All Technology Sucks

A list of rants about technology most
people don’t care, or even know, about

Kai Lyons

ISBN: 9798427957267

PREFACE

General Topics:

- Operating systems
  - Linux
  - Windows
  - MacOS
  - Android & ChromeOS
  - Google Fuchsia
- Programming Languages
  - Python
  - JavaScript
- Blockchain
  - Cryptocurrency & Non-Fungible Tokens
  - Web 3.0
- Artificial Intelligence
  - It doesn’t really exist
  - Self-driving cars
  - Self-learning algorithms

Table of Contents

PREFACE.....4
Chapter 1 – Operating Systems.....8
Chapter 1.1: Linux distributions suck.....10
Chapter 1.2: Linux is getting too big for its boots. 14
Chapter 1.3: Linux desktop environments.....18
Chapter 1.4: Speaking of Windows.....26

Chapter 1.5: Microsoft and Anti-trust.....28
Chapter 1.6: Windows 11.....32
Chapter 1.7: MacOS – If only….....40
Chapter 1.8: Android's Not Linux.....43
Chapter 1.9: ChromeOS.....45
Chapter 1.10: Google Fuchsia – Death to Linux?
.....51
Chapter 1.11: F is for Fuchsia.....53
Chapter 2 – Programming Languages.....57
Chapter 2.1: Introduction to Python.....59
Chapter 2.2: Python is for Prototyping.....61
Chapter 2.3: Introduction to JavaScript.....62
Chapter 2.4: JavaScript is not good.....64
Chapter 3 – Blockchain.....68
Chapter 3.1: Cryptocurrency.....70
Chapter 3.2: NFTs Suck.....72
Chapter 3.3: Web 3.0.....74
Chapter 4 – Artificial Intelligence.....78
Chapter 4.1: Artificial, but not intelligence.....79

OPERATING SYSTEMS

Chapter 1 – Operating Systems

In this section of the book, the topic of operating systems and how they are not – to put it lightly – great. An operating system is a piece of technology that allows the user to interface with application. Operating systems come with another smaller bit of software called the kernel which is the piece of software that the operating system talks to in order to get resources from the hardware. That is the TL;DR version of the kernel, at least.

Other terminology to note of.

- GUI: Graphical user interface, essentially when an application is a GUI it has a graphical user experience – non-text based, padded and colored with icons – with the purpose it is aiming to build upon.
- Desktop environment: A part of the operating system that allows for users to have a

graphical user experience of some form, though not necessarily a GUI. Uses a window manager, and then adds other software to make the window manager a little more feature rich.

- Window manager: Standalone applications that a desktop environment expands upon when making itself. Manages opening and closing applications, moving them on a screen, and generally managing all the low-level stuff. Some people actually run window managers over full-fledged desktop environments.

Operating systems are a difficult thing to get right, which is why there is so much for people to rant about. With the few operating systems that really exist in the public psyche:

- Linux: A kernel that has multiple developers working on multiple operating systems called

"distributions" of Linux. Linux itself made by a community, a non-profit (The Linux Foundation), and Linus Torvalds.

- Windows: The largest operating system on the market – excluding Android as these rants are desktop-oriented. Made by Microsoft.
- MacOS: A generic catch-all term referring to any graphical desktop operating system made by Apple Inc.

Chapter 1.1: Linux distributions suck

Linux is not one, but thousands of operating systems. It's so massive, new users look at the "distributions" – operating systems that use the Linux kernel – in terms of trees and branches. The largest tree is the Debian tree, Debian being the second oldest active Linux distribution, only beaten by Slackware. The Debian tree, while having an

above-average number of branches, is only so large because of one of the other trees, the Ubuntu tree. If you heard of a Linux distribution like ZorinOS, Pop!_OS, KDE Neon, Linux Mint, elementaryOS, or about a billion others, those are built on Ubuntu – or a derivative of Ubuntu. The problem with the Ubuntu fork plague is that anyone can do it. It's too easy to fork Ubuntu, which is why around a quarter to half of all Linux distributions are based on Ubuntu.

Don't believe it is easy? Here is a genuine tutorial on how to make your own Linux distribution based on Ubuntu, using tools that already do most of the work for you.

1. Download https://github.com/UbuntuBudgie/iso-builder, we will be using this to build a fork of Ubuntu.
2. In the folder iso-builder/etc/, open terraform.conf

3. Find the NAME option (should be on line 21) and change it to the name you want in quotes.
4. Find iso-builder/etc/config/package-lists.calamares
5. Modify it with the packages you want or made
6. in iso-builder/etc/config/archives add your repo URL to the .list file, and then add your own key
7. Run in a Docker container

While sure, okay, it’s a little but more complicated than this, it does work. These are the seven basic steps to forking Ubuntu with one of the easiest tools designed to fork Ubuntu. When it is this easy to fork a Linux distribution, no wonder people would want to take advantage of these tools and make their own Linux distribution. It’s almost a right of passage to make a Linux distribution, but

there are other ways to do it, especially the most fun one.

Doing a Linux From Scratch install is easy enough, you take the book, follow the instructions, then eventually bam you have a bare-bones operating system running on Linux. Now, a Linux From Scratch system is easy to make, but difficult to maintain because you don't have daddy Ubuntu – or any other upstream – to maintain packages for you. You have to install everything from source, but that's not necessarily a bad thing. Once you're done, run a few magical commands then you have a downloadable file. This is a book of rants, not a tutorial on how to make a Linux distribution. You can figure out what to do by yourself if you really want to make something.

Anyways, my point is that it is way too easy for another piece of garbage distribution to appear and continue to clog up resources and make it difficult to support other smaller distributions which

may actually have a solid goal in mind. It’s already bad enough supporting a Linux distribution with zero cash flow, zero thanks, and zero help, but to also have to compete for resources with a distribution that doesn’t care about improving Linux but is designed to make a million dollars or some other worthless goal, without putting in real effort.

Chapter 1.2: Linux is getting too big for its boots.

This is not talking about user base, no. Linux, on the desktop is actually very small – and continuously failing to sustain long-term permanent growth. No, what is meant when it is said that Linux is getting too big for its boots is that Linux has a size problem. Bryan Lunduke – super fantastic content creator, knows corporate Linux like the back of his hand, and generally smart and experienced person – in his talk *Linux Sucks 2021*,

he mentions how Linux is gaining two million ish lines of code per year.

Let’s put this into perspective. Jurassic Park, the original theme park from the movie series *Jurassic Park* ran on a total of two million lines of code. That was considered a large amount, enough to hold dinosaurs. As it stands, Linux has roughly twenty-seven million eight hundred thousand lines of code as of 2020 estimates. Hold on to your butts, because that is nearly fourteen Jurassic Park’s worth of code, and about one new Jurassic park is being added per year. That’s a lot of code.

Okay, what’s the big deal? So there is a lot of code, it isn’t like the more code there is the harder it is to maintain. Except, that’s exactly how this system works. It’s difficult to manage more and more code. Let’s say theoretically, someone wanted to print the Linux source code and read it like a book. They read at a pace of 10,150 lines per

day. It would take seven years to get through the code at that rate. Seven years. That's just to read it, not to mention looking into it and fixing it where needed. There are times in every program where something needs to be fixed, especially something as critical to the world as the Linux kernel. With how many lines of code there are in the Linux kernel, it's no wonder why new vulnerabilities are being discovered more and more. Linux has a problem with its size, and soon its going to be even more difficult to notice these issues. Linux needs thousands of eyes on it at all times just to survive,

Okay, so maintenance is an issue when Linux is at this size, but it's not like it's fixable. Oh but it is fixable. This Linux size issue is relatively easy to solve. The Linux Foundation needs to build a dedicated "code efficiency" team to work on the Linux kernel's many issues regarding its size. Doing so will help eliminate major bugs and vulnerabilities hiding in plain sight, and will also

allow the kernel to be more efficient which is always a good thing in servers. While doing this will be more complex than just “take big code make less code”, but that is the general idea that will take the project and make it far more sustainable long-term.

Chapter 1.3: Linux desktop environments

With desktop environments, they are a chaotic mess of opinions and are generally a useful tool, but with confusing problems. For all intents and purposes, we will ignore the endless number of standalone window managers, and focus on the main few desktop environments.

- GNOME: Currently on version 4X, and first released in March 1999.
- KDE Plasma: Built by the KDE team, with its predecessor K Desktop Environment 1 being released July 1998 with KDE Plasma 5 being in development since 2014.
- MATE: Being in development since 2011, and built as a fork and continuation of GNOME 2.X
- Cinnamon: In development since 2011 by the Linux Mint team, and until the release of the Ubuntu Cinnamon Remix, seldom found on

any other distributions of Linux. Built on GNOME 3.X

- Budgie: Developed by the Solus distribution since 2013.
- Pantheon: Developed by elementaryOS, it is a MacOS-esque design. It is primarily independent, and almost completely hand-made.
- LXQt: In development since 2013.
- LXDE: In development since 2006, the predecessor to LXQt.

This is not a complete list of desktop environment options on Linux, but they are the few that are considered the “major” desktop environments. The reason they are listed out is to show how many there really are. Desktop environments are one of the few ways each Linux distribution are truly different from one another.

Desktop environments are so commonly the true deciding factor between distribution choice, many suggest to tell new users what desktop environments to use, and not which Linux distribution to use. This is not a good thing.

If the deciding factor is a smaller aspect of the entire operating system itself, that means Linux has hit the point where Linux is practically a singular operating system. This means there are fewer points of contention between Linux systems, and fewer innovations are being made as the reason to innovate is nullified by there being no real points of competition.

In most Linux distributions, there are three points of decision one would normally make. Package manager, userland, and desktop environment. As of late, the differences between package managers is minimal. Using APT on Ubuntu is barely any different from DNF on Fedora (for all intents and purposes, smaller differences do

still exist). With the userland, almost all Linux distributions have settled on a GNU + Systemd userland. There are very few systems that disagree with this, and they are by no means considered “major” competition between bigger players like Ubuntu, Fedora, Debian, or even comparatively to Slackware.

So with desktop environments, are they the last frontier for innovation within Linux? Yes, but also possibly no. New desktop environments seldom appear. One of the newest efforts to make a new desktop environment is from the System76 company, who are building their own COSMIC desktop environment.

While many smaller Linux distributions and operating systems are creating custom desktop environments, it isn’t common-place for these systems to be using something more custom, when developers have often done most of the hard work

already, some distribution developers may just be asking “why reinvent the wheel?”

It’s not reinvention, but improving upon commonly placed concepts. Especially right now with desktop environments, there needs to be innovation to compete in areas like speed, functionality, and form. It’s never fun to say, but most desktops fail to support all three.

- GNOME has an interesting form, but hurts itself in speed and functionality coming after GNOME 4X especially.
- KDE Plasma: Has fantastic function, and is really fast, and it has decent form. It is a prime example of what desktop environments should be doing but aren’t.
- MATE: Has speed for ages and has some of the best functionality, but by default its form is a little off.

- Cinnamon: Being built as a fork to GNOME 3.X, it does somewhat suffer in the speed aspect in tiny amounts, but functionally it’s fantastic. It’s form is beautiful, though highly generic as a Windows-esque desktop environment.
- Budgie: Fantastic with form, and its functionality is good, but it isn’t the fastest desktop on the market.
- Pantheon: Like KDE, it’s just check, check, check. It could theoretically do a tiny bit better with its form, but in all honesty it’s a tiny nitpick.
- LXQt/LXDE: Since both have similar developers and development spheres, both check the speed box. Its form could do with some work, and it needs some better functionality as many things are still unstable – especially in LXDE. Both being Windows-esque hurts its form points.

As presented, each of the main desktop environments do well still, but all of them can do better. One thing most users will see is either GNOME, a MacOS-esque, or a Windows-esque environment. This is no accident. Most desktop environment developers are looking to make their desktop very user-friendly out of the box. This is more harmful than good, because these systems often don’t look unique, thus defeating any purpose they have as many users will choose based on which looks the closest to Windows 7/10-esque.

The problem with Windows-esque desktop environments especially is that they are targeting a market that will likely never arrive: Windows newbies who barely understand how to use Windows, live alone Linux or one of the BSD branded systems.

Desktop environments need to take the time to create a design that works, that breaks away from a Windows-esque environment for a more

custom, better suited for users in general. If that means some design aspects should be Windows-esque, so be it. Just don’t be Windows-esque to mimic Windows.

Chapter 1.4: Speaking of Windows

Windows is a genuinely garbage operating system merely kept alive by its application ecosystem, and its multi-billion technology corporation backing it. Microsoft can keep Windows on top practically forever. As will be elaborated on further in the Windows 11 chapter, as well as the anti-trust chapter, Microsoft is well known for its ability to keep projects on top with nothing but marketing and good relations with its partner hardware manufacturers.

Microsoft built the vast majority of its empire off of Windows and MS-DOS. MS-DOS has a dirty history. From one of the core reasons for causing Gary Kildall to lose much of his business, to being completely stolen software, and being at the core of several instances where Microsoft broke anti-trust law. Windows has a long, dark history.

MS-DOS was created using the QDOS (Quick and Dirty Operating System) system as a base. QDOS was directly stolen from Digital Research Inc's CP/M operating system. CP/M at the time was proprietary software, but is now available as open-source software. QDOS, later known as 86-DOS, was in development until 1981, just under half a decade before the first release of Windows.

Windows and MacOS both took the idea of a graphical user interface from the same corporation, Xerox. The first desktop environments were made at Xerox PARC, where they were experimenting with the new technology. While graphical user interfaces have existed since the 1960's, with video conferencing existing before the ARPANET.

While Xerox was the first – releasing their GUI operating system Xerox Star in 1981, while MacOS would release in 1983, and Windows in 1985. It's now rare to find an operating system for

desktop users that is graphical only, excluding major exceptions that have multi-targeted systems like FreeBSD which is primarily a server operating system that is usable as a desktop operating system, without it being its only purpose.

Chapter 1.5: Microsoft and Anti-trust

While this section isn't directly about Windows, rather a quick history of Microsoft's Internet Explorer browser, however, this is a preface to the Windows 11 section. The history of Microsoft Internet Explorer (or IE) is important to note for the current in-progress history of Windows 11.

The history of Microsoft breaking anti-trust law is long and extensive. Specifically, the late 1990's and early 2000's browser wars that Microsoft would later temporarily win until the

existence of competing browsers Google Chrome and Mozilla Firefox came in the mid-to-late 2000's.

During the browser wars, the biggest browsers were Mosaic and Netscape Navigator. Mosaic being an older browser with minimal support, it would later collapse in market share on it's own with Navigator taking crown. Then came Microsoft. Before the explosion of the internet, Microsoft – at the time lead by Bill Gates – was not keen on the whole idea of the internet. After the explosion, Microsoft realized it was a mistake to ignore this new market.

At the time, Netscape Navigator's kingdom seemed impenetrable from all angles. Especially for new commercial browsers, as even an undercut in price would not necessarily take Netscape Navigator down. But Microsoft, at the time having the largest operating system platform – even at the time – had a scheme to destroy Navigator.

Microsoft’s secret weapon was to pre-include Microsoft Internet Explorer on Windows for free. This would be an immediate death-blow to Netscape’s business. The problem being that the browser being free (a new concept at the time), Netscape Navigator had zero chance of actually competing unless they did the same. But the convenience at the time to just use Internet Explorer because it was preinstalled onto the worlds most popular operating system on the market.

That wasn’t all. Microsoft Internet Explorer still had to support many new technologies that few to none were currently supporting. Not to mention, Internet Explorer was built off of the Mosaic browser (Another example of Microsoft used an existing technology to get a head start, rather than starting with custom technology). This means Microsoft had a browser, but not necessarily the best for most customers. Netscape Navigator could

have survived if Microsoft decided to play fair from there, but of course they didn’t.

Microsoft had power over the platform Netscape Navigator needed, meaning Internet Explorer would be ready for release before Netscape Navigator could. With Windows 95/98, Microsoft did not allow other developers to develop their software on builds of the operating system before release, giving a head start to Microsoft.

It was this platform control, and head starts that allowed Microsoft to win the browser wars for a short while. Microsoft did have to fight with the Federal Trade Commission over the anti-trust laws and monopolization of the browser market that Microsoft was pulling. Netscape Navigator would not be saved, and Microsoft would face minimal consequences.

To end on a good note, Netscape Navigator did continue in memory through the developments at Mozilla and with the still rather popular Firefox

browser. Not to mention, browser market share is still rather varied, even though Google Chrome has the largest market share.

Chapter 1.6: Windows 11

Windows 11 is everything. A new type of disaster. A return to old habits from Microsoft. Not to mention, breaking their own planned release schedule. Windows 11 is a mess of a release from Microsoft, and it really does have its issues. Windows 11 is even a repeat of Microsoft’s older habits from last chapter, but with a nice new twist. Currently, Microsoft isn’t being punished for what they are doing with this version of the operating system.

Before the repeat of the anti-trust issues of days past, there are the other issues of Windows 11 that are not nearly as extensive to mention as

the new anti-trust issues from the same operating system release.

Windows 11 isn’t generally compatible. While some older hardware can support it, due to the new security requirements it is almost required to use newer hardware. This wouldn’t be a huge issue, if Windows 10 – with better hardware compatibility – would be turned into “Windows” as a rolling release system to work alongside new Windows releases, but it isn’t.

Windows 10 was supposed to be – according to Microsoft – the last major release of Microsoft before turning into a rolling-release system. This was what was expected, this is what was told to the masses back when Windows 10 released in 2015. Why did they change this? Well, for Windows 11.

Windows 11’s biggest flaw is with how it sets its default applications, because that’s where the new anti-trust issues arise. With Windows 11, the

new settings to replace Microsoft's new browser (Edge) is far more painful than ever.

One aspect of Microsoft's dominance that wasn't mentioned in the previous chapter was Microsoft's usage of Internet Explorer for every aspect of the operating system in order to make Internet Explorer be more system integral. The new settings in Windows 11 mimic this exact behavior.

In the settings of Windows 11, Microsoft's Edge browser is default for many file extensions, and actions. This wouldn't be a bad thing, if it wasn't intentionally designed to be as abusive and difficult to replace as possible. For example, Microsoft Edge is separately set as the default for both *.html* and *.htm* files, which while both are practically the same type of file.

This new abuse from Microsoft Edge doesn't just end there. There are some applications for Edge that cannot be changed by default, such as the default browser – plus Bing search engine – for

search within the application menu’s search functionality.

Microsoft’s forced venture into browser support is to collect as much data as possible with data collection systems within the Edge browser, and not to mention the default search engine in Edge is Microsoft’s own Bing browser. While this issue exists in Windows 10, it’s to a far lesser extent. Microsoft’s blatant abuse is harmful towards new users and competing browsers, which is the definition of an anti-trust issue. What other applications are set to Edge as the default? Well, it’s not as easy as saying applications, because it’s specific file extensions.

Everything from HTML, HTM, SVG, PNG, JPEG, JPG, PDF, SHTML, and a million others set to a Microsoft Edge default. Microsoft is blatantly forcing new users who don’t want to spend the time changing each and every individual setting to Microsoft Edge, or forcing other browser makers

like Mozilla and Google to make their own settings changers within the installation of their own browsers.

Microsoft is out of their mind if they think people are going to let them get away with this. While these massive issues are nowhere close to make Windows users do a mass migration to an operating system like Linux, or even MacOS for many users.

If users become complacent with this blatant abuse of power from Microsoft, users might as well lose the right to have selection of a browser on Windows in the first place.

Complacency is the biggest concern with this whole situation. Something needs to happen, but it’s concerning when Microsoft holds a practical duopoly with Apple in the desktop operating system space because of Microsoft’s stronghold with application support.

Sadly, until Microsoft Windows executable files run naively – not through a compatibility layer as many applications can now – on an operating system like Linux, there will be no mass exodus from Microsoft to a more open operating system like Linux.

This genuinely sucks. While Linux has learning curves, all operating systems have them. Linux is generally well positioned to take over Microsoft's monopolization of the desktop operating system market, only if users can get the applications they need. Linux is in a catch twenty-two against Microsoft where it needs users for application support to come – as developers don't want to support a platform they will not see a solid user base. However, without application support, Linux is less likely to get new users.

Microsoft does support applications on Linux, including VSCode – a popular code editor – and

Microsoft Edge, but that doesn't necessarily mean new users will be switching to Linux any time soon.

Contrary to popular belief, web-applications will not – at least in the near future – replace desktop applications. If everything a user needs is web-based, then is more than likely to run on a browser on Linux, making Linux a viable option for a new user. However, for applications like Microsoft Office, Adobe Creative Suite, and the vast majority of games simply do not run (naively) on Linux. This gives Microsoft a strong position, with its biggest competition coming from Apple's MacOS which supports many of the same applications.

This makes Microsoft's continued monopolization of its platform even more egregious, as it already has full control over so many markets. Fun fact, this book was written with LibreOffice Writer, on Ubuntu Cinnamon Remix. From an outside perspective, Microsoft is getting far too abusive.

It’s not unreasonable to ask Microsoft to not be nearly as abusive as it is being. Microsoft could open-source its entire operating system, and while yes they will be forced to compete – especially with Linux taking every aspect it can into its own sphere – they would still be one of the top dog operating systems, alongside MacOS.

Microsoft’s abuse is for the sake of data and money, not even expansion. This data is important to Microsoft because it allows Microsoft to sell advertising on their Bing search engine, as well as on Windows itself. This doesn’t excuse their actions what-so-ever.

How can this issue be fixed? Boycotting Windows 11 to the best of one’s ability. Windows 10 is still supported until 2025, and could theoretically be extended by a couple more years by user security support, anti-viruses, and more. If users can still stick to Windows 7, which lost extended support in 2020, then users can stick to

Windows 10 to make a point out of it against Microsoft's abuses.

Chapter 1.7: MacOS – If only…

MacOS is a nearly perfect operating system. It is largely based off of various UNIX-like operating systems and a lot of FreeBSD source code – FreeBSD being my usual daily driver. Not only that, it's a solid operating system, that looks nice, is functionally great, supports most applications, is secure as all heck, but… is proprietary, and not available on hardware not made by Apple.

The biggest problem with MacOS is that it is a closed-door ecosystem, that only officially allows Apple's hardware to be used. It's a problem, because Apple can then charge practically whatever they want for the computer itself. Apple is notorious for its general pricing abuses, with what is called the "Apple Tax", the price of a "free"

operating system included with “premium” hardware. This hurts.

While yes, users can build a “hackintosh”, it’s often unclean and has compatibility issues with many applications. While Apple doesn’t crack down on hackintosh projects – with a couple asterisks – it is still technically piracy and violation of the terms of service of the MacOS terms of service.

The asterisks are the times companies tried selling hackintosh computers, which Apple took personally and so it shut those companies down with various lawsuits.

While hackintosh is an option for those willing to take a risk, it’s one most people shouldn’t take. Not to mention, it’s a difficult process that leaves you wanting more. Users would be better of shelling out thousands for an officially supported computer, or to not use MacOS at all.

In a perfect future, this wouldn’t be the case at all. MacOS-supported hardware would be

affordable and could be more than what Apple itself has made, and custom hardware configurations could officially exist.

While MacOS is a near-perfect operating system, the hardware it comes with being Apple made, and only ever Apple made, really hurts. Some users may want to use Apple's software, without buying a computer that may be as expensive as a car can often be, if not more.

When computers are worth more than cars, based on a premium "company tax" really hurts the consumer markets. When it comes to Apple, it still hurts, when there is no way to use valuable software on hardware anyone can buy and set up.

Apple should sell a standalone version of their operating system for users who want to use MacOS without spending thousands of dollars for a new piece of hardware. Cars often take loans to buy, and Apple expects you to just have the money, and it's not okay.

Chapter 1.8: Android’s Not Linux.

Android is a mobile operating system from Google that runs on mobile devices. It is the technical winner for the most used operating system, however, it does kind of cheat when it holds a practical monopoly (80% market share) of a market that is more global than desktop operating systems due to the better affordability and need for phones over most desktops and laptops.

As the chapter title suggests, Android is not a traditional Linux distribution – though being based on the Linux kernel. The big issue is that Android should not be looked at as a solution for Linux’s woes and problems.

This is more of a Linux rant, but Linux needs to stop using Android as a crutch and selling point for desktop operating systems built on Linux.

Linux’s claims to the Android market are about as strong as Google’s claims over Microsoft Edge’s market. While Edge does use Google Chromium – the open-source base of the Chrome browser – as Android uses Linux, it’s not a strong enough claim to really make Android a “Linux system”.

This even applies to ChromeOS, which is Google’s desktop operating system built off of the Gentoo desktop operating system – which is a Linux distribution. However, with how Chromium is used and marketed, it’s still rather difficult to consider the two systems in the same operating system family.

Chapter 1.9: ChromeOS

ChromeOS is a disaster of an operating system, yet the biggest innovation of our time. How it became the third largest operating system is both expected and shocking. Expected as in Google was expected to succeed in entering an open-for-the-taking market, shocking in how large that market was comparatively so large.

ChromeOS is a comparatively bare-bones operating system when it comes to application support – only recently supporting Android's application suite, and even more recently supporting Debian Linux packages to a minor extent. However, ChromeOS's bread and butter is web applications.

With ChromeOS's focus on web applications like Google Docs, Sheets, Slides, Drive, etc, it makes it a very strange and confusing operating system in concept. While the design and intent of the operating system feels borderline insane –

Web applications being a relatively new idea in the scope that they exist now. Google Docs only releasing in 2006, Google released ChromeOS back in 2011, which is only a five year difference. Ten years later to the release of this book, web applications have slowly taken place as anything stable and commonplace.

It would have been surprising to even see people with a Chromebook – a device that runs ChromeOS – in 2016, yet six years later ChromeOS has surpassed MacOS in market share. ChromeOS itself is a weird oxymoron of sorts where it is so good yet so bad.

What makes it good? The fact that a Chromebook from 2013 can still run almost as well as it did on day one. Chromebooks and ChromeOS is very versatile, as is the Gentoo base it uses.

ChromeOS also has everything most home-users need in a web browser, which many don’t need anything else to deal with anymore. If

someone had a grandparent that needed Facebook, a word processor, and video conferencing tool to see the family, ChromeOS has all of that in a bundle that have computers selling for less than three hundred dollars in some cases.

The reliability of ChromeOS makes it so the system can be used as long as the Chromebook has electricity, which is a good thing. While some newer applications may be too big or sluggish for older Chromebooks, much of the functionality still exists and still runs really well.

ChromeOS isn’t just reliable, it is also secure. While a web-based operating system seems super easy to target, with ChromeOS being near-immutable and most of the time applications cannot mess with the system software itself, it’s fascinating to see how secure of an operating system it can be, at least compared to Windows.

This isn’t to sell users on ChromeOS, heck I still prefer FreeBSD and Linux systems, but it is not

fair to just discount what the ChromeOS team at Google has done to make an operating system so good, though with many flaws.

ChromeOS's biggest flaw is that it is not perfectly these things. It's not perfectly secure, it's not perfectly reliable, it's not perfectly application rich. However, it often claims to be these things. Chromebooks have had scam advertising to pretend their devices can do things it can't.

Chromebook isn't a replacement to a computer, it is a sidekick to a normal desktop computer. ChromeOS isn't even too special, projects like Ubuntu Web Remix are aiming to try and at least shake up the current status quo of the Chromebook "web-top" (laptop but for the web only) monopoly.

While Google's monopoly in this "web-top" sphere was gained not by brute force – unlike Windows – but rather through an opening in a new market that spawned on accident.

ChromeOS is essentially now Android built on top of the Gentoo operating system, making it less special as every new release comes out. It’s still special, just not as special as it once may have been considered. The Android x86 project has been around since 2009, two years before the ChromeOS system released, and years before ChromeOS really started focusing on Android applications in 2016.

While ChromeOS is becoming a more and more generic operating system, especially with containerized applications for Linux software becoming more and more of a focus, it hurts.

Not to mention Chromebooks becoming more and more expensive with no real reason to be so expensive. While yes, there are still Chromebooks that sell for less than five hundred dollars, there are other Chromebooks that don’t have a low price tag. Some Chromebooks go for as high as a few thousand dollars.

However, many Chromebooks still sell for even sub-two hundred. Most still sell for a lower price than most Microsoft Windows laptops. One of the cheapest laptops – Pine64's Pinebook (discontinued) – was roughly $100, and Raspberry Pi's 400 series goes for $100 per product (without a screen) so Chromebooks still sell for some of the lowest possible prices for a laptop, which they provide valuable services, while still being nonsensical when it comes to its purpose (especially currently).

Chapter 1.10: Google Fuchsia – Death to Linux?

Google Fuchsia, no matter how much Linux users don't want to admit it, is an eventual replacement to the Linux kernel for Google's operating systems options. "*But Google said…*" Whatever Google has said about Fuchsia not being a replacement to/for Android and ChromeOS is wrong. There is no reason for Google to be experimenting with a new operating system without intending to replace their current offering with something better.

Google has good reason to replace the Linux kernel – coming from a Linux user standpoint – and not just Linux's kernel having problems. Google's current offerings are Linux-based, meaning they have to follow the same developments and rules as Linux.

Going back to an example from the Linux desktops section, GNOME. Many desktop operating systems built on Linux use the GNOME desktop environment, but that also means they have to follow GNOME's design guidelines for newer versions, which isn't always appreciated. Developers have to either find a new solution that better follows their guideline, or build their own desktop environment that follows their own design guideline.

Essentially, this is what Google Fuchsia is: A replacement not because Linux has a problem, but rather Linux has its own vision that Google no longer accepts or agrees with in some capacity.

It's also important to mention that operating systems like dahliaOS even are experimenting with the Google Fuchsia operating system as a base operating system. Google's Fuchsia is a replacement to Linux, like it or not. Google Fuchsia

will eventually be the replacement kernel for Android and ChromeOS.

Google Fuchsia is a lot like the NT kernel for Windows. Back in the early 1990s, Microsoft was looking to replace their MS-DOS based operating system with something better suited for a modern age of Microsoft. In 1995, Windows 95 was released. The Google Fuchsia project is Google's NT.

Chapter 1.11: F is for Fuchsia

There is a reason that Google Fuchsia is referenced to as Google Fuchsia and not simply Fuchsia. The operating system itself is still incomplete, and even with projects like dahliaOS making it more complete, it's still not ready to be the replacement everyone expects it to be.

Google Fuchsia will also not destroy Linux as many fear. While some operating systems not made by Google will use it for their base, it's going

to be far and few between. Google Fuchsia is by Google, for Google. Not really the Linux killer many claim it will be.

While Google Fuchsia will replace Linux within the Google-sphere of products, Linux and BSD-based projects will continue to be Linux and BSD-based projects. Linux will still be the main kernel every developer in need of a custom operating system will go to in order to make their operating system, except for the exceptions with the BSD operating systems being used as a base for operating systems development.

It will be interesting to see this new operating system family spawn, and where it will go in the future. But for now, Google Fuchsia is a blink in Google’s eyes of what the future will be, not what the future is.

PROGRAMMING LANGUAGES

Chapter 2 – Programming Languages

Programming languages are a hot topic for many. A programming language is a syntax and application to compile/interpret the instructions to do a specific purpose. There are hundreds of programming languages, maybe even thousands. Many programming languages fit as prototyping languages, and not always a production-ready programming language. Other programming languages are just terrible with application and package management.

Some terms to note:

- Compiler: Turns source code into machine language to be executable.
- Interpreter: Runs through the source code line by line executing as it goes. Usually

slower than a compiled language, but are often far more multi-platform friendly.

- Just In Time compiler (JIT): A type of compiler often used by an interpreter to make its execution a little faster.
- Indent-based: Uses indentation to define code blocks. Example in the introductory Python chapter.
- Bracket-based: Uses curly brackets, parenthesis, or regular bracket to define a code block. Example in the introductory JavaScript chapter.

Chapter 2.1: Introduction to Python

In order to rant about the problems with Python, it needs to be introduced and described. Python is an indentation-based, object-oriented, functional, duck programming language. A duck language essentially means “if it walks like a duck, talks like a duck, looks like a duck, it is probably a duck”, with it’s class structure. An example from Wikipedia on duck typing is provided in the following code excerpt.

```
class Duck:
    def swim(self):
        print("Duck swimming")

    def fly(self):
        print("Duck flying")

class Whale:
    def swim(self):
        print("Whale swimming")

for animal in [Duck(), Whale()]:
    animal.swim()
    animal.fly()
```

The output given should give an error for the Whale class which does not have a function for flying, as whales cannot fly. However, both whales and ducks can swim. A whale is not a duck because it can not fly.

With this example of Python’s syntax, developers often also use indentation to define a code block. This is not a Python tutorial, so there won’t be too much detail on how this works, but that is how code blocks are defined.

Chapter 2.2: Python is for Prototyping

Python is often used for everything. From web development, to “artificial intelligence” – which does not currently actually exist, but that’s a rant for later.

With Python being a very simplistic, english-like language, it’s perfect for prototyping, but not really as much as an in-production language.

Python is the worlds most common language, but it does come with some caveats that most developers tend to ignore. Python is an interpreted language, meaning it can be run anywhere, and using a JIT on many platforms also helps speed the language up by quite a bit, but it can still be much to inefficient as a binary executable.

An interpreted language has to be distributed via the entire source code, or a minified version of it if possible. This means every indentation, whitespace, etc adds to the total size of the project.

This isn’t great, and should be avoided if at all possible. While there are methods of compiling Python into binaries, that ultimately removes the purpose of Python’s ability to run on all platforms as its binaries would be restricted to the ones it is compiled for.

Chapter 2.3: Introduction to JavaScript

JavaScript is a web-based scripting language that through mad science alone became a general programming language that is used similarly as much as Python. JavaScript first appeared in 1995, coming from the Netscape company for their web browser.

JavaScript uses the ECMAScript standards, as to be standardized on all platforms. This standardization is good, however, JavaScript is not. JavaScript is a bracket based language, meaning indentation means very little, if anything at all.

An example piece of JavaScript code courtesy of Wikipedia.

```
// Arrow functions let us omit the `function` keyword.
// Here `long_example` points to an anonymous function value.
const long_example = (input1, input2) => {
   console.log("Hello, World!");
   const output = input1 + input2;

   return output;
};

// If there are no braces, the arrow function simply returns the
expression
// So here it's (input1 + input2)
const short_example = (input1, input2) => input1 + input2;

long_example(2, 3); // Prints "Hello, World!" and returns 5
short_example(2, 5);  // Returns 7

// If an arrow function only has one parameter, the parentheses
can be removed.
const no_parentheses = input => input + 2;

no_parentheses(3); // Returns 5
```

In general, JavaScript is a very generic looking language with many issues. JavaScript is also mostly an interpreted language.

Chapter 2.4: JavaScript is not good

JavaScript as a programming language is simply a chaotic mess of different competing technologies, and incoherence between platforms. Even with the ECMAScript standard, JavaScript can still be disorganized and disconnected between platforms like NodeJS and general browser JavaScript.

The problem with desktop JavaScript solutions like Deno and NodeJS is that they are fragmenting the language into many different parts. With JavaScript being a scripting language for web-based Java, a technology that barely continues to exist, JavaScript's continued popularity of a language is astounding to say the least.

JavaScript is not only a mess in configuration, but also a mess with it being an interpreted language built for front-end web design being used for non-web technologies.

Using JavaScript for things like desktop applications – using ElectronJS for example – is like using hand-sanitizer as hand-soap. Sure it's possible, but as a concept it doesn't make sense.

JavaScript is designed to be used in the front-end of JavaScript applications. Superset languages like TypeScript do a better job at being well designed as a desktop-application programming language solution, as that is the core purpose of those languages.

It's not the end of the world if JavaScript continues to be as popular as it is, but it is still nonsense as JavaScript was never designed to be used for the things it is being used for, and it often shows.

The Blockchain

Chapter 3 – Blockchain

There is no technology form so new yet so strange as the blockchain. From digital currency you can pay your taxes with, to digital images claimed like they are physical paintings. The blockchain is a fascinatingly weird new technology.

The blockchain is essentially a decentralized and distributed database of information that is primarily designed for trade. Like a public ledger of things people own through the blockchain.

As a technology, the blockchain is a strange and interesting new set of technologies. Created in 2008 to serve bitcoin ledgers and public transactions, the blockchain slowly but surely became one of the largest modern technologies today.

As a concept, nothing is inherently wrong with the blockchain. It is only when one reads into

the subsections of the new technology where flaws form.

The technology was designed primarily for Bitcoin, but it is now used for all sorts of cryptocurrencies and the newly popular non-fungible tokens (or NFTs) and the new concept Web 3.0.

This book will not dive deep into the concepts of the blockchain, more so just a general overview to keep things simple as things can get really complicated and confusing quite quickly.

Chapter 3.1: Cryptocurrency

Cryptocurrency like Bitcoin, Litecoin, Dogecoin, Shiba Inu Coin, and what feels like a billion other cryptocurrencies in existence is what the blockchain is designed to support.

The problem with cryptocurrency is that it's like stocks in both concept and variability. In the past, one Bitcoin would have gone for as little as one cent. Currently, one Bitcoin would be a transaction worth tens of thousands.

No one expected cryptocurrencies to succeed as well as they did, but is it fair to call cryptocurrencies a currency?

As mentioned, cryptocurrencies are more like a form of stock rather than an actual currency. Even popular stock trading platforms have been supporting cryptocurrency as something you can buy, just like a stock.

While there is nothing inherently wrong with this, some government agencies have been looking at cryptocurrencies all the wrong way. The state of Colorado became the first ever state to accept cryptocurrency as an acceptable form of tax payment.

While nothing is necessarily wrong with trading cryptocurrencies in order to pay taxes, it is still an odd idea given cryptocurrency's stock-like nature. It's really no different if you pay 0.02 Etherium in taxes as your 2 shares of Amazon as some strange form of tax payment.

While nothing is wrong with it, it doesn't mean it's not a bad idea. Paying taxes in cryptocurrencies is just too risky, given that 25 Bitcoin just a few years ago would have been the award money for 8th placing participants in competitions – as an example.

Chapter 3.2: NFTs Suck.

The concept of an NFT does not work with the blockchain. Non-Fungible Tokens, or NFTs for short, are the idea of buying and selling images as a form of cryptocurrency. There are only so many images, so there are only so many NFTs per new NFT.

NFTs commonly break copyrights to build a more popular image, and then sell them for absurd prices on the blockchain.

Fun fact, the blockchain actually doesn't actually support images – or even videos in the cases of some NFTs. So while someone could buy the Charlie Bit My Finger video, ownership would not be proven via the NFT but rather a separate hash to identify as the text based equivalent of a certificate.

While copying an NFT is technical copyright infringement, one can still hold copy to the image without said certificate.

The purpose of this is essentially like a new form of art collection, with people spending hundreds, even thousands of dollars for a picture of some animal or figure with a specific design in order to trade it later for more money, or just to keep it for gits and shiggles.

NFTs are genuinely terrible as a technology. They are copy-pasted images with extra features like hats, facial expressions, copyright infringements, and all that for a certificate of ownership worth about as much as printing the words “I own this thing” with an identification number for said thing printed below it.

Chapter 3.3: Web 3.0

Web 3.0 is the so-called future of the internet, but in reality is simply a scam designed to screw over old people into thinking Bitcoin and Etherium are the most important parts of our future. The definition of a scam, according to Merriam-Webster is "a fraudulent or deceptive act or operation", with fraudulent meaning, according to Merriam-Webster means "characterized by, based on, or done by fraud : deceitful". Essentially, the goal is to deceive people into an idea. When Googling "Web 3.0", just to find resources to better define Web 3.0, the first result essentially told me "Web 3.0 is a myth", which is not a good sign.

Genuinely, Web 3.0 is a failure of an idea, as the blockchain was never designed to handle such large technologies like media and web content.

The problem claiming decentralization is that it's not really decentralized. Sure it will run off the blockchain, but that's not really helpful of a point

when the blockchain is heading towards centralization itself. Companies like Google, Meta, Microsoft, etc will be able to control and regulate - or push for a regulative body as the government itself is looking at a blockchain future – especially with states like Colorado accepting cryptocurrencies for tax payments, meaning it is going to be regulated.

Every major new technology will be regulated by one body or another. The FCC regulates television, radio, and internet services. The FTC regulates how companies can act. The Linux Foundation is a non-governmental entity that regulates Linux. It’s all to common to find regulation, that later creates centralization.

Artificial Intelligence

Chapter 4 – Artificial Intelligence

Artificial intelligence is the idea that computers can think on their own. This section will mostly claim the idea of artificial intelligence as a farce and explain why self-learning algorithms do not truly constitute intelligence, and how it is practically impossible for a binary computer like all computers are to be as intelligent as a human.

Computers run via strict instruction. Developers cannot be general with a computer, as a computer needs specific instruction, and in reality cannot even be randomized. Randomization in computers is often a set of complicated equations based on environmental factors to generate what seems random.

Computers only do what they are told, and the idea of them being fully intelligent is just false.

Chapter 4.1: Artificial, but not intelligence

As stated in the general chapter introduction, artificial intelligence does not exist.

The definition of intelligence from Merriam-Webster that will be used as the basis of this argument is “the ability to apply knowledge to manipulate one's environment or to think abstractly as measured by objective criteria”, as other definitions do not necessarily fit the definition this book is trying to set, or is way to general as with one of the definitions being “the ability to perform computer functions” meaning “printf(1+1)” is technically intelligence, which is not what most people mean when it comes to the idea of artificial intelligence.

The difficulty of artificial intelligence is the definition of intelligence itself. Intelligence is a very abstract concept, but it is one that is important to define as intelligence is the basis of the idea of AI.

To expand on the Merriam-Webster definition, for an artificial intelligence to be artificial intelligence it needs to independently apply knowledge to concepts.

An example of this would be to take an image processor, and it has to separate blue whales from great white sharks into different folders. The idea is that the artificial intelligence would continuously test based on examples given to it via human interaction. This itself doesn’t inherently break the idea of artificial intelligence, but it does make it struggle as a computer can only understand so much of the differences between a great white, and a blue whale.

How this works is complicated, but in general, whether an image is of a great white shark or blue whale is based on thousands of images for each, and the computer may still get it wrong.

The problem is, computers aren’t independent when it comes to the testing. Each image is predefined and new images may bring confusion based on subtle changes.

These subtle changes bring confusion to a computer, and to truly determine intelligence in a computer, a computer needs to be more than binary. Binary is far too limited for computers not the size of actual city blocks be anywhere close to “intelligent” in the same way we consider humans and animals intelligent in their own rights.

Chapter 4.2: Don’t let cars drive cars

If AI is a set of specific instructions given to a computer via inexact means, what does it mean when the same methods of teaching an algorithm the difference between a great white shark and blue whale. Disaster.

Unlike humans, computers are faster at making the operation, but cannot actually independently determine each and every scenario humanly possible. While a bug hitting the windshield may not confuse a human, it can confuse a self-driving car which may see it as some kind of obstacle.

Self-driving cars are just a bad idea, as the computers need to be large and complicated, which leaves room for bugs and issues. It comes back to the same binary issue, computers run on a binary system when brains do not. A computer essentially is less complicated than a human brain, making it inferior.