Freedom Respecting Technology Definition
bringing about the Next Generation of Open Source, Free Software, Open Knowledge, Open Culture, and Technological Freedom

(current) version 2.5.0 | 2024-03-25 | check here for newer versions

note: This is (generated via ebook-convert from) a single handwritten HTML file without any other embedded resources and can thus fully/cleanly be saved as a self contained unit for offline reading. It's rather annoying that some mobile browsers generally don't have web page saving functionality but this is a defect with them not WHATWG standard HTML as an open file format. (Laptop browsers aren't much better because if a page references say images it generally gets saved as a "blah.html" file and a "blah_files" folder instead of both being wrapped by a folder that can then later be moved/copied as one unit and still benefit from it's subcomponents being easily accessed as desired.) Copy and share as widely as possible but don't make modifications without consulting the authors first.

author information

initiator and primary author:
Georgiy Treyvus, makesourcenotcode the_swirly_thing gmail le_pointer com

reviewers, coauthors, supporters, and other helpers:
{insert (nick)names, pseudonyms, and/or email addresses as desired by contributors here}

contents

intended audience and compact summary

This document is for those that care about the sharing of knowledge in an open accessible manner. Not just theoretically, but practically.

Truly open knowledge and true technological freedom fundamentally require trivial ease in fully and cleanly copying allegedly open digital works in forms useful for offline study.

For example, in the case of software, the overly narrow focus on easy access to the main program sources isn't enough. Trivial access to offline documentation, for any official documentation that may exist, is critical.

Needing a constant network connection to properly study something claiming to be open isn't freedom. Needing the site hosting an allegedly open work to always be up isn't freedom.

We think system wide in terms of the full Open Knowledge Set associated with any given technology. We strongly believe withholding any parts of the OKS from easy offline study is fundamentally no better than withholding any part of the main source material.

It's time for Open Source and Open Knowledge to truly be for everyone, not just well off people with reliable Internet connections.

motivations

Technology in various forms is influencing if not outright controlling ever more aspects of our lives with no signs of slowing down. Both on an individual and collective/collaborative/community level we must understand and control the technology we use or it will control us. The latter may happen by itself in the event of strong artificial intelligence. More likely it may happen by the hands of the few developers who truly understand it and who have a depressingly high probability of being employed/bribed/coerced by unethical corporations and/or regimes. However the latter scenario comes about it's absolutely unacceptable.

Thankfully we are not alone in our sentiments here. Long before most of this document's authors were born many people started work in this direction. An example is Richard Stallman who formalized these sentiments as well as certain good cultural tendencies he'd seen with regards to the sharing of knowledge which were slowly disappearing into what became the GNU Project and the Free Software movement more generally. This also inspired other movements of free (as in freedom) culture/technology including but not limited to Open Source and Creative Commons. All in all these movements have done a tremendous amount of good and are huge steps in the right direction. Ultimately it is on the shoulders of these giants upon which we stand as we write this document in an attempt to progress yet further.

Herein we attempt to fully articulate and address longstanding and growing frustrations we've been feeling for years if not decades. These originate mainly with regards to Open Source Software, to a lesser extent Free Software, and extend well beyond just software into technology more generally.

While many FOSS(Free and Open Source Software) projects are often made in good faith and we often can't express in words how deeply we appreciate some of them the fact is that the vast majority of (sometimes very promising) FOSS projects fail badly to properly deliver on certain critical aspects of openness, accessibility, and technological freedom. In spite of being fully compliant with things like the GPL(GNU General Public License), FSD(Free Software Definition), and/or OSD(Open Source Definition) thus being theoretically free many FOSS projects fail in making technological freedom practical. Unfortunately there are a lot of mostly artificial barriers that (usually unintentionally) stand in the way of many interested people (including very skilled competent ones) exercising the crucial freedoms of using, studying, understanding, modifying, copying, (re)distributing, further developing, and contributing back to FOSS projects in a truly free, autonomous, and self sufficient manner.

Thus we seek to develop this FRTD(Freedom Respecting Technology Definition) driven by the principles of radical understandability, radical

discoverability, radical accessibility, radical open knowledge, and radical self sufficiency. (A crucial consequence of that last bit is that it can, should, and hopefully ultimately will also lead to healthier and significantly more resilient individuals and local communities.)

We aim to carefully study, formalize, and explicitly forbid as many of the devastating failure modes we've seen in FOSS projects at both the user and developer level as we can. (We aim to eliminate the mostly artificial boundaries between users and developers.) We aim to carefully study, highlight, formalize, and explicitly require at least the bare minimum set of practices which some FOSS projects have engaged in (even if only accidentally in some cases) that have given end users real practical technological freedom. Though we won't be able to do it perfectly we aim to have something that as much as possible formalizes and captures the very essence of technological freedom. We aim for something which can guide ourselves and other well meaning developers of technologies including but not limited to software, firmware, hardware, protocols, technical specifications, algorithms, scientific research, knowledge, media (informative and artistic), and combinations thereof(which is what most things are) towards building a truly free and healthy world.

While this document is intended primarily to address problems with Free Software and Open Source Software a lot of the issues discussed here pertain to all software and to technology more generally. Thus we hope this will be read by, of interest to, and acted upon by a wide and diverse audience. We feel that this document can help even developers of closed source software make many aspects of it better as well as provide good food for thought to other related source available movements such as Fair Source. This is because as one often hears freedom is not free whether the cost comes in the form of effort, labor, time, money, or something else. For a technology to support freedom even partially let alone completely it must have certain qualities. Not trying hard to take away people's technological freedoms is not enough. Active steps must be taken to add qualities that create and preserve freedom. Some side effects of technologies having some of these qualities are better user and developer experiences as well as better internal design.

In addition to this definition document, which itself is almost an FRT(Freedom Respecting Technology), we intend to release a few other FRTD compliant technologies to the public both for their personal utility and more crucially for peer review.

These projects are intended to demonstrate and capture the spirit of FRTs in ways that even the best possible version of this document will likely fail to. We want people to feel in a hands on manner the difference between FRTs and typical FOSS. We want people to feel it viscerally. These example projects will almost certainly come in the form of software as things like hardware for example are largely outside of our current areas of expertise. Help realizing FRTs in other mediums is enthusiastically welcomed and would be deeply appreciated. So please get in touch!

We want to test the approachability, discoverability, learnability, accessibility, and understandability of these sample projects on a wide variety of people from both the user and developer perspectives. We want to assure that skilled developers aren't adversely affected by any of the requirements of the FRTD. We want to see if novice developers find the projects approachable for hacking on. We want to see how quickly and easily a random user can cross into developer territory to modify a given FRT to be more in line with their desires. Again being able to do this in theory is not enough. It must be practical.

Finally we hope that this definition and the technologies we release will be the beginning of a movement at least as large and successful as FOSS was. We dream of a world where people can combine FRTs realized in various mediums like hardware, software, etc into full solutions to their technological needs and wants.

main thesis

Failure of FOSS projects to provide the crucial technological freedoms we seek stems from their failure to be reasonably understandable to people (even sometimes ones with significant expertise) in a reasonably self sufficient manner. Understandability and self sufficiency are the two absolutely necessary prerequisites for technological freedom. Hence we

examine what it takes to provide both.

Open Source is an absolutely necessary but often insufficient subset of Open Knowledge which is the real goal and the key to understandability. The crucial but still ultimately narrow focus on source access often distracts us from focusing holistically on maximum possible access to the full Open Knowledge Set associated with a technology which is what truly matters.

What do parts of the OKS(Open Knowledge Set) look like? In the worst case they can be a little abstract and even inaccessible/intangible. For example in the case of software an important such element of the OKS is certain practical educational information exposed by the UI(user interface) in the environments where that software can be run. Other OKS parts are much more straightforward. Another even more important element is both the built and source forms of the Help Information Set a subset of the OKS consisting of things like any officially written documentation, diagrams, instructional/demonstratory videos, and so forth that may exist. Indeed a lot of what this document will focus on is this HIS(Help Information Set), though it's critical always to think system wide in terms of the whole OKS.

Most systems are useless without the HIS component of the OKS from both a user and developer perspective. Depending on how self describing the components of the technology/system in question are the size of the HIS will vary. Sometimes it can reasonably be nonexistent given the other elements of the OKS. However for all but the simplest technologies the HIS will be necessary. Happily many (though not a majority of) FOSS projects do a somewhat decent job of providing the HIS in circumstances where most reasonable people would deem it necessary and/or desirable. The main issue we have is with how the HIS is not accessible in a reasonably self sufficient manner.

Ultimately understandability comes from the full OKS being as ubiquitous and easy to access and cleanly copy (with little to no extra cruft) as humanely possible in both built and source forms. At the very least this must be possible in wholesale fashion and also ideally in more targeted subsets which aids certain kinds of accessibility we'll touch on later.

While we're on the subject of open knowledge it is important to be cognizant that knowledge is not quite the same thing as raw factual information. Knowledge is contextualized information and the FRTD is about Open Knowledge and not Open Facts. This contextualization is best thought of as some meta facts that highlight/enumerate the existence of all the other facts either directly or recursively and help them make sense in some kind of conceptual framework. Often a very small bit of carefully placed context can take a system from absolute zero to literal superhero without this being labor intensive on the part of implementers.

The other key to freedom is technologies enabling self sufficiency. It's crucial to maximize this as much as possible by minimizing the things one is dependent on to study, use, and further develop a technology.

One very important thing this means is the full OKS being cleanly enumerated and easily available for offline use. Not just the main source code and any official built executables as is often the case. Withholding any other parts of the full OKS from easy offline access/study/use is absolutely no different than withholding the main program sources. Whether they are given away for free or sold (for mere reproduction cost or some not extortionate profits) we don't care. The full OKS must be usable in a self sufficient manner in any FRT.

Special emphasis here is on built versions of the HIS. At the very least it must be easily possible to easily enumerate, identify, and conveniently obtain the full HIS. It may also in some cases make sense to make targeted subsets of the HIS easily identifiable and downloadable.

The ability to study, work, and otherwise operate offline as much as possible is hugely important. Unfortunately it's significantly undervalued by a disturbingly large portion of people right up to the point where they find themselves needing it and not having it. FOSS projects requiring users and/or contributors to have Internet access more than is strictly necessary are absolutely unacceptable. It is crucial that FOSS projects both encourage a taste for as well as enable self sufficiency to the greatest degree possible. True technological freedom means among other things not being a dog on the leash of one's Internet provider.

This document's primary author lives in New York City which is a relatively prosperous and technology heavy part of the world. Nonetheless while he usually has access to high speed Internet or at least some kind of Internet the scenarios where he doesn't have it but could use it occur often enough to be noticeable, annoying, and a cause of lost productivity in both research and work.

Furthermore the primary author is both friends with and acquainted with several people who recently/currently did/do not have Internet access at their place of residence. Or at best have spotty access. In many of these cases these people simply can't afford it. In other cases while they can afford it they are still on a sufficiently tight budget that the expense of buying Internet access was not justified as they could access it often enough not to be totally crippled in their life at their local library, in some cases through their local university, in some cases via the extortionate data plans for their mobile phones, and sometimes due to the generosity of neighbors who let them borrow their WiFi. This is true even now in 2023 and was even more true before the coronavirus pandemic(especially in late 2016 when work on this FRTD first started).

Some people the primary author knows even go so far as to deliberately not buy Internet access even though they can easily afford it because they find it to be too much of a distraction in their life and that it gets in the way of both productivity and family time. Yet even they have sometimes been significantly inconvenienced by not being able to download certain documentation that would have been quite useful for informational, educational, productivity, leisure, and/or entertainment purposes. The way FOSS projects inadvertently punish such people for making a rational and healthy lifestyle choice to not always be connected is unacceptable.

Another important point to keep in mind is security. Some friends of the primary author (as well as to a lesser extent the primary author as well at times) have made a point of not being connected to the Internet more than is strictly necessary. It is well known that exposure to a network especially one as big and potentially dangerous as the Internet significantly increases the chances of compromise. Any FOSS project that forces people to be

connected to the Internet for it's study/use more than is necessary encourages them to potentially jeopardize their security and essentially punishes them for trying to stay safe. This is unacceptable.

This wreaks absolute havoc on people trying to get important work done whose threat model warrants using air gapped computers totally disconnected from any network. This has been a real issue for a scientific software consultancy the primary author interviewed with where people working on the more sensitive projects there temporarily had to leave a secure room/facility, go look up some Help Information of FOSS tools they were using, go back to the secure room, and then try to use it from a printout at best or memory at worst. This also is an issue for human rights activists and organizations trying to safely leverage technology to counteract the force/power amplification that technology also gives to oppressors.

Also, forget security for a moment. Normal regular people definitely need offline documentation.

Of course the key thing to keep in mind is that this is the kind of stuff that's happening in one of the more prosperous regions of the world even in the year 2023. In many other places these problems are exacerbated by many orders of magnitude. In many less prosperous places Internet access is notoriously unreliable, sometimes bordering on nonexistent. Also, forget flakiness for a moment. Even now, in the 2020s, literally billions of people alive today have never once been online at all. Ultimately FOSS projects that that force users to constantly need to be online to use or study them inadvertently perpetuate and increase inequality almost as much as some actively malicious predatory monopolistic interests. The rich will be able to study, learn, get better educated, get more skills, build better infrastructure, get better incomes, and get yet more disproportionately rich both with regards to money and overall quality of life. The poor and less fortunate remain largely trapped in their cycle of misery and desperation. They never get a meaningful chance to lift themselves up from that situation and into a better, more prosperous, and happier life. Both advantages and disadvantages of these kinds generally tend to compound exponentially in feedback loops the latter of which we must avoid. Let's not mince words

here. Any FOSS project which does not enable self sufficiency to the largest extent reasonably possible is a big part of the problem.

Instead of forcing people to have to be lucky all the time with regards to a reliable Internet connection any halfway decent FOSS project would make it so that if someone gets lucky just once they can grab a comprehensive HIS along with any other desired OKS parts and then be able to work self sufficiently regardless of future network access. They would also be able to help their friends and people in their community by making copies for them. If they're feeling more entrepreneurial perhaps they can even sell copies. This is something they may also need to do just to break even depending on the exact distribution mode to offset the cost of the distribution media. FOSS projects really could do worse than enabling self sufficiency, more generosity, more trade, and accelerated development for people especially in less prosperous communities. There's a world of difference between people having to get lucky once and having to be lucky all the time.

Furthermore even in the more economically prosperous parts of the world unnecessary dependence on an Internet connection is very unhealthy both for individuals and communities. The Internet for the most part has a rather tree like and hierarchic structure which has several unpleasant ramifications.

Even in the world's less oppressive countries there exist Internet kill switches. These have rarely been used because the regimes of these less oppressive countries often find the Internet to be yet another useful medium via which to propagandize, disinform/misinform, and spy. Another reason is regimes understand that some limited freedoms like a relatively uncontrolled Internet often make for significantly more productivity and commerce among their populations resulting in significantly more taxable revenue. Finally because of it's hierarchic nature most Internet traffic has to pass through certain centralized exchange points where it can potentially be spied on or otherwise manipulated/censored. Yet more devastatingly these centralized points can be shut down by a regime if the cost-benefit dynamic of keeping the Internet working changes. One such scenario might be people are using it to organize en masse in an attempt to address legitimate

grievances that no amount of easy spying enabled by the Internet's hierarchic topological structure can contain/disrupt the movement as would be possible were it still small. Or it could be something much more mundane like misguided attempts by many countries trying to stop cheating on the national exam day.

Perhaps worst of all even if country A's regime is relatively benevolent and doesn't want to shut down the Internet there for malicious reasons the fact is country B's regime does and perhaps one day will. Just about all countries have enemies whether open or secretive. Needless to say the centralized exchange points at or near the root of the tree in any given country make an extremely appealing target for (the intelligence agencies of) enemy countries and can safely be assumed to have been compromised covertly or otherwise by one or more of them. One day some country may very well shut down some other's Internet and with it a significant part of the critical infrastructure the latter depends on resulting in significant devastation. Oh, also, forget well funded intelligence agencies, forget warring nation-states, sometimes this can literally be pulled off by some random individual from the comforts of their home.

Of course we're getting way ahead of ourselves here. A totally random natural disaster can achieve similar effects and a disturbingly large portion of any given country's critical infrastructure isn't even ready to handle that. Also let's not kid ourselves about the climate crisis and the increasing chances, frequencies, and severities of such disasters. The only real question is what century the technologies communities can operate in a self sufficient manner to respond after the crisis will be from.

You dear reader(s) do not want to be relying on the Internet more than is strictly necessary. We believe in a global network transcending all borders that people can use to collaborate, exchange ideas, solve pressing global problems, and organize mutual aid if they wish. The world needs such a network and needs it the millennium before last at the absolute latest. Unfortunately that network is not the Internet in it's present form and likely won't be the Internet in any of its future forms either much as we wish to be proven wrong here. Even in the event a global mesh network of the same scale and span as the Internet were to exist because say Hyperboria, IPFS,

Scuttlebutt, or something like them just randomly caught on and spread like wildfire we still do not want to be dependent on that network more than is strictly necessary. We deeply value cooperation and a healthy interdependence as it makes life easier and raises standards of living. However we also deeply value self sufficiency and independence. In most regards these two possibilities are not mutually exclusive.

Another thing worth noting is that many parts of the world don't even have reliable electricity let alone Internet access. What this implies is that an FRT's HIS should try to account for that. For example in some rare cases where an educational video really is by far obviously the best way for an FRT implementer to get some information across that's fine. But in most cases it really isn't and precludes people from doing things like printing out all or some targeted subset of the HIS at a library or somewhere similar to enable undisrupted study.

Again it's totally fine and even great to choose to rely on each other for various things. The key word here however is choose. What's not so great is to have to rely on each other. Sure there are many times in life doing so is utterly unavoidable. Nonetheless there are just as many times when it absolutely is and forming such unncessary dependences results in fragile unhealthy individuals, communities, and relationships.

We cannot stress enough that understandability via complete Open Knowledge Sets and self sufficiency are key to true technological freedom.

core FRT(Freedom Respecting Technology) requirements atop those defined by FOSS

1. All tangible OKS(Open Knowledge Set) parts must be available for easy self sufficient offline download, access, and use. The full entirety must be obtainable in a very small and tenable number of downloads for any given release of an FRT.

2. Sometimes there are intangible OKS parts like experiential elements with educational information about an FRT's use, operations, and/or workings exposed in some very dynamic user interface of a running

program designed for platform/environment A. Obviously this information can't really be made available to someone with platform B. That's fine so long as tangible OKS material isn't held back.

3. There must be an OKSE(Open Knowledge Set Enumeration) that documents all available tangible elements of the OKS so that people can reason about the parts of it they may or may not already have and the parts of it they may or may not want. Additionally people must be able to reason about exactly what FOSS compatible rights they have. Major intangible elements must be mentioned too if they are the primary mechanisms via which the FRT is intended to be studied.

4. The OKSE shouldn't be fancy, full of ceremony, or otherwise verbose. It just needs to clearly and accurately enumerate all official educational material associated with the project. There may be certain rare cases where it may make sense to have a brief rationale in the OKSE explaining why certain common elements aren't present in the official OKS or certain rare ones are. Ultimately small is beautiful. Nobody likes reading or writing this stuff more than strictly necessary. Not to mention we need to keep bureaucracy to the absolute minimum. The OKSE just must clearly exist, accurately enumerate everything, and be clear that the technology at hand is an FRT. That last bit is so people know immediately they have certain practical freedoms mere FOSS doesn't necessarily give. Also this helps virally spread awareness of FRTs existing as a distinct concept interested people may want to look up. Finally this acts as a signaling mechanism that the FRT isn't just accidentally doing the right thing, but rather very deliberately. We've seen FOSS projects that seemed to be doing the right thing, but then stopped having their documentation be available offline, usually accidentally, sometimes intentionally. In addition to the raw informational aspects, the OKSE is there to bring peace of mind, and make it clear the FRT at hand is not like the FOSS projects just mentioned. Emotional and psychological factors like this absolutely matter.

5. The OKSE must be easily and prominently visible/discoverable at the FRT's Point Of Initial Discovery which will usually be something like

a FOSS project's home page on the web where an interested party lands after first learning of the FRT's existence by some means.

6. The OKSE must detail the FOSS compatible licenses/rights under which all the various OKS components including but not limited to parts of the main program sources and documentation are released to help people quickly and accurately reason about rights and responsibilities they have.

7. For example in the case of software which is the primary author's area of expertise the full OKS includes at least any program sources, the full built existing HIS(Help Information Set), any source materials from which the HIS has been derived, and any officially built executables.

8. Again the built HIS must be easily downloadable for offline use in it's entirety including both Ramp Up and Reference materials for both users and developers. If the HIS can be shown/viewed online without one needing to run a build on the HIS sources it can be just as easily available offline to them too.

9. Sometimes the source material for the HIS of a FOSS project isn't available. But even when it is (which happily is somewhat often) one shouldn't have to deal with the hassle of setting up and running the documentation build/generator tool and it's whole recursive tree of dependencies just to build/get the HIS. It's an important skill which we absolutely encourage everyone to have as we don't want a culture of helpless dependency. However it's not one anyone must need to use when dealing with FRTs.

10. We've seen many cases where we not only have to build the HIS from source material but then the built HIS wound up being a static set of HTML, CSS, and JavaScript files constituting an offline site of sorts. Disturbingly often it then wouldn't work properly or at all unless we ran a web server locally. Avoid this. We've seen plenty cases of a HIS consisting of exactly this composition that don't require running a local web (or any other) server to read them. We could just go and open

index.html in the offline documentation with our browsers and study.

11. If an FRT's offline HIS is an offline website clone with lots of HTML files and some JavaScript the FRT should strongly consider adding in some fully offline search functionality if we're already executing JavaScript anyway. The tooling to do so is out there and it was so beautiful to behold when we've seen it done.

12. There may be some edge cases where the HIS is something truly dynamic because that's by far the best way to communicate certain knowledge and that's fine. Odds are overwhelming though that your FOSS project isn't one of them.

13. Highly dynamic (aspects/subsets of) a HIS like for example Jupyter style notebooks requiring a local web server or ones with lots of videos can sometimes be appropriate as they're the best way to communicate certain knowledge. But often they are not and shouldn't be used lightly. Remember to enable low tech study of as much of the HIS/OKS as possible like for example printing out some documentation at a library. Yes, AI will quite likely change this, but programatically searching through videos is nowhere near as easy as searching through text(or better yet other forms of structured but still human friendly data).

14. Even images while often necessary should not be overly relied on as they're inaccessible to visually impaired people using many kinds of screen reading software. Alt text and/or other such captions should be provided wherever reasonably possible as a mitigation. This also benefits people operating in text only environments. Again, while AI may change this, FRT implementers shouldn't assume or require an overly high tech environment lightly.

15. On the subject of images and videos in offline website clones (or other mediums that can embed/reference them in a self contained fashion) prefer videos to GIFs. This is more the fault of things like browser implementations than GIF itself. But currently the constant default looping/repetition of GIFs can sometimes make it quite hard to parse where the demo starts and stops. Nor can we pause, rewind, forward,

speed up, or speed down selectively. This interferes with proper study.

16. Related to the above mentioned failure mode we've seen situations where there were videos rigged to autoplay, repeat, and had the video controls (play, pause, etc) disabled resulting in an absolutely stellar replication of the above GIF related issues. (In some cases this may have been related to quirks in our browser and not the FOSS project under examination.) Bottom line the videos in an FRT's HIS must not autoplay and must not hide controls.

17. On the matter of internationalization a good faith attempt should be made. We understand that many projects are badly underresourced and can't do a perfect, decent, or even any sort of job here. Perhaps recent advancements in artificial intelligence can help mitigate this. (It would be nice for there to be a universal language in addition to our native ones. And unlike Esperanto actually neutral and actually widely used. If such a language ever truly emerges an FRT's HIS among other things should probably prioritize this before focusing on other languages. Indeed defining the specification for such a language may be a worthwhile future FRT to build in and of itself for linguists with the background, skills, and desire. Seriously. English is utterly terrible. Grammar books are often comically contradictory. For example they can't even agree on something as simple/logical as the Oxford comma for cleanly distinguishing the last two items in a list with 3+ items. This is without even getting into the other uses of commas. Or other aspects of grammar. We have exactly zero optimism this document is grammatically correct for any definition of correctness readers may be expecting.)

18. Much as with internationalization a serious good faith attempt must be made at accessibility. We understand many projects just don't have the resources and/or expertise to do this right or even at all. But those that do must never treat accessibility as an afterthought.

19. Disturbingly often a FOSS project's built HIS just isn't available for offline study. Don't do that. It must be downloadable for offline use for

all FRTs. Period.

20. We absolutely encourage people to be versed in using things like web crawling and web mirroring software. But these skills absolutely shouldn't be necessary just to study an FRT.

21. In some cases of extreme interest instead of giving up we wrote custom web scrapers to achieve needed results. Sometimes we just weren't otherwise able to coax the various crawling software into getting more or less the exact set of pages we needed. Or we were but then still had to do a lot of offline scraping/cleaning to eliminate various cruft included in each of the hundreds of cloned pages which added up to significant storage space. The fact that we had to do this for something claiming to be open is absurd. FRTs must not force this pain on anyone.

22. Examples of said cruft include ads which are fine online. They're even encouraged when done ethically to help FRTs cover hosting costs and hopefully get actual well deserved revenue. Ads are forbidden in offline HIS versions.

23. The other cruft includes but is not limited to all sorts of extra decorations, headers, footers, and sidebars with navigation links to things on the broader project site like marketing fluff that most reasonable people wouldn't consider within the scope of the OKS let alone the HIS. This is not to say we hate/forbid things like decorations in the HIS content filled parts of the page. We do not in any way want to dictate expressive decisions made by the HIS implementers or otherwise constrain their freedom. We are merely lamenting how maddeningly hard it is to get a clean, cruft free copy of data that FOSS creators wholeheartedly wanted to be spread. A crufty website clone is not a valid form of the HIS for any FRT which has one.

24. Having discussed cruft/bloat let's explore in the opposite direction and examine how small we can go. An FRT can absolutely be just a single text file with some educational information and/or program source. It can even be a small text snippet in a comment on some random forum

post provided it can be cleanly copy-pasted in one go with very little scrolling to a text file or word processor document without cruft or formatting errors given the constraints of typical current text editor, word processor, clipboard, web browser, etc implementations. If the FRT's intended OKS is just the actual content of the message and not really the way it's styled/displayed nothing beyond the contents of the text snippet with the OKSE embedded at the beginning is necessary. If styling/display matters for proper experience of the material and there's some kind of nontrivial styling done any source material needed to produce it is logically part of the OKS. In that case it must be disclosed even if only as an adjacent FRTD compliant text snippet or a link to some self contained file.

25. While we're on the subject of needing to disclose any sources from which an FRT's HIS is made let's talk about how not to do it. We've seen cases where the built HIS for a FOSS project was essentially an offline website bundle. However inside that built bundle was also the source material from which it was built. If we were lucky it was all in a unified subfolder we could easily delete to reclaim persistent storage space. When less lucky the source files were side by side with the built HTML files which can be harder to get rid of. And of course this leaves us wondering in what instances we failed to notice the problem and both our network bandwidth and persistent storage space was being wasted. In our case life is short and storage/bandwidth is cheap. For many others stuck on old/modest systems with limited connectivity this is a showstopper at larger bundle sizes. FRT implementers must not put HIS source material into the built HIS unless they have very good reason.

26. Regarding the above perhaps this could make sense if generated HTML pages provided a quick link to whatever source file they were generated from for the curious. Obviously this wasn't the case in the situations we lamented and in our opinion the gains would be marginal even if it were. Perhaps those that wanted to customize their HIS copy with their own notes could benefit somewhat from the sources being right there for quick rebuilds. Even then this would require the presence of the HIS build tooling and the knowledge to use it.

Needless to say in cases where we saw stray HIS source material we found no sign of build instructions eliminating this theory as a logical explanation. But, even if we did find something, we feel like there are only marginal gains from this use case too. Any exceptions to the above must have much stronger justification than what we've historically seen.

27. We've seen many projects where the contents of the HIS as defined by the boundaries of what constitutes the project are not adequately enumerated and we can't reason about what parts of it we have nor what's even available. A failure mode to avoid by clearly enumerating the whole HIS as part of the OKSE.

28. The whole built HIS must be obtainable in one go/click/download for those who want just that and not the whole OKS purely for study purposes without say downloading and running the actual software. Again the HIS is everything outside of comments in the main program source and empty marketing fluff on the website. Educational elevator pitch type material is logically a part of the HIS's Ramp Up subset where the HIS properly exists and the larger OKS where it doesn't. The full HIS just by itself must be obtainable as one download. To minimize pain and bureaucratic overhead we allow a few exceptions. One is FRTs which are one source file with a lot of contiguous commentary at the beginning discussing design rationale and usage which is effectively an HIS. Forcing implementers to split this out into a separate file is pointless bureaucratic overhead. Another exception is made for very small FRTs that consist of multiple files/folders provided that the HIS can be trivially noticed and separated out even by a novice after downloading. Among other reasons this is for scenarios where someone wants to study an FRT but cannot run the software on the machines they have.

29. Some FOSS projects allow user comments to augment pages of the web based HIS. These comments often have good information in them. If an FRT implementer's project does this the augmented HIS must be available for offline study as well as the regular version. This could be from some kind of periodically generated or better yet near real time

snapshot. The same goes for things like officially maintained wikis on a both a wholesale and per article level as they are logically part of an FRT's HIS. Wikis administered by external parties outside the FOSS project should also strongly consider becoming FRTs.

30. A sufficiently large HIS should be broken into smaller useful (ideally disjoint) subsets that are easily downloadable in a targeted fashion. This way people with poor/limited Internet access and/or persistent storage space can still study the parts of the FRT they find interesting instead of being prevented from studying it entirely. (Specialized tools that can recover/repair from failed downloads can mitigate this but are still not enough.) This kind of breakdown may not always be possible especially in cases where there are a lot of tight interdependencies between the subsets. But a good faith attempt should be made once the system gets large. Often failure here means the FRT is poorly designed. But not always as the real world often has a lot of intrinsic unavoidable complexity. Similar principles apply to other parts of the full OKS.

31. Dependencies between any existing subsets of the HIS should be specified as formally and machine readably as possible. (Perhaps openly standardized methods of doing so will emerge or already exist in some obscure place.) This facilitates grabbing useful subsets via whole dependency trees both manually and in an automated fashion with various kinds of packages managers. Similar principles apply to other parts of an FRTs full OKS.

32. Dependencies between distinct FRTs should also be specified as formally as possible for similar reasons. Imagine being able to use a package manager pull in an FRT textbook whether by just by itself or also with the whole dependency tree of prerequisites one needs to understand it. Not to mention this can potentially save distributors and packagers a lot of headaches if done in a standardized/reusable way.

33. We've seen cases where the Ramp Up subset of the HIS was available for offline study but the Reference subset was not even though it very much existed. An example is standard library documentation for

programming languages. Don't do that. This may enable people to get started with a technology but will prevent them from progressing if they need to look up a nontrivial detail while offline.

34. We've also seen cases where the full Reference subset of the HIS was available for offline use but not the Ramp Up subset even though high quality ramp up material existed. This is catastrophic as Reference material is often useless if one doesn't have the conceptual background to make sense of it that the Ramp Up part provides. This is a prime example of Open Facts instead of Open Knowledge which is exactly what we must avoid.

35. Analogously sometimes we've seen cases where Ramp Up + Reference material is available offline on either the user level or developer/internals level but not both. Must avoid trap.

36. On occasion we had what we believed to be a full HIS for a FOSS project. Then we came upon something like an offline web page trying to source a video from somewhere online instead of properly including it in the offline HIS. Don't do this. It'd be one thing if it were some recommended external community resource. However here we're talking something that was clearly intended to be part of the official HIS. This problem has also manifested for other knowledge filled HIS material consisting purely of relatively simple things like text and images. Full HIS availability means full HIS availability.

37. Some FOSS projects have the full HIS available for offline study but make them difficult to find unless one is really looking on their website. Don't be one of them. We are so burned out by frequent lack of the offline HIS in the majority of FOSS projects that we may simply assume they're not there and go study, use, and contribute to another project which does this right instead.

38. There are scenarios where it's obvious that there is a complete HIS (or useful subset) available for offline study but the link to it is broken. Be sure this isn't your FOSS/FRT project dear reader. To get at the HIS the FOSS project intended to be as public as possible we sometimes had to

use the same kind of techniques as when conducting a penetration test searching for IDOR(Insecure Direct Object Reference) vulnerabilities. Penetration testing and hacking skills are something we strongly encourage everyone to have. But they should not be necessary even in their most basic forms to use/study an FRT.

39. We've also dealt with FOSS projects which used to have the offline HIS available but then no longer seemed to with newer releases. The HIS just wasn't obviously or nonobviously linked/visible anywhere on their websites. However the HIS was actually there if we used IDOR vulnerability hunting type techniques to find them. We really really wish we were making this up. All the hard work FOSS projects put into assembling the HIS was wasted for those users and contributors that don't have reliable Internet access, absurd amounts of patience, and penetration testing skills. Avoid this.

40. Then there are the FOSS projects that used to have a full offline HIS but then actually stopped instead of just seeming to as above. Another must avoid failure mode.

41. There are FOSS projects/packages where there's a full offline HIS and it's installed on a system by default. However when one invokes the help functionality of that project it either a) points to the online set instead of the offline set present on the system and/or b) fails to disclose the presence of the offline set at all. To add insult to injury sometimes the rather undiscoverable offline HIS is quite large and eats significant persistent storage space. The endlessly innovative ways in which FOSS projects manage to pull crushing defeat from the very jaws of victory itself never cease to astound us. Yet another failure mode to avoid and a small thing to change for a large impact. There needs to be an unbroken discovery chain from the main entry points of the HIS to any other part of the HIS.

42. It should go without saying that a properly designed offline HIS references the official online one for easy discovery of Sources Of Truth so to speak.

43. Sometimes powerful help systems that then correctly point to a robust local HIS exist but it's not obvious how to invoke said help system or even that it exists. Varying interfaces have varying constraints that may or may not allow broadcasting of the help system's existence in a clearly visible yet unobtrusive fashion. But a good faith attempt must be made.

44. We've seen situations where there's a full offline HIS consisting of very solid Ramp Up and Reference material, it's obvious how to invoke the generic main help functionality, but then the help points us to somewhere in the Reference or Late Ramp Up parts of the material. Frustratingly enough there exists good Early Ramp Up material for basic mental model building and pointers to the other available help. But the system still failed to point to it. FRTs must avoid making that mistake. For FOSS projects this is another tiny fix for humongous gains.

45. With FRTs the most important thing for implementers to focus on is capturing as much of the knowledge they want to share as possible into the OKS without being overly concerned how that's distributed amongst the set's parts. That said the OKS parts should be kept as uncoupled/as reasonably possible. We've seen FOSS projects where the text files making up the HIS were embedded into the built executable and couldn't also be read by other text viewers/editors present on the system. Avoid coupling like this.

46. An FRTs offline HIS must be published in at least one standardized open format. (Or better yet FRTD compliant format. This requirement will likely be tightened in future FRTD versions as FRTs hopefully become more widespread.) If the offline HIS is a bundle consisting of many different kinds of files at least one of the bundles must consist purely of open formats.

47. Most real life systems including FRTs have both direct/indirect dependencies/subsystems/subcomponents they're built from. Ideally FRTs would be built from the ground up purely in terms of other FRTs. Such an FRT is defined to be a Rank 1 Freedom Respecting

Technology. R1FRT for short.

48. A Rank 2 Freedom Respecting Technology (or R2FRT for short) is defined as having only FRT and non-FRT FOSS dependencies.

49. A Rank 3 Freedom Respecting Technology (or R3FRT for short) can have only FRT, non-FRT FOSS, and source available dependencies.

50. A Rank 4 Freedom Respecting Technology (or R4FRT for short) is anything that doesn't qualify as an R1FRT, R2FRT, or R3FRT.

51. Optional build or execution time dependencies impact the above mentioned classifications based on the least freedom respecting component actually used.

52. When talking about an FRT it can occasionally be prudent to state what version of the FRTD they comply with and/or what Rank an FRT has to better assist people being able to reason about what kinds of freedoms the FRT in question affords them.

53. The above mentioned taxonomy is likely oversimplified and still subject to change as our understanding of the problem space improves. So don't take it too seriously or use it too often. It should be good enough to give useful shorthand in conversations about degrees of (un)freedom. But it's not a substitute for deep nuanced conversations.

54. Suppose one implemented an adapter for the interface of a mere FOSS or totally unfree external system to make using that system more comfortable. Since the external system is not logically part of the adapter's implementation it doesn't prevent the adapter from being an FRT. Adapters should be clearly identified as such and clearly specify the external systems they are wrapping. All other build and execution time dependencies of the adapter proper must of course be FRTs.

55. Use sane default (file) names for things like projects, the offline HIS, etc so that they can be searched for in an automated manner on one's computer. FRT implementers should avoid crazy characters that cause

issues in things like shell commands. We're tired of trying to figure out how to name a file holding the HIS for the TLA*\F+ project so that we actually have a chance of being able to find it later.

56. Many FOSS style works of research are structured as a crufty yet still not properly savable blog post. Our attempts to save it with our browsers pulled in both extra cruft and failed to pull the educational demo videos embedded from some external video sharing site. Linked data sets and program sources needed to reproduce the research also obviously don't get saved. It's fine for FRTs to expose research this way. But there must be a cruft free offline bundle with the full OKS obtainable in a very small number of downloads (ideally one) for all FRT works of research.

57. Given how much we discussed the HIS component of an FRT's OKS we want to emphasize that the FRTD is not necessarily a call for a bigger HIS. It's a call to think system wide about the whole OKS. It's a call to ensure that the full existing OKS is adequately open, discoverable, obtainable, retrievable, and usable offline. It's a call to make sure the whole OKS remains so as new components that logically belong there come into existence or old components are enlarged. This is the bare minimum prerequisite needed to practically ensure the freedoms our predecessors intended.

58. Again, the FRTD is not a call for a bigger HIS / more documentation. Sure, there are times that can help, there are times it's arguably necessary. More often than not, however, we believe it to be counterproductive. The main focus for implementers should be thinking system wide and capturing as much of the knowledge they want to share as possible into the OKS via the full multitude of channels available to them. Often narrative documentation is a deeply suboptimal channel for this knowledge capture. Real talk. Virtually nobody likes writing this documentation, let alone maintaining it. Virtually nobody likes reading it either, necessary as that may often be for them to truly study, use, or further develop a system. Mainstream approaches to documentation simply do not scale. We've seen FOSS projects with documentation quality most would agree is in the top 1%

of what's out there that still had nontrivial omissions or misleading inaccuracies. Here's what we believe to be a more tenable approach. First make every part of the system as self describing as possible whether that's program sources, UI, or whatever else. When the documentation inevitably fails it's extremely desirable for the sources of truth about the system on both a user and developer level to be as clear and self explanatory as possible. After that there will still be plenty of cases where system components are reasonably self describing but the system as a whole is not. Documentation will still need to be written and maintained to close that gap, but on a much smaller scale than would be needed otherwise. All this documentation needs to do is empower people with a big picture overview and some context on where to start digging around in the otherwise self describing components. This is a much more attainable objective. Furthermore, many things that were/are traditionally mostly/fully narrative prose documentation very arguably shouldn't be, especially in security contexts where ambiguity is dangerous, even more especially in the contexts like security protocol specifications. Perhaps in the future AI may make mainstream documentation more scalable, but until then we leave readers with these anecdotes to think about.

59. Much as we wish this went without saying, identifier/variable/symbol names are a valuable channel via which to transfer knowledge in the context of software development. We've seen way too many FOSS projects that used cryptic identifiers that made studying the system very hard in spite of us having a good grasp on both the programming language and problem domain at hand. This would be perfectly fine if that's how those developers legitimately thought of the concepts behind those identifiers in their head, but that's almost never actually the case, which leads us to question what kind of openness and knowledge sharing is that? Identifiers in program sources must reflect how the developers think of the concept behind them internally. Exceptions can be made in some very specialized scenarios. For example, perhaps a program is being developed on or for a heavily resource constrained system that struggles to fit source files into memory. In that case it can be reasonable to shrink down identifier names even if they start bordering on cryptic. Or perhaps the program

sources are being written for some legacy system whose programming language only permits short identifier names. Generally, even in short (sub)programs of say five lines where the identifier has limited scope, identifiers should model how the developers thinks about them internally. Let's not kid ourselves, that five line scope often has a way of becoming a 500 line scope.

60. As AI is being used to generate various kinds of works in whole on in part we require that FRTs disclose all nontrivial parts of them that were generated with AI. We furthermore require that the AI system used for the generation be disclosed with as much specific information as is reasonably possible like say the version number of the AI system/model and the exact prompts/inputs used to trigger the generation. Just as disclosing program sources allows people to learn new programming techniques this will allow people to learn AI usage techniques they may not have been aware of prior. Withholding this information is fundamentally no different than withholding part of the main program sources or publishing intermediate sources output by some preprocessor. If available to FRT authors under terms/circumstances where they can reasonably share them the sources and methodologies powering the AI system used for generation must be disclosed for all it's aspects including model architecture, data collection, data preprocessing, model training, model validation, and inference. Furthermore parameters the trained/optimized AI model used during generation as well as the training data leading to those parameters should generally be disclosed. FRTs must not disclose parameters or training data in scenarios where the AI system was trained on people's private and/or otherwise sensitive information unless explicit permission to disclose is granted by the people whose information powers the model.

61. If the FRT at hand is itself some kind of AI system similar principles apply. Sources and methodologies powering the AI system must be disclosed on all fronts including model architecture, data collection, data preprocessing, model training, model validation, and inference. The trained/optimized AI model parameters and the training data which led to them should generally be disclosed. Again FRTs must not

disclose these if the AI system was trained on people's private and/or sensitive information unless explicit permission is granted by the people who it pertains to. Even if this permission is granted seriously consider using techniques like differential privacy before disclosing the data, disclosing the model parameters, or training the AI model.

concluding remarks

Hopefully after having read thus far it should now be obvious that many technologies even though compliant with the various FOSS definitions still fail to provide practical technological freedom. A newer clearer vision for technological freedom was long overdue and this is our attempt to rectify the situation.

In some cases the problems mentioned here are deliberate. Some FOSS projects are frankly just bait by predatory entities to tempt vulnerable targets. These targets could be idealistic people. These targets could be strategic thinking organizations trying to maximize their options by avoiding unfree systems locking them in. Either way they wind up trapped in expensive commercial "support" contracts instead.

To be clear we have nothing against entities selling FOSS products and/or support for them. We all need to make a living. Our issue is with so called "open" systems not providing us the tools to help/serve ourselves when clearly they (almost) exist or could exist with some not very labor intensive modifications.

This document is not for the makers of the predatory FOSS projects mentioned above. This is for the many FOSS makers that seem to have no visible/obvious bait profit motive, that hopefully have some kind of healthy profit acquisition capabilities, that have a genuine enthusiasm for open knowledge, and that have a genuine desire to make the world a better place. It's here to make explicit the things they accidentally/misguidedly omit/misprioritize.

We hope that this document inspires the creation of many FRTs right from the start, independent evolution of FOSS projects into FRTs driven by their

developers, and also such evolution driven by user request. We hope organizations trying to bolster their public image and/or win developer goodwill start developing proper FRT initiatives instead of just mere FOSS initiatives.

Are the requirements stated herein more work for creators? Quite possibly. Though many FOSS projects are much closer than they think to being FRTD compliant and only need to make quite minor corrections for major benefit.

Aren't implementers already often horribly badly overworked, underresourced, and stuck dealing with entitled users? Very much yes. Sadly the primary author is ashamed to have been among those entitled users at points in the past.

So why then should FOSS projects aim for FRTD compliance when they have limited resources to invest? Often when one has limited resources and a big vision they wish to execute the best use of those limited resources is conducting activities that will bring in more resources until there are at least sufficient resources for the situation at hand.

How do contributors to FOSS projects come about once they discover the project by some means? They try meaningfully studying it beyond the initial marketing fluff to see if it's suitable for their objectives whether directly or with some reasonable amount of customization work. Then once they've studied enough to meaningfully use the project and do so for a while they get a deep gut feeling of its current behavior/workings and a sense of the gap between those and the desired behavior/workings. At that point they finally know enough to contribute meaningful fixes that the project maintainers would likely find useful and incorporate. After some time if there's a consistent track record of contributions they themselves may be promoted to a maintainer if that's what they want. This is what the FOSS contribution pipeline often looks like.

Often what happens to this pipeline in non-FRT FOSS projects is that it gets disrupted very early on by newcomers not being able to meaningfully access/study the full Open Knowledge Set in a reasonably self sufficient

manner let alone use it, further develop it, and contribute those developments back. FRTD compliance should be prioritized over other often less important feature work because True Open Knowledge is the meta feature that builds the community around a project and thus helps scale the resources closer to what's really needed for other development. The alternative is a death spiral of burnout once resources run out.

Open the knowledge and the builders will come.