind of dominant thing. Advertising – Continue reading below advertising – Continue reading below - Continue reading below