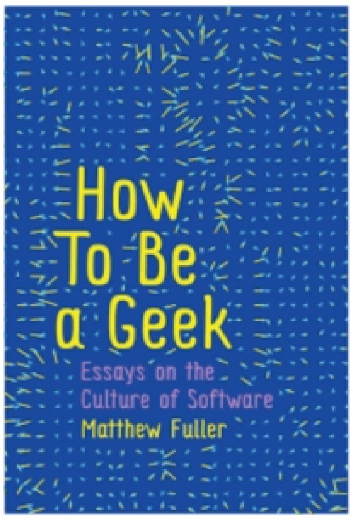
Book Review: How to be a Geek: Essays on the Culture of Software by Matthew Fuller

In **How to be a Geek: Essays on the Culture of Software***,* **Matthew Fuller** *explores the bits and bytes that have reshaped our world through a collection of essays that examines the figure of the geek and software cultures. While the lack of cohesive thread and use of terminology means this collection is best suited to scholars already familiar with the field rather than newcomers, the book contains some useful and astute insights, writes* **Wendy Liu***.*

***How to be a Geek: Essays on the Culture of Software*. Matthew Fuller. Polity Press. 2017.**

Find this book:



Software is eating the world more than ever before, and we're all facing the consequences. Airbnb is raising house prices in cities around the world; Uber and Deliveroo are preying on a pool of un- and under-employed workers to create precarious labour with few protections; Facebook and Google have leveraged widespread adoption of their free products to become unavoidable gateways to digital advertising and, as a result, indispensable cogs in various global value chains. The world is now teeming with multinational corporations whose billion-dollar valuations stem primarily from their intangible assets, most notably software, and they're moving fast and breaking things without any regard for the social consequences.

This is the scene onto which Matthew Fuller's *How To Be a Geek: Essays on the Culture of Software* has emerged. Software is becoming increasingly crucial to the functioning of our world and yet is still frequently thought of as inaccessible to outsiders—instead, it is considered the domain of the 'geek'. Published last April by Polity Press, *How To Be a Geek* aims to demystify software: to get behind the screen and explore the interstices between the bits and bytes that have reshaped our world. Fuller himself is Professor of Cultural Studies at Goldsmiths University, and has published several other books on the subject of software studies, including *Software Studies: A Lexicon* (MIT Press, 2008) and *Behind the Blip: Essays on the Culture of Software* (Autonomedia, 2003).

How To Be a Geek is Fuller's latest offering on the topic, and it consists of eleven academic essays that were authored or co-authored by Fuller and originally presented or published between 2011 and 2016. It's certainly a welcome contribution to the topic, but its presentation leaves much to be desired: it's an uneven and ill-defined collection, combining a mishmash of different approaches, topics and writing styles that never quite cohere into a logical whole. Contrary to what its title may imply, this is not a book about 'how to be a geek'. Those who are looking for a first-person account on the cultural aspects of software production should look elsewhere—Paul Graham's *Hackers & Painters* or Ellen Ullman's *Close to the Machine* come to mind—as Fuller's approach is more about investigating the figure of the 'geek' through the lens of cultural studies, treating it as an intriguing specimen for academic analysis.

That's not to say that the book is wholly unsympathetic; it's just that its primary tone is quite detached, almost clinical at times. Fuller's definition of geek is to be 'over-enthused, over-informed, over-excited, over-detailed'; someone who expresses an 'extraneous kind of desire' when faced with 'an interesting problem' (2), which Fuller identifies as the key persona behind modern software creation: 'geeks created the internet and fight over its meaning' (2). This is not a book for software developers looking for a critical perspective on software culture; *How To Be a Geek* is *about* the geek, not *for* the geek, and its goal is to shed some light on software culture for an academic audience, ideally one with some knowledge of social theorists like Michel Foucault and Gilles Deleuze.

Image Credit: (Deedee86 CCO)

The book is divided into four thematic clusters: 'Histories'; 'Entities'; 'Aesthetics'; and 'Powers'. 'Histories' offers theoretical deep dives into two different paradigms in computing history. 'The Obscure Objects of Object Orientation' looks at the history of object-oriented programming, while 'Abstract Urbanism' investigates the use of agent-based modelling in the field of urban planning from the 1960s onward. Although there are some interesting insights in both, neither is likely to be especially captivating for the general reader. Both are quite long and dry, and the combination of arcane details of developments in computing history and ethical considerations drawing from social theory can make for a context switch that is at times dizzying.

'Entities', which contains three essays broadly touching on contemporary software culture, makes for smoother reading. 'Software Studies Methods' provides a short overview of software studies and how it arose from what Fuller calls the 'elective solipsism of disciplines' (56)—the tendency of, say, social theorists to refer to computational concepts like algorithms without recognising their cross-disciplinary nature. 'Big Diff' is an in-depth study of GitHub, a 'large-scale distributed software repository', exploring its influence on the culture of software production through its key role in software workflows in free and open source software communities. Finally, 'The Author Field' is a microscopic genealogy of the particular piece of metadata within software known as the 'author', considering the philosophical implications of such a field in the context of software version control as well as its applications for forensics.

The 'Aesthetics' section, containing five essays, is a little harder to pin down, but it's unlikely to be of interest to anyone who isn't already well-versed in computational aesthetics, which Fuller broadly defines as 'involving both perception and composition' (8). 'Feral Computing' is one of the stronger pieces in this section, and it offers an intriguing cybernetics-inspired perspective on software creations that go 'wild', escaping the intentions of their creator. Even stronger is 'Just Fun Enough To Go Completely Mad About', which begins in a conversational register with Fuller ruminating on his own experiences with computer gaming. The rest of the piece describes various unexpected phenomena that have emerged from gaming culture, including Let's Play, 'speedrunning' and Twitch Plays Pokemon, in which the playfulness aspect breaks out of the controlled confines of the game itself and projects outward like some sort of ludic aura.

The final section, ‘Transparency’, contains only one essay, ‘Black Sites and Transparency Layers’. (Puzzlingly, the introduction mentions the existence of an essay called ‘Algorithmic Tumult and the Brilliance of Chelsea Manning’, which is described as an attempt to ‘maps the politico-computational condition of the present in relation to the themes of posthumanism’, but which does not appear in the table of contents nor anywhere else in the book.) In ‘Black Sites’, Fuller traces the origins of the term ‘black box’, referring to a system that can be viewed solely in terms of its inputs and outputs as its internal workings are unknown. Following Bruno Latour, Fuller explains that successful technologies often undergo a process of ‘blackboxing’, whereby they become more and more opaque over time, and proposes a cybernetic interpretation in which the black box and the person working with it merge. There are some thoughtful insights here, primarily economic: Fuller makes a compelling link between the idea of a black box and neoliberalism, whereby the market is perceived as the ultimate black box, thus justifying the neoliberal drive for increasing marketisation. In a similar vein, platform companies like Uber and Airbnb are presented as market-based examples of using interfaces to obscure relations that are put in place.

Overall, *How To Be a Geek* is a useful addition to the field of software studies, geared primarily toward academics who are already enmeshed in the field. There are some astute insights to be found as the book investigates software through the lens of cultural studies, though it will require some work on the part of the reader, as the prose blends terminology from both disciplines in a way that can be hard to follow. Its primary drawback is its light editorial touch, resulting in an admixture of essays that do not have a cohesive thread tying them all together. Those who are new to the topic are therefore recommended to start with Fuller’s 2008 compilation *Software Studies: A Lexicon*, which consists of shorter and more focused essays that serve as a better introduction to the field.

Wendy Liu is a software developer with a BSc in Mathematics and Computer Science from McGill University. She is currently a student in the Inequalities and Social Science MSc program at LSE as well as an economics editor for *New Socialist*.