Pollyanna: A Hyper-Accessible Hyper-Compatible Web Framework

April 1, 2022

Ilya Gulko <thankyou@mit.edu>

Winner of MIT SIGTBD 2022 (N-1)-way tie for Best Paper!

Google Bard and GPT-4 contributed to this paper.

Pollyanna was formerly named sHiTMyseLf.

Introduction

Pollyanna is a Web frontend and backend framework for creating websites like blogs, forums, and online community spaces. It includes basic pieces for user identities, user-owned items, and mechanisms for labeling, listing, linking these things together. This paper talks specifically about the challenges and solutions to making websites built with Pollyanna accessible and compatible, to minimize the chance of a visitor experiencing a problem with accessing it.

We want to reduce the difficulties visitors might confront when accessing sites constructed with Pollyanna. The genesis of Pollyanna was inspired by the need to address prevalent frustrations associated with accessibility in existing online community spaces. In the course of its creation, we have selectively adopted the foremost features and concepts already in practice within these virtual communal spaces.

We want to create all-inclusive digital environments that parallel the physical world's integration of wheelchair ramps and elevators – it's the metaphorical embodiment of our approach to building universal accessibility. While these installations often accommodate but a fraction of the population, their necessity is acknowledged universally. Wheelchair ramps and related accessibility devices also create unexpected benefits for others, such as people with limited mobility, those using carts to move heavy items, and people with child strollers.

Ideally, we want to create for the user an experience of being eagerly accommodated and catered to, not unlike when visiting a luxury hotel or resort. A corner store or gas station may turn you away without shoes, but if you were to appear without shoes at a five-star hotel, you would probably be offered slippers.

We want to create for the user an experience of being eagerly accommodated and catered to, similar to that of a luxury hotel.

Similarly, the philosophy that guides the design and functioning of the Pollyanna framework encompasses the smooth accommodation of various devices, browsers, and configurations, irrespective of whether they constitute an extra challenge or represent a minor demographic.

The ambition behind our focus on accessibility resolves to diminish any potential discrepancies a visitor might confront while accessing websites built using Pollyanna. With this paper, we aim to present factual and practical insights into the problem-solving approach we've adopted to make our framework as accessible and compatible as possible.

Our primary desire is not to market Pollyanna, but rather to inspire a more inclusive approach towards website development. We aspire to demonstrate that Pollyanna offers a practical solution to the pressing problem of digital accessibility while yielding measurable and tangible results. We present you with an introspective look into Pollyanna – not as a sales pitch but as an informational guide and an invitation to help build a more inclusive digital landscape.

Motivation and Philosophy

In creating Pollyanna, the motivation was to address all the major accessibility frustrations we experienced while using existing online community spaces, and adopt all the best features and ideas. We believe that everyone should have equal access to the information and resources that the Web has to offer, regardless of their abilities or devices.

We propose that, similar to the Cool URIs principle, changing the interface or service mechanics of a community website without user consent should be avoided. This is because such changes can be disruptive and confusing for users, and can erode trust in the website.

Changing the interface or service mechanics of a community website without user consent should be avoided.

It's much easier to write a website for someone with abilities similar to ours, and it's impossible to predict every combination of abilities for every user of a popular resource. This is why it's so important to design websites with accessibility in mind. When we make assumptions about our users, we risk excluding those who don't fit neatly into our preconceived notions.

As a developer-designer interested in maximizing accessibility it can be a mistake to make assumptions about the user, their access device, and their ability to configure it. However, such assumptions can also create useful constraints. For example, by assuming that users are interested and motivated to access the resource, we can focus on making the website as easy to use as possible.

In the case of Pollyanna, the assumptions we have made are that the user is interested and motivated to access the resource. We have also assumed that the user has no choice over their device, browser, or configuration, but is technically literate (or is aided by another, who is technically literate), and motivated to dig deep to seek workarounds for these limitations. These assumptions allow us to focus on designing a website that is as accessible as possible to everyone, regardless of their circumstances.

While we have implemented a prototype localization and internationalization feature, most of the interface has not yet been translated. Therefore, we currently assume that users have a working knowledge of English.

Accessibility and User Experience Features

In the development of Pollyanna, great emphasis has been placed on ensuring expansive accessibility. The particulars, as discussed below, have been primarily influenced by an array of user scenarios that we encountered during the development process.

Firstly, for users leveraging older devices or browsers, we prioritized a minimally intensive interface, ensuring the compatibility of our platform with older technologies. Care was taken to progressively enhance advanced features and exhaustive testing was performed on an array of older devices to maintain optimal performance. This also allowed for accessibility via low-resource browsers like Dillo and

NetSurf.

Understanding that some users may choose to disable JavaScript support in their browsers, for both security and accessibility reasons, as some devices may not be as performant with JavaScript enabled, especially across the wider web, we painstakingly guaranteed that all vital features function smoothly without the need for JavaScript.

Understandably, not all users have access to high-speed network connections. To address this, we tested with throttled and generally slow connections, minimized our code size, and introduced an additional low-bandwidth mode.

For the keyboard-centric users or for those who lack the use of pointing devices, we ensured that all page elements were accessible via keyboard input through tab stops or shortcut keys. This was validated with keyboard-specific browsers such as Luakit, qutebrowser, and w3m.

We ensured that all page elements were accessible via keyboard input through tab stops or shortcut keys.

Recognizing unforeseen scenarios such as unavailability of keyboards, an on-screen keyboard is included within our own platform, to accommodate cases where the device's own on-screen keyboard malfunctioned or was unavailable.

In the cases where users had a sentimental inclination towards certain devices with personal histories, we performed extensive testing on a diverse array of browser versions.

Recognizing the needs of visually-impaired users, we carried out testing with screen readers under the guidance of visually-impaired users.

Keeping inclusivity in mind, we also catered to users without personal devices by performing copious testing on public or borrowed devices. We would like to extend our gratitude for the generous device libraries from Verizon, AT&T, Best Buy, Apple, T-Mobile, Sprint, and Cricket.

For users with limited network access, reliant on public WiFi service, we engineered capabilities for offline browsing and content preservation, ensuring offline drafts for later submission retained composition and timestamp.

Acknowledging restricted network environments that block direct resource access, we facilitated service migration, data download for offline usage, and proxy service access to ensure network accessibility challenges did not hamper user experience.

In response to non-private environmental scenarios, inconspicuous designs were created, allowing the user to conceal their username and logged-in status, promoting user comfort and conversation.

Through every step of our development process, consideration of users, their unique circumstances, and their varied devices has been paramount in the creation of Pollyanna, resulting in a platform encouraging accessibility and fostering a user-friendly experience.

To ensure accessibility even in cases when the original resource is unavailable, such as when domains expire or servers go offline, we have designed Pollyanna to be friendly to archiver services such as the Internet Archive.

Language and Cultural Considerations

Understanding that the user might not have literacy capabilities or the ability to read in a particular language, we have introduced unique solutions to overcome these challenges. First, we integrated a basic string substitution localization feature. This feature allows individuals who can read in other languages to effortlessly navigate the interface. This feature is currently in a prototype stage.

For those unable to read text, we incorporated an innovative emoji-based language setting. This alternative language setting is currently in a prototype stage, but it has the potential to significantly expand the framework's accessibility for users who find traditional text-based navigation challenging or impossible.

However, the challenge remains to support users with older devices that aren't compatible with Unicode and cannot read languages supported by ASCII encoding. Addressing this obstacle is a noted concern, and efforts are ongoing to implement a feasible solution to cater to this user group.

Further accessibility measures have been undertaken to accommodate users that don't have access to a graphical environment. Extensive testing with text-mode (console) browsers such as Lynx, Links, and w3m has been performed. This ensures the framework's functionality and accessibility irrespective of the graphic capability of the user's device. We also tested the ability to archive content offline for later reading using tools such as wget and curl.

Pollyanna also addresses the varying needs of users regarding their choice of input character sets. We have added the facility to switch input character sets within the web page. It includes a 'translit' mode for Cyrillic entry and a Dvorak keyboard layout. This feature can be accessed either by remapping keypresses or by using the on-screen keyboard. This functionality requires JavaScript, but it has the potential to be implemented server-side as well.

In the architecture of a hyper-accessible and hyper-compatible web framework like Pollyanna, every accessibility challenge presents an opportunity to push the boundaries of what's possible. While it's simpler to cater to only a majority, the true accomplishment lies in addressing the diverse combination of abilities of potential users. As our framework evolves, we continue to learn, adapt, and innovate, keeping accessibility at the forefront of our development agenda.

Every accessibility challenge presents an opportunity to push the boundaries of what's possible.

Evolutionary Development

In order to support these diverse access scenarios, we decided on a development approach that mirrored the evolution of the Web itself. We chose to do this to exploit the Lindy Effect, which states that a technology is likely to be supported for a time which is proportional to how long it has already been

around. Thus, we chose the longest-supported portions of HTML and JavaScript for use in this project.

The base features were first developed for universal compatibility with older browsers, including Mosaic, Netscape, and Internet Explorer, and only then enhanced with JavaScript and stylesheets.

Testing with these browsers was performed throughout the development process to ensure that backwards compatibility was not lost.

Encouraging a Safe Environment

A key component of accessibility is the user's sense of safety and acceptance when visiting a website. If a user feels unsafe, threatened, attacked, or at risk, the website is no longer accessible to them.

A website can encourage a safe environment in two ways. One is by encouraging positive contributions from users. This can be done by highlighting existing good content, providing clear guidelines for acceptable behavior, creating a positive and supportive community culture, and rewarding users for positive contributions.

A key component of accessibility is the user's sense of safety and acceptance when visiting a website.

The other way to encourage a safe environment is by making it easy to flag or report negative contributions, and potentially hiding them from view until a moderator can respond. This is a form of community moderation.

Pollyanna allows for community moderation to have multiple layers, by allowing any user to flag or report a post, but also allowing these actions to be audited by trusted moderators and operators. This mitigates possible abuse of the moderation features.

In addition to the benefits mentioned above, community moderation can also help to improve the quality of content on a website, and to make it more attractive to users.

Enshittification Resistance

The term "enshittification" was first introduced by Canadian-British blogger, journalist, and science fiction author Cory Doctorow in 2022. It refers to the process by which a system or product becomes progressively degraded over time due to flawed design or implementation.

One of the key goals of Pollyanna is to be "enshittification resistant." This means that Pollyanna is designed to discourage modifying website design and mechanics non-consensually. One way Pollyanna achieves this is by making it easy for users to port their profiles between different instances. We also designed features that encourage kindness and community health, such as semantic voting and labeling. For extreme and time-sensitive situations, we designed an “emergency stop” feature, which allows for pausing the site by a user.

Pollyanna profiles are backed by public key infrastructure (PKI), currently using GnuPG and OpenPGP integration, which means that they can be easily verified and trusted by other instances. This makes it easy for users to move their profiles to a new instance if they become unhappy with the current one. It

also allows for an instance to split into several sub-communities. Pollyanna is designed to be modular, allowing for the substitution of other PKI systems, such as OpenSSH, OpenSSL, Bitcoin, Ethereum, and Dogecoin.

The portable data dumps, essentially an “insurance policy” for the users, allows cloning of an entire community instance, and protects users even from non-consensual changes put into effect by operators and administrators, such as non-consensual changes to the design, introduction of advertising, limiting of information access, and opaque changes to information filtering.

Taking advantage of this insurance does require the users to make regular backups of server content.

Testing Methodology

Testing is essential to ensure that both the interface and the backend are usable and to maintain backwards compatibility. This means testing the framework across a wide range of devices, browsers, and accessibility needs. We tested the web framework to ensure that users could understand the interface, complete common tasks efficiently, and access it on a variety of devices and browsers. Two lenses of testing were adopted:

- **Task-based user studies**: We recruited new users to perform common tasks such as registering, sharing content, uploading, searching, voting, and replying. The user interface was then refined based on observed difficulties. In user studies, participants were either given a device or used their own. They were then required to perform a series of tasks, while we observed their interactions and later made interface adjustments wherever difficulties were experienced. The users were typically asked to open the website, register for an account, find some content and read it, vote on the content, post a reply, and post their own content. The order of the tasks was randomized to ensure that they could all be done in any order.

- **In-depth testing by a knowledgeable user**: A developer performed actions on real-world devices with certain limitations. This covered a wide spectrum of devices and scenarios to ensure comprehensive accessibility and compatibility. A broad scope of devices and scenarios was considered in the testing, including older handed-down devices, public workstations, library computers, LinkNYC kiosks, airport kiosks, office center computers, hotel computers, found devices, friends’ computers, devices with limited website access, web-enabled TV units, flip phones with small displays and T9 text entry, and e-ink devices like the Amazon Kindle and B&N Nook. We covered a wide range of operating systems, screen sizes, and network speeds.

In addition to this, we also performed testing from an operator’s perspective:

- To ensure access to users without access to a graphical environment, we performed extensive testing with text-mode browsers like Lynx, Links, and W3M.

- We regularly test the installation and upgrade process on a variety of operating systems, including Trisquel GNU/Linux, Ubuntu, Debian, Raspbian, Fedora, FreeBSD, LinuxMint, macOS, and Windows.

- We test all configuration options to ensure that they work as expected, such as changing themes and toggling features like JavaScript integration. We also test customization features, such as re-templating of pages.

- We regularly test the moderation and auditing features to ensure that they are easy to use and effective, and understandable and transparent.

- Due to the closed and restrictive nature of iOS and Android, we have not yet been able to perform meaningful backend testing on these platforms, but we have achieved promising results with running Perl and PGP tooling on both.

- Operators should be able to revalidate the notarization chain and all referenced messages and votes, and to query the index database in-depth to investigate anything they may find interesting or concerning. They should also be able to filter the content and to annotate it both publicly and privately.

- Operators can easily create a complete or partial clone of an existing instance of Pollyanna, including user accounts, threaded discussions with nesting, timestamp values, notarization chain integrity, valid public keys, and signatures. Archives should be easy to transport and synchronize using the operator's preferred method, such as zip and upload, RSS, CSV or PSV files, a version management system like Git, or sneakernet.

Conclusion

We have discussed the challenges and solutions to making websites built with Pollyanna accessible and compatible. We have outlined a development approach that mirrors the evolution of the Web itself, and we have described a testing methodology that ensures that accessibility is built in from the start.

Pollyanna is a valuable framework for creating accessible and compatible websites. It is a powerful tool that can be used to create websites that are accessible to everyone, regardless of their abilities, including physical abilities and mobility, technical literacy, and the unique capabilities of their device and browser.

We would like to see Pollyanna adopted, used, and improved upon for building websites. But more than that, we encourage Web developers to reconsider their accessibility and compatibility strategies. As the Web becomes ubiquitous and access to Web sites a necessity, we can no longer allow ourselves to leave it to the user to provide their own accessibility. We must bend over backwards to accommodate them, simply because we have this option at our disposal.

As the Web becomes ubiquitous, we can no longer allow ourselves to leave it to the user to provide their own accessibility.

Our experience shows that, just as with wheelchair ramps, addressing one use case or demographic that is quantifiable also helps many other use cases that are not anticipated ahead of time. By making a website more usable for one demographic we made it more accessible for many others that we had not thought about ahead of time.

Acknowledgements

This paper wouldn’t be possible without significant help with proofreading from Google Bard.

OpenAI GPT-4 was also used for some composition with scripting help by Kliment Serafimov.

Additional Reading

Pollyanna on GitHub
https://www.github.com/gulkily/pollyanna

sHiTMyseLf.com
http://www.shitmyself.com/
https://www.shitmyself.com/

Librarian's Letter to Google Security (docs.google.com)
https://news.ycombinator.com/item?id=32304320

How far back in time can I take my website's design? (ajxs.me)
https://news.ycombinator.com/item?id=37505798

Designing content for people who struggle with numbers
https://service-manual.nhs.uk/content/numbers-measurements-dates-time

Lindy Effect
https://en.wikipedia.org/wiki/Lindy_effect

Usability of Old Computers
https://archive.is/https://blog.karthikkumar.org/usability-of-old-computers-f06a7f00f72e

Cool URIs don't change
https://www.w3.org/Provider/Style/URI

Hacker News Guidelines
https://news.ycombinator.com/newsguidelines.html