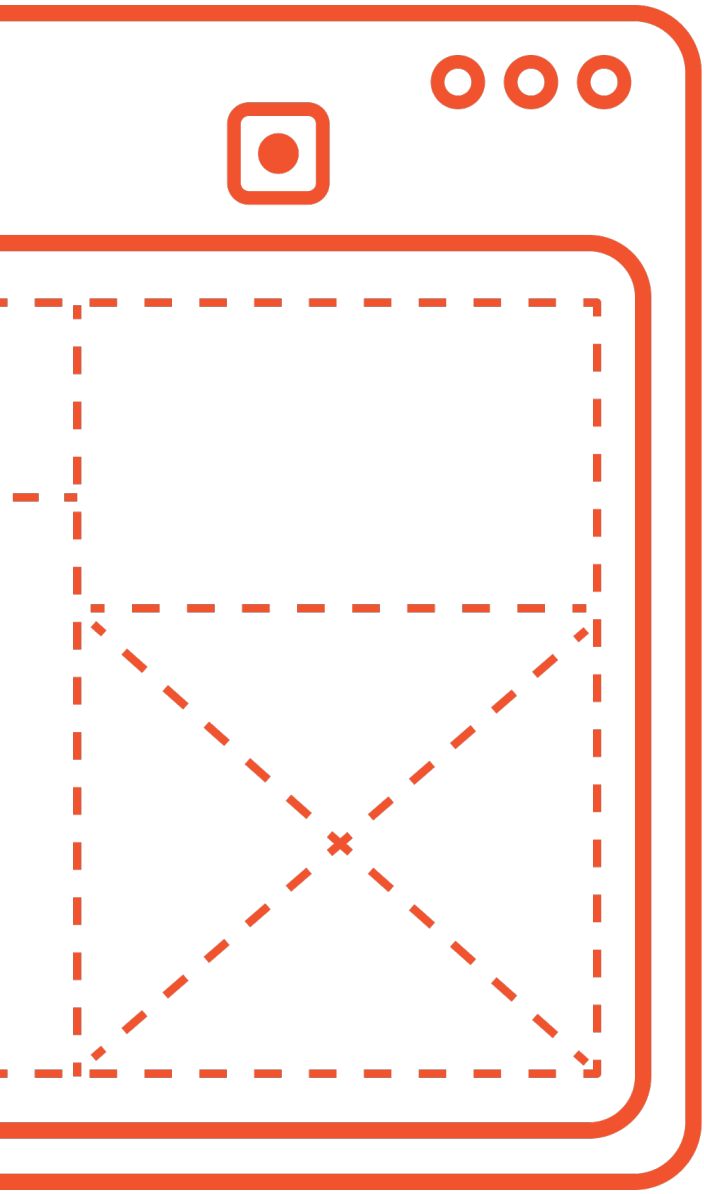
hidden visuality of the web



Under deconstruction

Christine Brovkina

Table of contents

Becoming user: introduction 4

Website is a lot of things: approaching the notion of a website 10

- Website is an utility 14
- Website is an experience 19
- Website is a place 21
- Website is anything 26
- Website is a tree 28
- Website is never ready 35
- Website is a webpage 40

Looking through the window: on relationship between browser and website 44

- Browsing context and rendering process 47
- Cash runs everything around me: browser wars and the power of default 51

Exposing the wires: browser extension as a tool to understand what a website is. 68

Browser extensibility: from bookmarklet to the browser extension 69

Anatomy of a browser extension 73

Changing the perspective 78

Little sharp tools: browser extensions documentation 84

Borderline Explorer 85

Hide&Seek 87

Headspace 89

Field Expedition 90

Resources 92

Colophon 95

Becoming user: introduction

Over the last decades the Internet stopped being the exciting new medium and became ubiquitous, almost invisible for many. The way we browse the internet also changed — old web metaphors like "surfing" don't make sense anymore and serendipity navigation is almost extinct with platform-bound curated bubbles of interest owning and shaping our relationship with the web. This shift in our relationship with the Internet has not gone unnoticed by scholars and critics. Tiziana Terranova[1] sees the Internet as *residual* technology, still "an effective element of the present", but less legible and intelligible than it used to be, effectively subsumed by 'Corporate Platform Complex' — complex of privately owned online services that call themselves "platforms". These platforms are digital "gated communities" with strong ownership of data, software and infrastructure. Standards and protocols developed as a part of the project of creating the Internet as a public and open network still operate, but they are increasingly buried under a thick layer of corporate ones. Louise Druhle[2] builds on this concept, inviting us to imagine to-

[1] Tiziana Terranova. "After the Internet. Digital networks between capital and common", Semiotext(e), 2022.

[2] Louise Drulhe. "Critical Atlas of Internet", https://louisedrulhe.fr/internet-atlas/

day's Internet as a surface, with major platforms (such as Google, Meta, or Amazon) acting as centers of gravity. These platforms are deeply embedded in the web's surface, creating steep slopes that pull our online activities downward in a form of digital drift.

In order to be able to move forward, sometimes it is necessary to look back — how did we get here, how did we became “users”? Olia Lialina and Dragan Espenschied[1] believe that there seems to be little time for deep reflection on relationship between computer and user in between the total neglect of computers to their total adoption and ubiquity. They describe how role of “user” changed over the time: from the deeply personal, almost religious in nature, depicted in 1982 movie ‘Tron’ to the cynical view that is still perpetuated today — “users” as incapable, needed to be constrained and highly entertained and also continuously exploited as content producers and ad-clicking revenue generators.

[1] Olia Lialina, Dragan Espenschied. “Digital Folklore Reader”, merz & solitude, 2009.

During the 1990s despite growing the separation between so called “hackers” and “users”, each forming its own native subcultures, whereas the culture of amateur web empowered people to build the Internet by building

their own web pages. As computer and Internet access became more widespread, paradoxically, fewer people recognised the value of their own contributions, making it increasingly difficult to reflect on the medium itself.

Professionalisation of contribution and the "new economy" in the late 1990s —early 2000s while presenting platformisation as progressive and revolutionary, showed the lowly users their place again — amateur web practices became an object of countless jokes, professional web designers and developers started to compile lists showcasing "worst websites ever", reinforcing the notions of "a good website" and "user-friendly interface", which *of course* needed to be designed, developed and maintained by professionals. This turn, as Tiziana Terranova[1] points out, was preceded by the exponential growth of internet activity since the early 2000s — a mass sharing, uploading, posting and discussing of content that took place initially mostly by the means of non-proprietary software and platform capitalism can be seen as a reaction to the mass participation.

Describing the emerging new role of the Internet "user" Terranova[2] writes:

[1] Tiziana Terranova. "After the Internet. Digital networks between capital and common", Semiotext(e), 2022.

[2] Ibid.

> Over the time, the user has morphed from master to addict, as behaviourist interfaces that are designed with the purpose of maximising engagement corrupt collective intelligence by facilitating the spread of fake news, conspiracy theories and hate speech. Instead of the "hacker", the "influencer" has become the new heroic figure, the focus of subjectivation.

The internets "back to the land" movement is one of the reactions to this inequality. It reflects in counterculture called "hand-made web". J.R. Carpenter[1] first evokes the term "hand-made web", stating that in today's highly commercialised web of multinational corporations, proprietary applications, read-only devices, search algorithms, Content Management Systems, WYSIWYG editors, and digital publishers it becomes an increasingly radical act to hand-code and self-publish experimental web art and writing projects. He writes[2]:

[1] J.R. Carpenter. "A Hand-made Web", https://luckysoap.com/statements/handmadeweb.html

[2] Ibid.

> The more proprietary, predatory, and puerile a place the web becomes, the more committed I am to using it in poetic and intransigent ways.

This sentiment is echoed by other proponents of the hand-made web movement. Becca Abbe[1] also stresses that it is important to demystify the inner workings of the web so that individuals may regain control of the tools that build it. It is

[1] Becca Abbe. "The Internet's Back-to-the-Land Movement". https://www.are.na/editorial/the-internet's-back-to-the-land-movement

absolutely necessary today to re-evaluate the status quo and help give users the power back. Olia Lialina[1] writes: “Users must understand their integral role in the process, demand comprehensible systems, work for better computer education, begin to see themselves as developers again.”

[1] Olia Lialina, Dragan Espenschied. “Digital Folklore Reader”, merz & solitude, 2009.

Inspired by these ideas and the apparent need for re-negotiation of individual’s role on the web, I've decided to embarked on a project that aims to contribute to this movement. With this project I want to scratch the slick two-dimensional surface of a website, exposing some of its wires in hope to contribute to the plan to truly reconnect users and developers, restore reasonable relationship between people and their favourite medium — the web.

A poet, essayist, lecturer, and environmental activist Gary Snyder once said, “The most radical thing you can do is stay home”. In the context of the Internet, the most radical thing you can do on the web today is to betray the platform, build and run your own website.

Website is a lot of things: approaching the notion of a website

What is a website? This question could come from a curious 5-year-old or even an inquisitive alien trying to understand human technology. It is common knowledge today what a website is. I saw some websites today and so most probably did 5.44 billion people worldwide in April of 2024[1]. There seems to exist some common assumption of what we wean, when we say 'website'. Then why ask it again?

Total disclosure now — this is a pseudo-naive question. The one that is asked as if the asker doesn't know the answer, even though they likely do (or think they do to a certain degree). It often appears trivial or even borderline stupid but is intended to provoke deeper thinking, reflection, or discussion about a topic that is assumed to be well understood. For me this a perfect starting point, it cannot get any better.

Defining once and for all what a website is and what it is not seems like an elusive task. The

[1] As of April 2024, there were 5.44 billion Internet users worldwide, which amounted to 67.1 percent of the global population. Of this total, 5.07 billion, or 62.6 percent of the world's population, were social media users.

"Digital 2024: April Global Statshot Report", https://datareportal.com/reports/digital-2024-april-global-statshot

very flexible and ever-evolving nature of this medium resists the rigid terms of a singular, fixed definition. Websites can take on many forms and serve a multitude of purposes, from simple informational pages to complex, interactive applications. This diversity makes it challenging to pin down a concrete definition that encompasses all possible variations and uses of websites. But what we actually can do, is try to approach the notion of a website through all the different views on the matter we may encounter. Whether one started surfing the web in the 1990s or joined the Internet-party later, whether one have been building and deploying web pages from scratch or just actively posting on social media platforms, whether one has a background in design, marketing, software development, or claim to have no technical skills at all, whether one is relying on assistive technology to access the web or just use the pre-installed browser on particular device — all these experiences shape our notion of the web, and in turn, we shape the web to fulfil our expectations of what a website should look and feel like, creating an endless feedback loop. In this chapter I would like to take a look at several perspectives to what a website is and to reflect on what those definitions may reveal about our ideas and expectations.

“A website is a construct based on code and assets that empowers the visual output to serve as an open space.

It is a digitally accessible representation and can be a tool like an application or shop, a (non-linear)book telling a story, or a collection of (multimedia) resources — but also an artistic practice, free-spirited to the point of being completely useless.”

The web has developed from the “World Wild West” with just a small amount of impact on our daily lives, to a more unified but at the same time hybrid place surrounding us permanently. However, in the age of social media, the role of websites has evolved significantly. While they still establish brand identities, showcase work, products, and services, and attract or bind (potential) customers, many people are now primarily active on social media platforms, so a referenced website often only serves as a business card and proof of credibility. This is particularly true for small businesses and individuals.
Nevertheless, websites still hold a crucial niche, providing a space where detailed information can be shared and a deeper connection with the audience can be established. They enable global access and exchange of information.

Jakob Grommas und Victoria Dietz

grommas-dietz.com

Website is an utility

The first approach I want to examine is what I call the *utilitarian* perspective. It focuses on how a website serves its purpose, how web pages are organised and connected, and how users navigate through the site. The Mozilla Foundation, a non-profit organization promoting internet openness, innovation, and participation, defines a website as "a collection of web pages which are grouped together and usually connected together in various ways."[1] This broad definition allows for different types of connections between individual web pages, which are themselves defined as "a document which can be displayed in a web browser."[2]

Margaret Rouse, senior editor of Techopedia among others narrows down Mozilla's definition, describing a website as "a collection of publicly accessible and interlinked web pages that is identified by a common domain name"[3]. This approach emphasises the role of the domain as a qualifying criterion, acting as a border that separates one website from another — much like a political border separates nations. Sara Culmann, in the panel discussion "The

[1] "What is the difference between web page, website, web server, and search engine? - Learn web development | MDN", https://developer.mozilla.org/en-US/docs/Learn/Common_questions/Web_mechanics/Pages_sites_servers_and_search_engines

[2] Ibid.

[3] "What is a Website? Definition, Types & Components", https://www.techopedia.com/definition/5411/website

Power(lessness) of Websites"[1] at the STRP Festival 2024, points out, that domain is controlled area by the owner within and can be disputed territory, much like on a real political map.

The utilitarian definitions also highlight the importance of navigation, the interconnectedness of web pages, and their necessary public accessibility. Interestingly, this definition reflects more the way *most* websites are designed and built nowadays than what a website *is*. It describes the way websites are organised and built, guided by "best practices", UX "laws" and what we already seen on the web.

One may argue that a website does not stop being a website if the navigation is obscured, or hyperlinks are removed or even when the public access is restricted. 'Unindexed'[2] was a website created by Matthew Rothenberg that continuously searched Google for itself over and over again. The moment it found itself in the search results it would irrevocably and securely delete itself, making the precise instant of algorithmic discovery the catalyst of destruction. Visitors were encouraged to post contributions to the site (which would also be destroyed when the site was detected) and to share the site with others, bearing in mind the impact their method of

[1] "Panel Discussion: The Power(lessness) of Websites STRP Festival, 13 April 2024", https://the-powerlessness-of-websites.vercel.app/

[2] 'Unindexed' on GitHub, https://github.com/mroth/unindexed

From a purely technical point of view, a website is nothing more than a combination of files, hosted on a web server of some sort, that may or may not be publicly accessible through the nternet.
From a more conceptual point of view, websites are conversation starters. They are a way for people — and companies — to engage in a conversation with other entities in the digital space. The reasons behind these conversations can vary: some are motivated by financial interests, while others by pure curiosity.

“In this context, websites are tools, no different from books or magazines or any other type of communication tool.”

Manuel Moreale

manuelmoreale.com

sharing would have on hastening the eventual discovery of the URL by Google crawlers. The site was finally discovered by Google after 22 days on Tue Feb 24 2015 21:01:14 GMT+0000 (UTC) and consequently instantaneously self-destructed milliseconds later. Prior to the automatic deletion it received hundreds of visitors and dozens of contributions. No backups were kept.

The utilitarian perspective on websites, while prevalent and practical, reveals both strengths and limitations in our understanding of what constitutes a website. This view emphasises functionality, organisation, and accessibility, which undoubtedly form the backbone of most contemporary web experiences. While the utilitarian perspective offers a solid foundation for understanding websites, it may not fully capture the diverse range of possibilities that websites can embody. This definition, though practical, risks limiting our conception of what a website can become.

Website is an experience

Thinking about short life and nearly instant death of 'Unindexed' project, I cannot help but ask: Are websites actually becoming dead if there is no one to look at them? Is website an experience?

Kim Lê Boutin, designer, lecturer, and curator of loadmo.re[1], in our conversation pointed out a rather fascinating aspect of websites. "Website is a performance", she says, "websites *exist* when there is someone to visit them". This perspective highlights several important aspects. First of all the temporal and ephemeral nature of the website - just like the piece of performative art exists only in the moment it's being performed, the website "performs" when it's being visited and interacted with. Each visit is unique — like how no two live performances are exactly alike, each interaction between a person and a website influenced by factors such as software used, device, quality of Internet connection, location, time of the day and personal context.

[1] "LOADMORE", http://loadmo.re/ or @loadmo.re on Instagram

Viewing websites as participative performances underscores the crucial role of interaction. Unlike traditional media where the audience remains passive, website visitors actively shape

their experience through clicks, scrolls, and input. They're not just spectators, but co-creators of the performance. This framing of websites as performances encourages website creators to think beyond mere functionality, considering the holistic, experiential, and social aspects of web interactions. It emphasises creating engaging, memorable experiences that unfold in real-time and invite active participation.

Website is a place

Next approach worth mentioning attempts to describe websites in spatial terms. These metaphors to the web are very old and can be found everywhere, even in the very name: web *site* - building *site,* Internet is then city and so on. But how can something non-physical be described in those terms?

The term “digital space” refers to a broad and evolving concept that encompasses various aspects of virtual interaction, communication, and technology. Nancy Wu in her thesis project ‘A Website Is A Room’[1] describes how we are connected with the space of a website. She writes:

> We are embodied within each web space that we enter. Our habits, mannerisms, preferences, and quirks manifest themselves in the way we behave within websites. Our dwelling within these web spaces is no less physical than our presence in actual rooms. We invest time and physical, mental, and emotional energy. Like actual rooms within our home, these web spaces have a real, physical impact on our wellbeing. Each space creates a certain feeling within its boundaries, through its structure, contents, and aesthetics.

[1] “A Website Is a Room”, last accessed 17.09.2024, https://a-website-is-a-room.net/

“Websites have architectures which are interesting, but for me, the way I eventually framed it for myself is: a website is a performance. Because that's my approach, websites exist when there is someone to visit them. I'm also trying to create websites that exist when several people visit them, so collective performances.

So yeah, I think of them as performances.”

With specific gestures on loadmo.re what I'm trying to do as well is to record people interacting with the websites to show human presence, human beings and not just the screen recording where it removes the human, you know. So the performance only happens when someone is there to see them. And so a website requires visitors, as a performance requires spectators and it's not just a performance, it's a participative performance because you also want to invite people to interact. So that's my definition and I didn't even think before this call of what I would answer, but it's interesting.

Kim Lê Boutin

loadmo.re

Malte Müller[1], a web designer, writer, and programmer, while stating that a website is "a markup document containing text and other media, served via a networked connection," approaches web design as architecture, suggesting that a website is a place and not an object. His approach started as a tweet, and is now formulated in a manifesto[2]. It states that websites are inhabited, inherently public places, which are local, despite their distributed nature.

Websites adhere to culturally established patterns, languages, and user expectations in similar ways to architecture. It also views websites as constructed cultural artifacts, which exist within frameworks and foster social discourses by establishing new ways of interaction or raising aesthetic questions. As material and construction are defining characteristics of architectural work, websites are characterised by their technological stack, used in the process of their creation. This manifesto aims to provide a starting point for expansive and critical discourse on website design.

When considering websites as places or spaces, both terms often used interchangeably. However, there seems to be a certain distinction between 'space' and 'place'.

[1] Malte Müller Homepage, https://maltemueller.com/

[2] "WEB DESIGN AS ARCHITECTURE", http://www--arc.com/

Designer Dan Ramsden[1], writes about structures in information architecture and draws a distinction between the two terms. He states that places have a defined structure, whereas spaces lack this inherent structure and are more open. The structure of places in the context of information architecture is often directory-like and hierarchical, while spaces, on the contrary, are less organised and more fluid. He describes spaces as a consequence of the evolution of web technologies and a more dynamic web. Traditional web design has been largely two-dimensional. However, with the widespread adoption of WebGL, we're seeing a shift towards more immersive, 3D web experiences. Such experiences often leverage spatial metaphors, such as rooms, buildings, or landscapes. This can make websites feel more like places to be explored rather than pages to be read.

The concept of "website as a place" represents a significant shift in how we perceive and interact with digital environments. This metaphor encourages us to think beyond the traditional view of websites as mere collections of pages and instead consider them as dynamic, inhabitable spaces that we actively experience and engage with.

[1] "Spaces vs. Places – structures in information architecture", https://danramsden.com/2014/01/15/structures-in-information-architecture/

Website is anything

Laurel Schwulst[1], an artist, designer, writer, educator, and technologist has a rather different perspective on what a website is:

[1] Laurel Schwulst Homepage https://laurelschwulst.com/

> "What is a website, anyway?" It's easy to forget. Today there are millions of ways to make a website, and the abundance is daunting. But at its core, a website is still the same as ever before: A website is a file or bundle of files living on a server somewhere.

Rather than focusing on the technical aspects or common features of websites, with her lean definition of a website, Laurel emphasises the limitless potential of the web as a medium: if a website is just "files on a server", it could be literally anything. It suggests an empowering message — the web is what we make it. In the article "My website is a shifting house next to a river of knowledge. What could yours be?"[1] she points out that what a website is depends entirely on the website creator:

[1] Laurel Schwulst. "My website is a shifting house next to the river of knowledge. What could yours be?", https://thecreativeindependent.com/essays/laurel-schwulst-my-website-is-a-shifting-house-next-to-a-river-of-knowledge-what-could-yours-be/

> My favorite aspect of websites is their duality: they're both subject and object at once. In other words, a website creator becomes both author and architect simultaneously. There are endless possibilities as to what a website could be. What kind of room is a website? Or is a website more like a house? A

> boat? A cloud? A garden? A puddle? Whatever it is, there's potential for a self-reflexive feedback loop: when you put energy into a website, in turn the website helps form your own identity.

Reframing websites through various metaphors – from shelves to plants, boats to rocks invites us to explore the vast creative potential of this digital medium. It is really fascinating how one's perspective shifts when a proper metaphor is applied. For instance, seeing a website as a shelf raises questions of what one might want to put on a shelf or take from it. What interesting juxtapositions do we create on our little curated shelf we call a website? What kind of shelf is your website? Thinking about a website as a plant moderates our expectations of it, emphasising the many factors influencing a single website as a part of the global network. "Plants can't be rushed," writes Laurel, "they grow on their own." Similarly, websites thrive if taken proper care of and can surprise one with unexpected "flowers" or even "fruits".

All the different metaphors Laurel brings up help us to reflect on the very nature of websites and our own practice of making them. They take off the pressure to create *the perfect website* that the professionalisation and commercialisa-

tion of the web exerts over web makers. “If a website has endless possibilities, and our identities, ideas, and dreams are created and expanded by them, then it’s instrumental that websites progress along with us”, concludes Laurel and I cannot help but agree.

This shift in perspective not only liberates us from the constraints of conventional web design but also encourages a more personal, organic, and evolving approach to our online presence. As we embrace this fluid and metaphorical understanding of websites, we open ourselves to new possibilities for self-expression and connection on the web.

Website is a tree

The tree, a powerful symbol of knowledge—often referred to as “the tree of life” — appears in many cultures and religious beliefs throughout history. This arboreal metaphor later extended to interpreting the evolving complexities of human understanding, from theological beliefs to the intersection of scientific subjects. Today, this scheme finds relevance in genetics, linguistics, philosophy, computer and information science, among many other areas. Manuel

Lima[1], in his book about information design, refers to the tree as one of the earliest representations of systems of thought. Trees have been invaluable in organising, rationalising, and illustrating various information patterns through the ages. The branched scheme of a tree primarily highlights hierarchical ordering, where all divisions stem from the foundational trunk, pragmatically expressing multiplicity from unity.

The arboreal organisational scheme is deeply ingrained in many different areas, from file systems and database indexes to XML/HTML markup. Numerous aspects of common website design and functionality can be directly mapped to properties and operations of tree data structures.

In tree data structures, the root node is the topmost node with no parents. Similarly, the homepage is the top-level page from which all other pages descend. Each node in a tree can have child nodes, just as each page on a website can have subpages hierarchically beneath it. Pages without subpages correlate to leaf nodes in a tree. The path from root to any node in a tree is unique, corresponding to the unique URL path for each page on a website. Tree depth, breadth,

[1] Manuel Lima, Visual complexity. Mapping Patterns of information. Princenton Architectural Press New York, 2011

“Then, for me, I view the internet much like a virtual landscape, where each website is a building. Naturally, they have an address, that tells you how to go somewhere. A domain might be like a building, where the individual URLs represent different rooms or places inside it.

When you enter a website, you might get a cosy feeling, you might instead feel it is a very corporate-feeling place or perhaps you just feel uncomfortable with a messy layout.”

I feel, as a web developer, much like an architect in the sense that I try to build the website in a way that allows the visitors to easily find their way, and I feel a bit like an interior designer while trying to create a certain vibe on the sites I build.

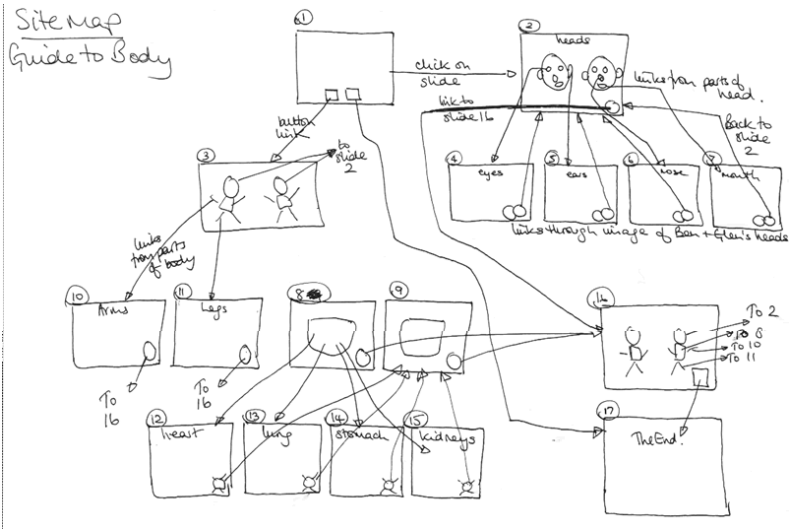
Since on the internet, anyone might stumble upon your address, I think it's important that we try to be accessible and inclusive with our sites, which includes things like making sure it loads quickly (for those with slower connections), it is well-designed (for those less tech-savvy) and also accessible in the traditional sense (for e.g. those with visual impairments).

@vrugtehagel
vrugtehagel.nl

and traversal algorithms can inform the design of sitemaps and navigation menus. Operations like adding, deleting, or moving nodes in a tree are analogous to adding, removing, or reorganising pages on a website.

Websites are often represented in a sitemap, which provides an overview of the entire website structure. While sitemaps are not required by any standards, most of the websites have at least one. There are several kinds of sitemaps, created for different purposes and containing different informations about the website. For instance, there are ones web designers create for themselves, to have an overview in the early stages of creating a website. These sitemaps are often loosely structured (some even drawn by hand on a piece of scrap paper), contain a lot of spontaneous notes and are subject to change over the course of developing a website.

Another kind of sitemap, sometimes called HTML sitemap is basically another page of the website, which references all other pages, revealing the structure of a website. This page serves as a roadmap for the visitors, which can come handy in case of a really complex websites or if the navigational concept is radically different from the website structure. In the early days



Hand-drawn Sitemapbrought as an example by Laura Franz at Typographic Web Design, http://www.typographicwebdesign.com/creating-a-system/exercise-analyze-a-site/

of the web, when the search engines didn't assume the Internet's "gatekeeper" role yet, it was a common practice to curate HTML sitemaps to help with navigation.

With the advent of search engines, the third kind of sitemaps emerged — the XML sitemaps. These have a strict structure, are loaded with metadata about the website, and are not meant for humans. In 2005, Google introduced the Sitemaps protocol as part of a plan to improve the coverage and freshness of their index. Sitemaps is meant for web creators who want to

provide detailed information about pages of the website available for the web crawlers to search engines. They contain structured data in XML format describing web pages — URL addresses, last update times, change frequencies, and importance relative to other site pages. Web creators quickly adopted this practice, since it promised faster indexing. Today, maintaining an up-to-date XML sitemap has become an obligatory part of website creation, especially in commercial contexts.

The metaphor of a website as a tree aptly captures the hierarchical structure and organisation of websites. The tree structure not only informs the technical aspects of website development, such as URL paths, HTML markup structure and navigation systems, but also manifests in practical tools like sitemaps. As the web continues to evolve, the tree remains a fundamental model for organising and understanding the complex information landscapes.

Website is never ready

What happens to a website over time? How does it change? Who is responsible for keeping website "alive"? Websites are dynamic entities that evolve over time, challenging the notion that they are ever truly "finished." This temporal aspect of web design and maintenance is often overlooked but plays a crucial role not just in overall lifecycle of a website but in the very understanding of what a website is.

Artist Tom Bubul's[1] approach emphasises the idea of natural aging in digital spaces. He advocates for allowing pages to retain their original markup and structure. Talking about "design tenets" of his personal website on "HTML Energy" podcast[2], he states that old pages retaining old markup should be considered kin to old subway stations retaining old signage with old typefaces, contributing to the character and history of the place. Pages shall be permitted to deteriorate, Tom insists.

The "Website as architecture" manifesto further reinforces this concept, suggesting that websites, like buildings, undergo processes of renovation, repurposing, or even decay over time. This perspective encourages us to view websites

[1] Tom Bubul Homepage's design tenets, https://tombubul.info/tenets.html

[2] Podcast HTML energy,, https://html.energy/podcast.html

as living structures rather than static creations. Laurel Schwulst's view aligns with these ideas, emphasising that websites are living, temporal spaces. She suggests that the inherent imperfection and unfinished nature of interactive digital environments contribute to their beauty and appeal.

[1] Pablo Somonte Ruano Homepage, https://pablo.sx/

Pablo Somonte Ruano[1], interaction designer and media artist brings attention to the often-overlooked aspect of website maintenance. He argues that the bulk of work in keeping a website active occurs after the initial design and coding phases. These ongoing tasks, such as updates, fixes, and management of hosting and domains, can accumulate and ultimately surpass the effort invested in the site's creation.

The temporal nature of websites challenges traditional notions of completion and perfection in design. Embracing the ongoing evolution of websites can lead to more authentic and engaging online experiences. This perspective encourages website creators to view their work as a continuous process rather than a finite project. By acknowledging the living, changing nature of websites, we can create more resilient and adaptable digital spaces that grow and evolve alongside their users and creators. This ap-

proach not only honours the history and character of a website but also ensures its relevance and functionality in an ever-changing digital landscape.

“I'm glancing over the obvious first: A website is a place, online, which people access through computers and computer networks. What each website does, and how it acts is the result of a negotiation between the people making and maintaining the website as well as of those who visit and use it. This negotiation is not always explicit, but the result of a complex dance of interests, hidden and apparent, between all of those involved.”

As a person who codes and maintains websites for various people and organizations I would like to briefly talk about websites from the perspective of reproductive labor. Most of the work of keeping a website active and useful happens after the design and code of a website is finished and consists of endless amounts of minor tasks such as content updates, fixing broken links, renewing domains, managing hosting and generally making sure to keep the lights on. These kind of tasks are generally overseen or not talked about as much in the context of websites. With this in mind I would finally say that a website, like a place, is a space which also requires care, attention and love for it to be special.

Pablo Somonte Ruano

pablo.sx

Website is a webpage

A website, in its simplest form, can indeed be a single webpage. Early websites were often single webpages due to technical limitations and simpler online needs. Even a single HTML file can constitute a fully functional website, containing all necessary information or functionality. As web technologies have evolved, we've seen the emergence of single-page applications in modern web development. While these are inherently more complex, they still embody the "single page" philosophy, dynamically updating content without requiring full page reloads.

Dan Rubin in his article 'Off the page' for 'Back office #3 Writing the Screen'[1] states, that mapping the concept of a page to the web has affected the way we design, create, curate content on the web. He sees web design in its current state as a strangely beautiful hybrid inheriting its principles, typography and language from decades of print graphic and information design, enhanced through layers of interaction, audio and video. However, Rubin suggests that the web is capable of much more, hinting at unexplored possibilities beyond the constraints of page-based thinking.

[1] Dan Rubin, "Off the page", Backoffice #3 Writing the screen, last accessed 17.09.2024, http://www.revue-backoffice.com/en/issues/03-writing-the-screen/02_rubin

Websites can be understood from multiple perspectives: (1) *technical* — a website is a collection of structured documents, typically in HTML format, transmitted via protocols like HTTP and hosted on servers; (2) *metaphorical* — websites can be conceptualised as various entities — homes, spaces, platforms, services, or experiences. This flexibility in interpretation reflects the diverse roles websites play in our lives; (3) *auxiliary representations* — these include sitemaps for navigation and search engine optimisation, browser's document models for rendering and interaction and accessibility trees for assistive technologies. These representations highlight the multifaceted nature of websites and their various interpretations by different systems.

The concept of a website has evolved significantly since the early days of the Internet. As technology advances and our understanding of digital spaces deepens, we may yet see websites transform into even more immersive, adaptive, and interconnected entities. They might become more seamlessly integrated with our daily lives, blur the lines between physical and digital realities, or take on forms we haven't yet imagined.

By continually questioning and expanding our conception of websites, we open ourselves to new possibilities in digital communication, interaction, and information sharing. The future of Internet lies not just in technological advancements, but in our ability to creatively reimagine their purpose and potential in our increasingly connected world.

As we continue to explore the possibilities of the web, it's crucial to look beyond conventional definitions and constraints. Instead of answering the question *"What is a website?"* one and for all, here is an invitation to ponder over this:

What else could a website be?

Looking through the window: on relationship between browser and website

Browsers today are massive, complicated applications that need to be many things: stable, secure, clean, simple, efficient, fast, extensible, and preferably open-source. However, all these features push browsers toward becoming increasingly inconspicuous. Some view this as a positive trait — finally being able to focus on website content, ignoring the "frame." Yet, this seemingly unobtrusive software still persists and influences how we perceive and interact with the web, whether we notice it or not.

A browser is much more than a useful frame around a website. Its unique relationship with websites contributes to shaping the web as we know it today. Louise Drulhe[1], in her visual research on the spatiality of the Internet, describes the web as a framed space.

[1] Louise Drulhe. "Critical Atlas of Internet", https://louisedrulhe.fr/internet-atlas/

She writes[1]:

[1] Ibid.

> Looking at Internet is like looking at a landscape through a window; we see a framed part of the whole view. In the case of the landscape, we can leave the room and see the whole stretch. However, this is not true for the Internet which only exists within a frame.

This "frame" — the browsing context — delimits its Internet space and influences its arrangement, making Internet architectures effectively dependent on this "frame." However, I would argue that "frame" is not an optimal metaphor for a browser, as its relationship with the website is complex, and its role in creating the visual representation we actively interact with is enormous.

A web browser is an unusual piece of software, carrying significant responsibility for how a website is presented to the person requesting it. The web *inverts control*, with an intermediary — the browser — crafting the visual rendering of a webpage while the webpage creator specifies rendering parameters and content to this intermediary. There are practical reasons for this design: creators of web pages trade *direct control* over every single pixel being drawn on the

screen for flexibility and forward-compatibility. However, this design has certain consequences.

Different browsers may interpret and render the same HTML document differently, leading to inconsistencies in how websites look and function across various browsers and devices. This ultimately influences our perception of websites and undermines the idea that a website is the same for everyone. It also influences the role of the website creator — since the browsing experience can be changed through browser settings and extensions, potentially overriding website designs, the role of the web creator is not one of ultimate control. It requires web creators to comply with certain decisions made by browser developers or organisations enforcing standards, to continually test and update their creations to ensure compatibility with multiple browser versions and features, creating an ongoing maintenance burden. This intricate interplay between various stakeholders in the web ecosystem is continuously negotiating and shaping the web and our idea of it.

Browsing context and rendering process

The role of the browser may appear rather straightforward conceptually — get a requested-website, render it perceivable[1] and allow to interact with it. But there are so many layers of complexity affecting the process, that developing web browser is by no means an easy task. Browsers nowadays consist of many complex layers and processes, the core ones being user interface, storage, networking, browser engine, JavaScript engine and rendering engine. Some might even claim, that browsers became as complex as certain operating systems. The inner workings of the browser, the process required to complete a rather trivial task to display a simple web page shows that there is enormous influence of browser architecture to what we call a website.

One of the fundamental concepts, helping us understand how browsers are dealing with websites is browsing context. Earlier browser shared the browsing context across websites, meaning that if something goes wrong in the process of rendering one website, the whole browser is affected. Today, every browser is multi threaded, encapsulating browsing context for every single

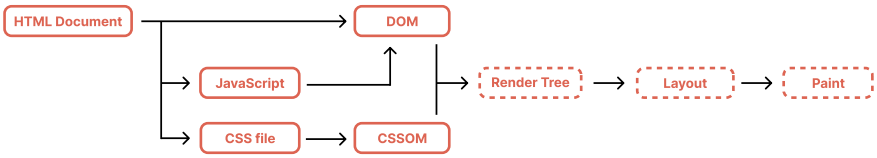
[1] Noteworthy here is the fact, that even though most of the web browser throughout history produce some kind of visual output, it is not implied, that it has to be. Rendering process is handled different in every browser, leaving the browser developers choice of how to implement the interpretors, the constraints and contents of a web page.

website, severely constraining the communication between browsing contexts. Certain HTML elements like <*iframe*> can instantiate further nested browsing contexts inside them. It can be imagined as a collection of closed environments, managed by the browser.

Inside every browsing context, a rendering process called *critical rendering path* is executed every time we request a website from the server. Having received HTML document browser's rendering engine will start parsing it — raw bytes to characters, characters to strings, strings to tokens and tokens become *objects* with properties and functions, and incrementally get linked together into a whole new data structure — Document Object Model (DOM). This structure is paramount to what we perceive as a website later.

As the browser reads through HTML, it will encounter stylesheets, then stop everything, request this specific file from server and repeat similar parsing process, this time creating an object model for style rules — CSSOM.

Next, both trees are combined to create *render tree* (or several trees to be precise), which is used by the process called 'Layout'(or 'Reflow') to



Critical rendering path flow

compute the exact size and position of every *visible* element, lastly the computed elements are displayed on the screen by the efforts of 'Paint' process.

But where does the JavaScript fits in this picture? The one very important part of the website architecture that makes the whole experience interactive? In some ways, browser treats scripts similar to CSS files. During the initial parsing of HTML document, if a script element is encountered, it will stop everything, fetch and evaluate the script before continuing on parsing. Since JavaScript can query and modify both DOM and CSSOM it is considered more ergonomic, to first execute the changes the script *may* contain than construct the whole DOM only for it to be overthrown afterwards. However, the execution of the particular script can be deferred till the end of DOM/CSSOM construction or asynchronously fetched to be executed when ready.

Any further processing of a web page (e.g., programmatically in script or through user input – scrolling, clicking, and so on) relies on DOM, not on the initial document received from server, any changes here will trigger 'Layout' and following processes to reevaluate. This rendering process (spare initial parsing) basically happens in a loop. Ideally 60 times in a second, the web page is being re-drawn based on document model created from the HTML document. Every single frame application code is run, styles are computed, layout is recalculated, a process called "garbage collection" happens and finally 'Paint' process is run again.

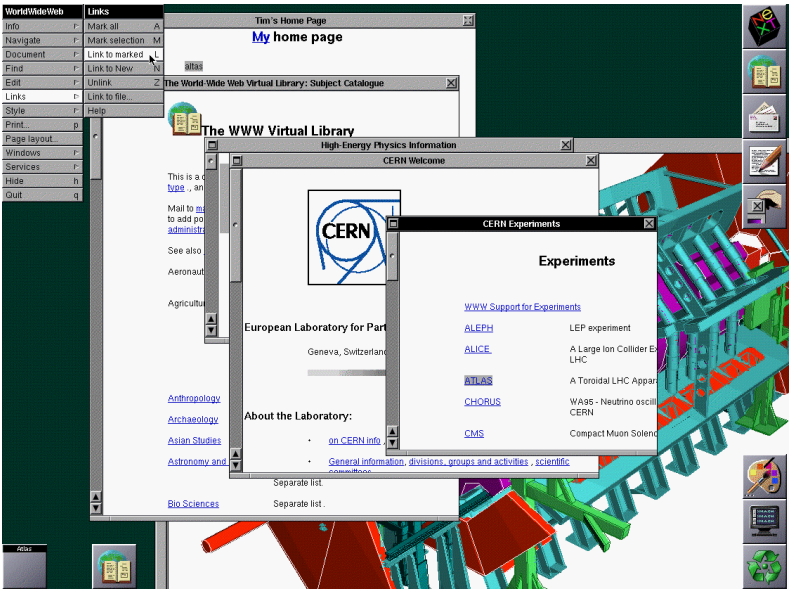
This way, what we refer to as a website is, in fact, a result of evaluation of initial document by the browser. The website we interact with by scrolling, clicking, typing, dragging is not really the document written by the website creator, but a complex multilayered object, created and mediated by browser

Cash runs everything around me: browser wars and the power of default

Since the browser is de facto the mediator if the web's interaction and the implementer of the web, its importance was recognised early on and spawned a fierce competition, often referred to as the 'browser wars.'

In 1990, Tim Berners-Lee together with Robert Cayo, developed the first browser, WorldWideWeb (later renamed Nexus to avoid confusion between the client software and the abstract space itself), which was released the same year. Reportedly, it took them around two months to develop it. The World WideWeb is more a proof-of-concept — it was a text-based browser, no graphics were displayed, the set of functions was limited to browsing, updating and hyperlinking. In 1991, third-party developers were invited to join the project.

By April 1993, CERN authorities were finally convinced to make the source code freely available and declared WorldWideWeb a free platform. In 1994 Tim Berners-Lee moved to Massachusetts Institute of Technology (MIT) and founded the World Wide Web Consortium



Screenshot of the WorldWideWeb browser, taken in 1993. The differences between this and the first edition would be that in 1990 NeXT was greyscale and the inline images such as the world/book icon and the CERN logo, would have been displayed in separate windows.

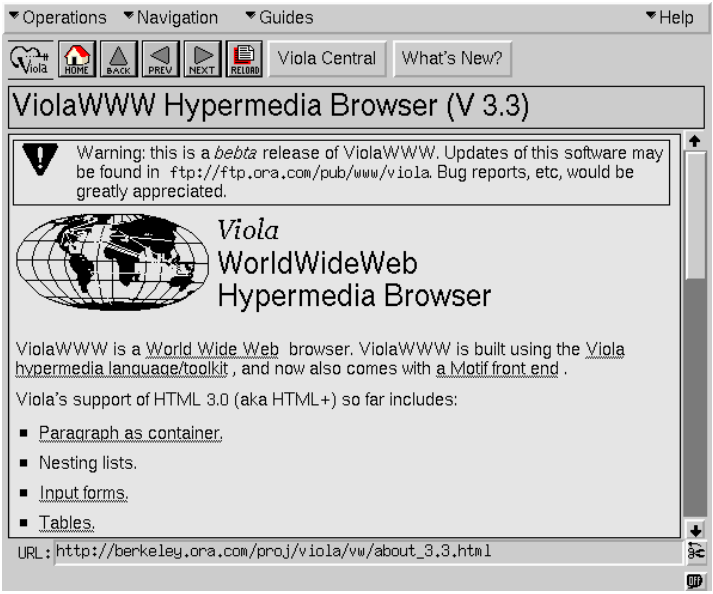
(W3C), where he remains a director. W3C is an international community dedicated to implementing standards on the web.

After the publication of the source code, many developers were inspired to develop and release their own software for browsing the web. Between 1992 and 1993, several browsers appeared: ViolaWWW, written by U.C. Berkeley student Pei-Yuan Wei, MidasWWW by Tony

Johnson of SLAC, tkWWW by Joseph Wang at MIT, just to name a few. One of the first browsers, the text-based Lynx[1], is still in use today. However, most of them were functionally still very close to the original browser and required a great deal of experience in computer programming just to get them up and running.

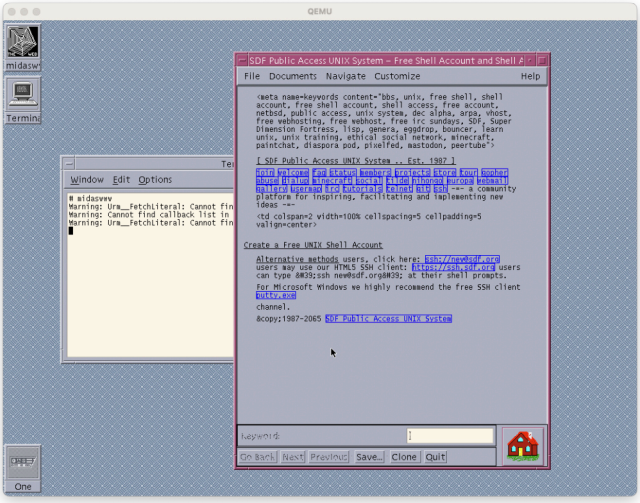
[1] Lynx Homepage, https://lynx.invisible-island.net/lynx2.9.2/index.html

In 1993, the Internet experienced rapid growth and academic institutions, government agencies, and businesses started to recognise the potential of the open web. There was a widespread demand for better software applications to navi-

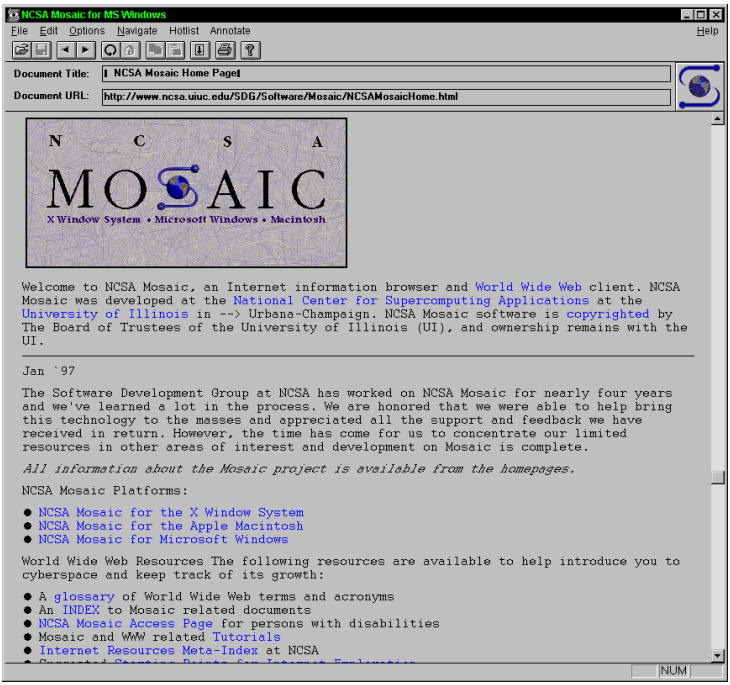


ViolaWWW hypermedia browser (v 3.3.)

gate this expanding digital landscape. Marc Andreessen, a programming student at the University of Illinois at the National Center for Supercomputing Applications (NCSA) and his colleague Eric Bina were able to provide. First of all, their browser Mosaic was easy to use and install. They were also the first ones to get inline images working - beforehand all images needed to be opened in a separate window. This feature alone made the web pages way more appealing. Another thing Mosaic's developers got right is establishing 24h customer support, addressing issues that had to do with applications' use, stability and installation.



MidasWWW 1.0 running on Solaris 2.6/CDE



Mosaic 1.0 running on MS Windows. On the Mosaic Homepage is the text, announcing end of browser development in January 1997

Tim Berners-Lee[1] writes:

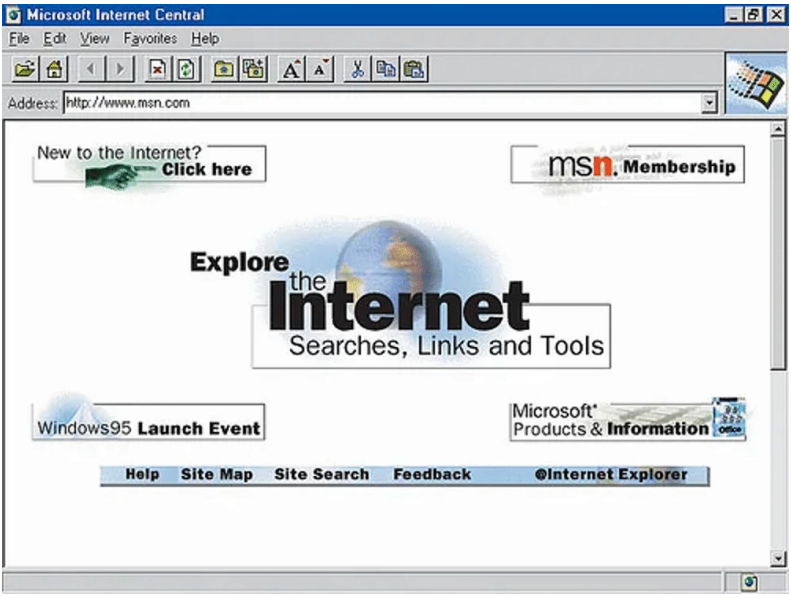
[1] "Frequently asked questions", https://www.w3.org/People/Berners-Lee/FAQ.html#browser

> Viola was more advanced in many ways, with its downloaded applets and animations way back then — very like HotJava was later. But Mosaic was the easiest step onto the Web for a beginner, and so was a critical element of the Web explosion.

In 1994, along with investor James Clark, Marc Andreessen founded the software company Mosaic Communications, but NCSA fought for the name and won. To avoid licensing problems, the company was renamed Netscape Corporation, and the browser was renamed Netscape Navigator. However, something the founders of Netscape did not take into account — they did not buy the source code of Mosaic from the NCSA.

Netscape Navigator 1.2 Browser for Windows 3.1 in 1995 displaying www.netscape.com. Interestingly, the field with website URL is called "Netsite".

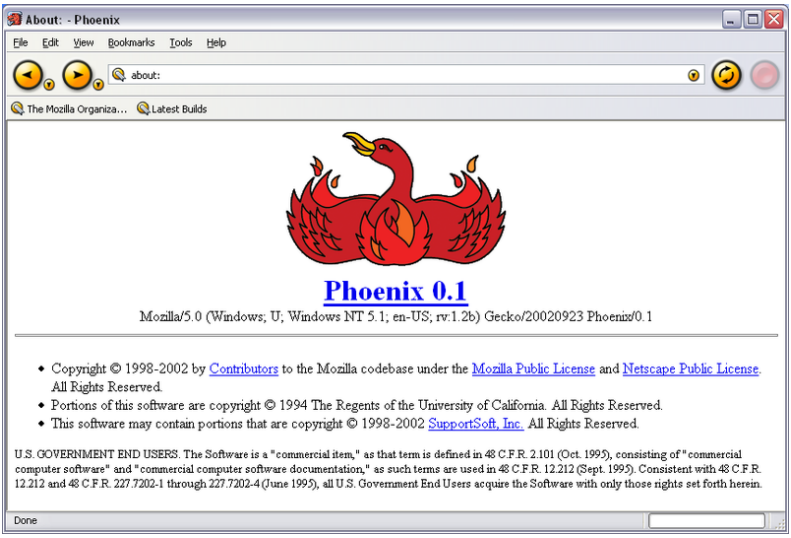
The first “browser war” started in 1995, when software giant Microsoft licensed old Mosaic code and built its own browser - the infamous Internet Explorer. Over the next years both companies worked restlessly to outbid its opponent, releasing better, faster versions as rapidly as possible. The biggest spoils of this “war” for todays web are JavaScript, which gave websites powerful computing capabilities, created and released by Netscape and Microsoft’s Cascading Style Sheets (CSS) — both technologies later became absolute standard for web page creation.



Internet Explorer 1.0 showing www.msn.com Homepage in 1995

The end of this race was brought upon by Microsoft, who in 1997 started shipping Internet Explorer 4 completely integrated in Windows 98 operating system and by the end of 1999 Microsoft had the majority of the market. Netscape could not keep up, decided to open source its codebase and created the non-profit Mozilla Organization to coordinate future development of this product. In 1999 Netscape was acquired by AOL which discontinued browser development by early 2008. The decision to integrate browser in operating system, although won the "browser war" for Microsoft, but also caught attention from U.S. authorities — in their opinion the state of affairs violated antitrust laws and brought charges against the company. This conflict was finally resolved in 2001, when Microsoft agreed to also provide Internet Explorer as third-party software.

The second "browser war" broke out in 2004, as smaller browsers were set to challenge Microsoft's supremacy. The hegemony of Internet Explorer spawned a large circles of haters, especially among web developers, who were eager to support anyone, who might oppose the giant. Risen from the ashes of Netscape, Mozilla Foundation launches its browser Firefox (to support the metaphor, first name of the browser was

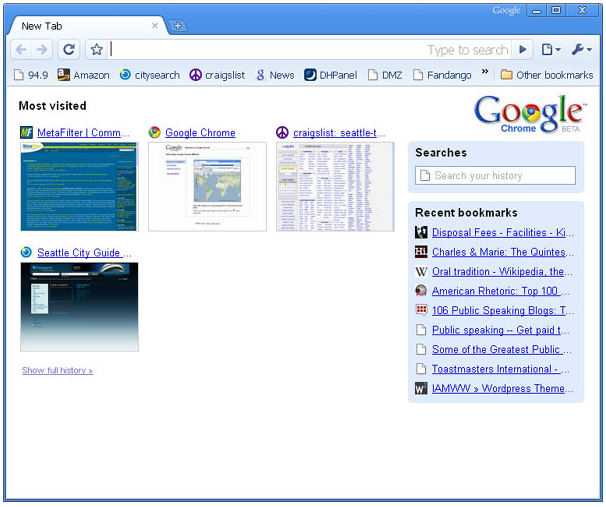


Startpage of Phoenix 0.1 browser, released on 23. September 2002. Later will become Mozilla Firefox.

Phoenix, then Firebird, later changed due to copyright issues) in 2004. Another opponent was Norwegian Opera Software with their browser Opera was also gaining popularity, being first browser to support tabbed browsing (called hotlist) and mouse gestures. Apple was also developing their browser named Safari with their own lightweight layout engine WebKit, which was released in 2003. From a technical point of view, Internet Explorer changed from an innovator to a catch-up. When it introduced new features, there were already implemented by competitors and offered better com-

pliance to emerging web standards than "quirky" Internet Explorer.

Firefox was prophesied to have a great future, which might have come if not for Google Chrome, launched in 2008. The Google's own browser was the fastest browser on the market, offered a brand new multi-threaded architecture, support for plugins and minimal interface, which will become standard for browsers in the future. All that combined with ultra short development cycles, which implemented more and more features, Chrome overtook leading posi-



Startpage of Google Chrome in Beta version as posted in September 2008 by Scott Berkun

tion for the first time by the end of 2011. On May 25, 2017, Andreas Gal, former Mozilla CTO, publicly announced that Google Chrome won the "second browser war". Since then Chrome continued gaining popularity and as of February 2024 its market share is 65,74%, followed by Safari (18,1%) and Edge (4,98%).

Why fighting a "war" over browser at all? Why is it so important to convince people to use your browser? Having the own browser lets software companies exert an *enormous* amount of control over the way people access the Internet. But is not most of the browsers nowadays free? How the browsers make money to support their ambition? Browsers are not necessarily supposed to make money directly by offering paid versions or subscriptions model to the people who want to browse the Internet, even though that was the case earlier. Today, they serve the goals of their creators in more *strategic* ways.

For example, Microsoft and Apple have their own browsers to make their operational system fully featured, which is, in fact, their main product. Shipping a feature-rich operating system is necessary for several reasons. A seamless and cohesive user experience right out of the box is

highly desirable, offering such experience allows the company behind it to gain competitive advantage. The primary goal of large software companies is often to build an *ecosystem* of applications. This strategy allows them to better control and shape their digital environments, using user data generated and shared within the ecosystem. Ultimately, this approach encourages users to remain within their product family.

Such extensive ecosystem control often makes a customer dependent on them for using a certain application, because switching to alternative becomes a complicated endeavour. This situation is known as *vendor lock-in* or proprietary lock-in. As one would expect, this state of affairs benefits the company only — whoever finds himself in a vendor lock-in can regain the flexibility and freedom of choice only by paying a high price, often needing to overthrow the whole system.

Google is directly interested in improving the overall experience of "being on the Internet" and keep people searching and browsing, since their main source of income remains online advertising. As of Q2 2024 Alphabet's revenue[1] sourced from advertising (Google Search engine

[1] Alphabet Announces Second Quarter 2024 Results, https://abc.xyz/assets/19/e4/3dc1d4d6439c81206370167db1bd/2024q2-alphabet-earnings-release.pdf.

Ads, YouTube Ads, Google Network ads combined) equals $64,61 Billion, being the 76,2% of the total revenue for mentioned above timeframe.
Google's Chrome browser is one project among others, like SPDY (HTTP/2) and WebM, aimed at making the Internet faster or cheaper. This effectively increases the amount of Internet users, or in other terms, the audience size for Google's advertising.

Brave, a privacy and security-focused browser known for its ad and tracking blocking features, relies on monetisation through a blockchain-based digital advertising service. Brave Rewards program can be enabled in the Brave browser in order to collect BAT (Basic Attention Tokens) for watching advertisements. BAT can be exchanged for other currencies, sent to websites or content creators in order to support them or used to buy gift cards. While Brave's approach of blocking third-party ads and trackers may seem beneficial for privacy, it raises some ethical concerns. By replacing existing ads with their own advertising system, Brave essentially hijacks the revenue stream that many websites rely on to sustain their operations. This practice could be seen as hypocritical — removing one form of advertising only to introduce another.

Mozilla Foundation, the non-profit organisation, that by their own claim aims to keep Internet open and accessible, and the developer of the Firefox Browser, is a behemoth. As of 2022[1], Mozilla (including the Foundation and the wholly owned for-profit Corporations), had total assets worth over $1.3 Billion. The majority of 2022 revenue, estimated in the report as $593,516 million comes from royalties (86%), subscription and advertising revenue (12,7%), not donations (1,6%).

[1] Mozilla Foundation and subsidiaries. December 31, 2022 and 2021. Independent Auditor's report and consolidated financial statements. https://assets.mozilla.net/annualreport/2022/mozilla-fdn-2022-fs-final-0908.pdf

Subscription revenue is described in the 2022 Audited Financial Statement as consisting of two parts: subscription to a web bookmarking service Pocket Premium and VPN service. Advertising revenues are generated through selling advertising space in Mozillas licensed content and products. For example, the 'New Tab/Tiles' advertising service places links to sponsored content when a new tab is opened in the Firefox web browser. Further the report clarifies, that 'royalties' refer to the certain percentage of revenues earned by its customers through their search engines incorporated in the Firefox web browser.

Search integration agreements between browser companies and web search engines are not ex-

clusive to non-profits; they're a common practice across the industry. While the operating system remains the main product for Microsoft and Apple, integrated browsers can create additional revenue through default search engine deals and other partnerships. For example Bloomberg reports Apple received a hefty sum of $20 billion from Alphabet to make Google a default search engine in Safari browser.

This raises the important issue of "defaults." Consider your own browser usage: Which one do you use? Did you actively choose it, or was it pre-installed on your device? Have you ever thought about switching to a different browser?

Jon M. Jachimowicz, Shannon Duncan, Elke U. Weber, and Eric J. Johnson[1] assert that when people face decisions with a pre-selected choice option — a "default" — they are more likely to stick with that option. However, their meta-analysis also reveals substantial variability in defaults' effectiveness, suggesting that both *when* and *how* defaults are deployed matter.

The finding that defaults are more effective in consumer domains directly applies to browser usage. Since choosing a browser is essentially a consumer decision today, pre-installed browsers

[1] Jon M. Jachimowicz, Shannon Duncan, Elke U. Weber, Eric J. Johnson. "When and why defaults influence decisions: a meta-analysis of default effects", https://www.cambridge.org/core/journals/behavioural-public-policy/article/when-and-why-defaults-influence-decisions-a-metaanalysis-of-default-effects/67AF6972CFB52698A60B6BD94B70C2C0#article

with default settings are likely to remain the users' preferred choice.The analysis has also discovered that defaults are more effective when they operate through *endorsement* (defaults that are seen as conveying what the choice architect thinks the decision-maker should do) or *endowment* (defaults that are seen as reflecting the status quo).

These mechanisms explain why people often stick with their default browsers, even when alternatives are readily available. It also sheds light on why tech companies invest heavily in having their browsers pre-installed, set as defaults, or acquiring a reputation as the go-to browser. Such strategies significantly influence Internet users' decision-making and, by extension, shape our perception and understanding of websites and ultimately the web itself.

Exposing the wires: browser extension as a tool to understand what a website is.

In the vast ecosystem of the Internet, websites often present themselves as sleek, unified experiences. Yet beneath this facade lies a complex network of code, scripts, and data flows that shape our online interactions. This chapter introduces a strategy that allow us to peel back different layers of a website and gain a deeper understanding of the web's hidden architectures: browser extensions.

Browser extensions, also known as add-ons, are software modules that can alter, subvert or distort our browsing experience. While commonly associated with content blocking or user experience improvements, these versatile tools offer far more than mere convenience. I would like to argue, that a browser extension, along with browser's developer tools and "View source" capability is an invaluable tool to gain a deeper un-

derstanding of what particular website is through intervening on the webpages.

By intervening directly in webpage rendering and functionality, extensions allow us to expose underlying structures and understand particular website design choices and their implications. Beyond their practical applications, browser extensions can become essential tools for researchers, developers, and (web-)curious individuals seeking to demystify the websites. Through hands-on interventions and real-time modifications, these tools transform passive browsing into an active process of discovery and analysis.

Browser extensibility: from bookmarklet to the browser extension

The introduction of Javascript by Brendan Eich and Netscape in 1995 opened a whole new avenue for browser customisation: *bookmarklets*, also called *favlets*. Basic idea of a bookmarklet is having a link in your browser bookmarks, clicking which runs anonymous, self executing JavaScript function as callback in the context of a web page. Bookmarklets are a simple tools, al-

lowing to modify the way one sees the website, extract data from a webpage and basically run any possible JavaScript function.

Bookmarklets became popular very fast — mostly distributed for free, they did not require installation or in-deep programming knowledge to be able to transform and manipulate web pages. In 1998 Steve Kangas launched bookmarklets.com[1] which maintained an extensive list of bookmarklets and answered frequently asked questions about using and troubleshooting ittle programs.

[1] Bookmarklets Home Page - free tools for power surfing, http://bookmarklets.com

Bookmarklets are indeed an interesting thing — since bookmarks URL in browser is usually limited to a certain number of characters, little programs have to be written efficiently and cannot be too complex and verbose. They also usually had to be minified, which is a process of changing human readable code by replacing variable and function names with a shorter, often more cryptic ones and removing identification and empty spaces as much as possible, this way reducing number of characters. A bookmarklet generally did only one thing and did it well. With bookmarklets everyone could easily put together their own “swiss knife” of tiny tools directly in their browser “Favourites”.

Increased implementation of Content Security Policy standard between 2013 –2015 in websites has caused severe problems for the bookmarklet execution and usage and combined with already existing encoding problems and not particularly sustainable architecture of bookmarklets, their usage started to decline.

Web browsers have long offered another method for modifying and interacting with web pages, known as add-ons, plugins, or extensions. While bookmarklets are considered a more “hacky” approach, extensions became the officially endorsed way to expand browser capabilities and functionality. However, for quite some time, managing and installing extensions was challenging, making them less appealing than bookmarklets, which were easily shared and straightforward to save and use.

The landscape changed in 2007 when Mozilla Firefox introduced its Add-ons Gallery. This solution transformed how users discovered and installed extensions by providing a searchable and organised index of add-ons that developers could contribute to. These officially supported add-ons underwent security reviews and could be installed with a single click. This development effectively marginalised bookmarklets.

Although bookmarklets remained in use, they gradually became less prominent and fell out of favour.

The modern approach to browser extensibility dates back to 1997 when Internet Explorer added support for extensions in its 4th version. Firefox and Chrome, have supported extensions since their respective launches in 2004 and 2009. Safari followed suit the following year, adding extension support in 2010. Major browser vendors each maintain their own browser extension marketplace where extensions undergo a security review and updates can be automatically distributed. As more people discovered the benefits of extensions, word-of-mouth recommendations drove adoption rates, creating a snowball effect of increased awareness, usage, and development of browser extensions.

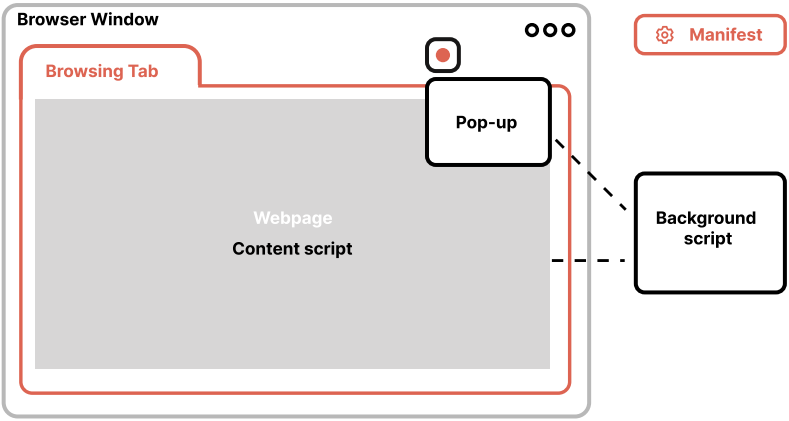
In 2024, Matt Zeunert from Debug Bear[1], having analyzed the category listings of the Chrome Web Store, reported 111,933 extensions available for installation. Interestingly, he also found that 85% of Chrome extensions have been installed fewer than 1,000 times, indicating that only a very small number of extensions ever reach wide popularity.

[1] Chrome Extension Statistics: Data From 2024, https://www.debugbear.com/blog/chrome-extension-statistics

Despite the fact that most extensions struggle to gain widespread adoption, their continued development and availability demonstrate the enduring value of custom tools enhancing web browsing. Browser extensions offer a level of customisation and functionality enhancement that is unmatched by built-in browser features. As long as there are web users with diverse interests, needs and preferences, there will be a place for browser extensions.

Anatomy of a browser extension

An extension consists of a collection of files, packaged for distribution and installation. The structure of a browser extension is rather flexible and often depends on particular developer's practice and the overall functionality of the extension. The only file that must be present in every extension is a browser extension manifest (manifest.json). It contains a number of metadata, describing the extension and request the permissions it requires. It also provides pointers to other files in the extension such as background and content scripts, markup for sidebars, popups, and options pages and other web accessible resources.



Main parts of a browser extension. Background script can communicate with browser API, content script as well as pop-up page. The permissions and resources are declared in manifest.

Background scripts allow the extension to respond to events that occur in the browser independently of the lifetime of any particular web page such as navigating to a new page, removing a bookmark, or closing a tab. Background scripts run in the context of a special page called a background page and do not get direct access to web pages. However, they can load content scripts into web pages and communicate with them through messaging API.

Content scripts are loaded into web pages and run in the context of that particular page, allowing direct manipulation of the page's DOM, just

like the normal scripts loaded in the web page. However, content scripts get a “clean” view of the DOM — content script does not see variables defined by web pages’ scripts and redefined properties. One practical consequence of this behaviour is that a content script doesn't have access to any JavaScript libraries loaded by the page, for example if web pages loads jQuery, the content script does not know a thing about it. Content scripts also cannot directly access normal page scripts but can exchange messages with them using the messaging API.

An extension can include various user interface components whose content is defined using HTML document, for example a ‘popup’, a dialog window that shows when the toolbar button is clicked or ‘options’, a page that's shown when add-on's preferences in the browser's native add-ons manager is accessed. All of these are a type of ‘extension pages’, which can include CSS and JavaScript files, just like a normal web page.

Background scripts allow the extension to respond to events that occur in the browser independently of the lifetime of any particular web page such as navigating to a new page, removing a bookmark, or closing a tab. Background scripts run in the context of a special page called

background page and do not get direct access to web pages. However, they can load content scripts into web pages and communicate with them through messaging API. Content scripts are loaded into web pages and run in the context of that particular page, allowing direct manipulation of the page's DOM, just like the normal scripts loaded in the web page. However, content scripts get a “clean” view of the DOM — content script does not see variables defined by web pages’ scripts and redefined properties. One practical consequence of this behaviour is that a content script doesn't have access to any JavaScript libraries loaded by the page, for example if web pages loads jQuery, the content script does not know a thing about it. Content scripts also cannot directly access normal page scripts but can exchange messages with them using the messaging API.

An extension can include various user interface components whose content is defined using HTML document, for example a ‘popup’, a dialog window that shows when the toolbar button is clicked or ‘options’, a page that's shown when add-on's preferences in the browser's native add-ons manager is accessed. All of these are a type of ’extension pages’, which can include CSS and JavaScript files, just like a normal web page.

The flexible structure of browser extensions is their absolute superpower. Like a Swiss Army knife for the web, these versatile tools allow us to rapidly craft targeted tools for an array of needs. The beauty of browser extensions lies in their precision. They can target specific web pages with accuracy, modifying content or adding functionality exactly where it's needed. At the same time, they can operate on a browser-wide level, handling tasks that span across multiple tabs or windows. In some way, browser extensions may embody the ideal of the web itself: a space where small, clever solutions can have a big impact on how we interact with the digital world.

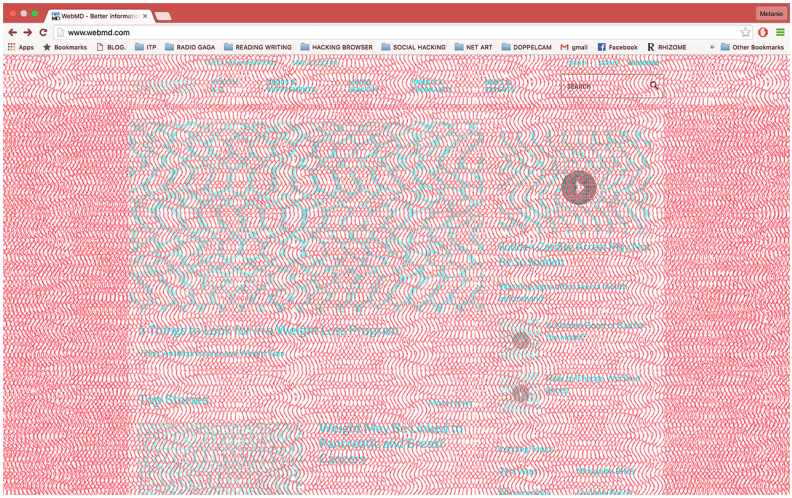
Changing the perspective

While majority of browser extensions are aimed towards optimisation of workflow and productivity-related tasks like blocking advertisement, translation, grammar checking, and password management, there is a great number of extensions, which change the way web page is rendered. These extensions emphasise certain aspects of web architecture or explore the possibilities of how a website can be hijacked and transformed.

Digital artist Melanie Hoff's project, 'Decodelia'[1], consists of a browser extension paired with specialised eyewear. This combination employs fundamental colour theory principles to transform web content into complex, unreadable patterns. The transformed content becomes legible only when viewed through red-tinted glasses. This project acknowledges that there is more than one audience for personal screens and uses low-tech methods to split the audience of the website in two: those who can decipher the content and those who cannot. This simple yet powerful approach adds a new dimension to the ongoing dialogue surrounding digital privacy in shared spaces.

[1] DECODELIA on Melanie Hoff's Website, https://www.melaniehoff.com/DECODELIA/

Suggested situation of Decodelia's usecase: browsing/working in the public spaces.



Decodelia browser extension is changing the way web page is displayed, rendering website only readable with red-tnted glasses.

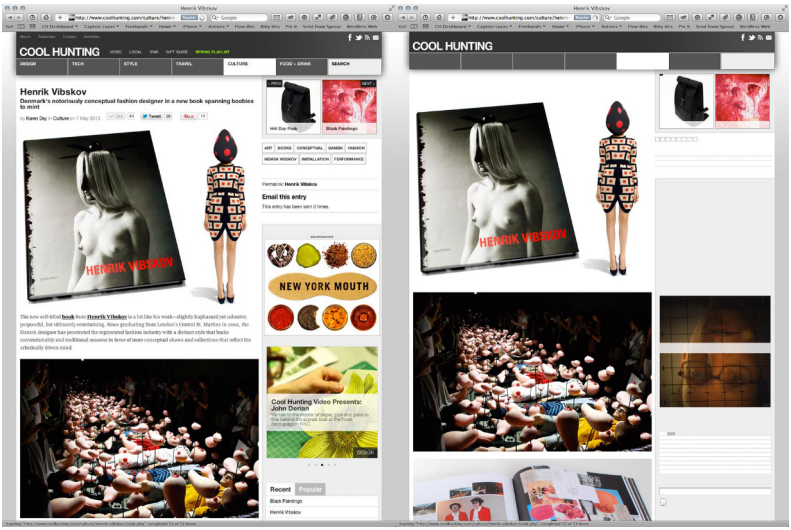
The ‘Wordless Web’ project, launched in 2012 by designer Ji Lee, is a unique browser extension that strips away all text from websites, leaving only images and empty containers behind. This simple yet powerful tool transforms the Internet into a visual gallery, highlighting the significance of imagery on the web and encouraging to appreciate pictures independently, rather than as mere accompaniments to text.

Lee[1] explains the concept:

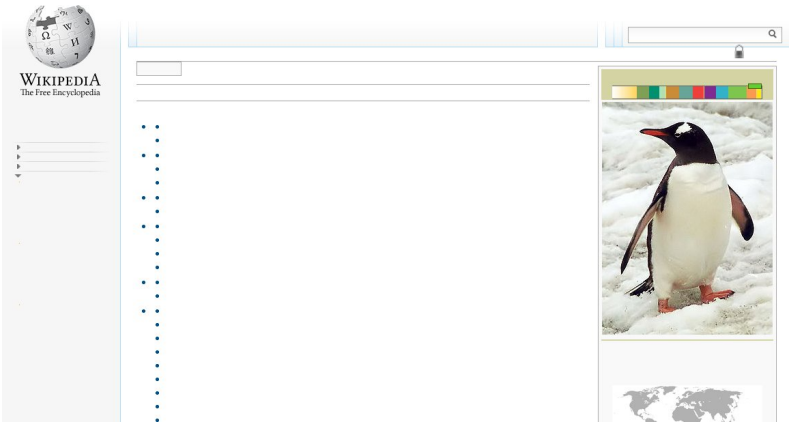
> “No text means no context,” says Lee. “You’re free to enjoy the images in their purest form, without names, labels, definitions, or purpose. It makes the pictures we see across the web more mysterious and open to interpretation of our own imaginations.”

This unusual approach to web browsing yields varied results. Some sites become more engaging when freed from textual constraints, while others devolve into what appear to be expansive digital billboards. Regardless of the outcome, ‘Wordless Web’ offers an intriguing exploration of the Internet landscape by inverting a fundamental aspect of website design.

[1] “Wordless Web”, last modified 08.05.2012, https://coolhunting.com/tech/wordless-web-ji-lee/



Before and after: The Wordless Web bookmarklet used on the "Cool Hunting" website.



Wikipedia article on penguins with all text elements "nuked" from the web page.

Rafaël Rozendaal's ‘Abstract Browsing’ from year 2014 is a multifaceted project encompassing both digital and physical elements. The digital component is a browser extension that transforms any website into a vibrant, abstract composition, stripping away the content. This temporary shift in perspective reveals what Rozendaal calls "the skeleton of the web." In the “Notes on Abstract Browsing”[1] Rozendaal explains:

> "Technology asks new questions about composition. I'm looking for unusual compositions. Anti-compositions, unhuman compositions, compositions that humans would not have created on their own."

The artist dedicated considerable time to exploring the Internet using this extension. When he encountered a composition that struck him, he captured it. Similar to digital photography, this process yielded a vast number of images. The true challenge lay in selecting which compositions would be materialised into physical form as tapestries.

These browser extensions challenge our perception of the web, transforming the digital landscape into new forms of art and expression. By altering how we interact with and view online

[1] “Notes on Abstract Browsing - Rafaël Rozendaal”, https://www.newrafael.com/notes-on-abstract-browsing/

content, artists, designers, and developers invite us to reconsider the nature of digital spaces and the relationship between the viewer and the website. As we continue to navigate an ever-changing Internet landscape, these projects remind us of the malleable nature of our online experiences and the potential for technology to reshape our understanding of the Internet itself.

Abstract Browsing 17 08 10X (Google Docs, Feedly, iMore, Top Ten Reviews, Waze, Reviews, Amazon, Nerdwallet, Google Drive, Twitter)

Little sharp tools: browser extensions documentation

As a practical component, I aim to develop a series of small browser extensions that help reveal hidden aspects of web architecture. Making these aspects visible allows for a better understanding of a web page's structure. But what aspects can be considered hidden? For the scope of this project, I define them as elements of website architecture that are not rendered visible by default browser representation. Another important consideration in searching for hidden aspects is their prevalence. I want to visualise architectures that are widespread to avoid (as much as possible) situations where my “little tools” yield no results.

Given the current dominance of Google Chrome in the browser market and its advanced API for browser extensions, I have chosen to develop these extensions for Chrome Version 128.0.XX (where XX represents minor updates and fixes not accounted for in this project).

The general methodology applied to all browser extensions in this series is largely based on Document Object Model (DOM) scripting. DOM scripting leverages the fact that a fully parsed and rendered web page exposes a set of interfaces, allowing for manipulation and processing of elements on the page. The execution of the content script is controlled by the background script, which awaits the manual activation in the browser window.

Borderline Explorer

Website layouts in the age of templates, frameworks, and integrations built on top of continually developing web standards have become rather complex, not always exactly matching the layout of the rendered page. This extension utilises a low-tech method of visualising this hidden architecture — it draws borders around all rendered elements, revealing the true dimensions of elements and their position in relation to one another as well as the overall composition of a web page. Additional color-coding denotes different types of web page elements, helping to gain insight into how a particular layout is composed. To visualise dimensions of website "building blocks", this extension adds a

number of auxiliary style rules, which are visually distinctive from those of the website. This browser extension enqueues/removes a stylesheet containing a list of CSS selectors, highlighting certain categories of elements: non-semantic elements, semantic containers, content (i.e., text, images, video, or audio), embedded content, canvas, and inherently interactive elements - links, forms, and buttons.

As always, this approach is not free of certain challenges. The first one I would like to discuss is the layout shift introduced by borders. Since the width and height of website elements are calculated according to the CSS box model, introducing even a minimal 1px border increases an element's width and height, which can, in certain circumstances, create a layout shift, especially if the layout relies on non-relative units. In most cases, it is tolerable in the context of this project, just introducing a small deviation from the original layout, but I have also encountered cases where even such a minimal shift breaks the layout completely. Another consideration is the choice of color-coding, since every website is built upon its own color scheme, and the images, graphics, and video introduce their own scheme, making a universal or website-based solution disproportionately hard to achieve. In

the end, it was decided to implement a small graphical interface that allows changing the default highlighting colors in case they match too well with the overall web page appearance. Additionally, this opens up room for expression, since different colors have certain associations in the context of interface and web design (e.g., red is error, green is success, yellow is warning). This way, highlighting the composition can be used to communicate and emphasise different aspects.

Hide&Seek

There are a lot of reasons web developers want some parts of their page hidden, from orchestrating "user experience" to just get rid of something really quickly, without rebuilding more complex parts of the website. This extension helps locate hidden elements, by rendering all intentionally hidden elements on a page visible and hiding everything else.

There are many options when it comes to hiding page elements, so the process of finding them contains several steps. First, we create an array with class names, which are commonly used in different frameworks and practices to hide ele-

ments. Next, we search for elements in the current page to match any of those classes. After that, we check if any of the elements of the website have a “hidden” attribute or inline CSS Rule attached to it. Querying CSSOM helps us to locate further CSS rules, hiding elements, in cases the webpage maker was creative and didn’t use conventional name for “hiding” class.

There are however several limitations of this approach. Lots of hidden web page elements have a size of 1x1 px or even 0x0 px, which is also a technique for hiding elements. Even after being “made visible” they are so small that they cannot be seen on screen. Thus, it may need some additional highlighting to draw attention to their position. Another “hiding” technique that needs its own approach is setting CSS rules for “color” and “background-color” to the value of “transparent” or matching the background of parent element(s). Finding universal approach to locating and making such elements visible is unattainable, since the style rules are tied to the markup, which can vary greatly. Finally, there is a case, where there are no hidden elements on the page been found by the extension. To avoid unnecessary confusion, the content script will inform us that this particular page has no hidden elements exposed to the browser extension.

Headspace

The head of an HTML document is the part that is not rendered in the web browser when the page is loaded, but contains a lot of information about the web page. The head is not meant for humans; it is more of a place containing metadata and resources, helping browsers and other applications make sense of the web page. Luckily, it is in the nature of every inquisitive human being to explore the very places that are not meant for them.

One can inspect the contents of this mandatory part of the website using the browser's developer tools, but it often requires certain experience in website making to interpret what is there. After all, this section is designed for parsing, not reading. This browser extension imagines what a <head> element might look like if it were rendered visible. Since the markup is already there, the approach here would be similar to the one used in “Borderline Explorer” – enqueue an additional stylesheet, which targets only elements in the head of the HTML document. This list of style rules takes advantage of modern CSS selector capabilities to access element attributes as well as DOM scripting to enhance markup. The goal, however, is to keep as

much of the original markup as possible while giving the normally invisible parts of the website their own visual “space” to inhabit.

Field Expedition

By reimagining the landscape of a website as an unexplored terrain, this extension offers a thought-provoking experience that challenges conventional web navigation. Upon activation, the visual presentation of the webpage is immediately altered by blurring all text, media, and focusable elements into a foggy, indistinct landscape. This transformation invites us to approach the familiar environment with a spirit of exploration. As a navigation aid, there is a small tooltip which hints at the overall size of the webpage and current cursor position, and the vague outlines of the website elements. The process of unveiling the webpage's contents is deliberate and methodical. One must carefully scan the landscape, moving the cursor with precision to locate the central points of various elements. As these points are discovered, the corresponding elements return into focus.

The experience of navigating a webpage with this extension enabled bears some resemblance

to navigating webpages with assistive technologies. It is an inherently slow and tedious process that shifts the focus from consuming content to understanding the fundamental organisation of digital spaces.

Certain components of the webpage remain continually obscured, either due to their size (being too small or too large to pinpoint accurately) or their hidden nature within the page's markup. This aspect of the extension serves as a poignant reminder of the complexities and occasional inaccessibility inherent in web design.

Resources

Anastasia Kubrak. “User-Agent: If everything is so smooth, the why I am so sad?”, https://anastasiakubrak.com/User-Agent.pdf

Becca Abbe. “The Internet’s Back-to-the-Land Movement”. https://www.are.na/editorial/the-internet's-back-to-the-land-movement

Cody Lindley. “DOM enlightenment: Exploring the relationship between JavaScript and the modern HTML DOM”, https://domenlightenment.com/

Dan Rubin. “Off the page”, Back office #3 Writing the screen, 2019

Eric Lawrence. “Demystifying browsers”, https://textslashplain.com/2020/02/09/demystifying-browsers/

Garry Ing. “a view source web”, https://viewsource.info/

Geert Lovink. “Stuck on the platform. Reclaiming the Internet.” Valiz, Amsterdam, 2022

Hito Steuerl. “Too Much World: Is the Internet Dead?”, e-flux, https://www.e-flux.com/journal/49/60004/too-much-world-is-the-internet-dead/

HTML Energy Podcast, https://html.energy/podcast.html

J.R. Carpenter. “A Hand-made Web”, https://luckysoap.com/statements/handmadeweb.html

Jay David Bolter, Diane Gromala. “Windows and Mirrors. Interaction Design, Digital Art, and the Myth of Transparency”, The MIT Press, 2005

Kristofer Tjalve, Elliot Cost. “Diagram Website”, https://diagram.website/

Laurel Schwulst. “My website is a shifting house next to the river of knowledge. What could yours be?”, https://thecreativeindependent.com/essays/laurel-schwulst-my-website-is-a-shifting-house-next-to-a-river-of-knowledge-what-could-yours-be/

Louise Drulhe. “Critical Atlas of Internet”, https://louisedrulhe.fr/internet-atlas/

Lukas Engelhardt. “New dependencies”, https://networkcultures.org/longform/2022/03/29/new-dependencies/

Maisa Imamović. “The Psychology of the Web Developer, Reality of a Female Freelancer”, DeepPockets #4, https://networkcultures.org/blog/publication/the-psychology-of-the-web-developer-reality-of-a-female-freelancer/

Manuel Lima. “Visual complexity Mapping patterns of information”, Princenton Architectural Press, 2011

Maximilian Kiepe. “event.preventDefault() — another view on designing websites”, https://event.preventdefault.net/

Olia Lialina, Dragan Espenschied. “Contemporary Home Computing”, http://contemporary-home-computing.org/

Olia Lialina, Dragan Espenschied. “Digital Folklore Reader”, merz & solitude, 2009.

Tiziana Terranova. “After the Internet. Digital networks between capital and common”, Semiotext(e), 2022.

Colophon

Master Thesis in partial fulfillment of the requirements for the degree Master of Arts in Digital Media at the University of the Arts Bremen, October 2024.

Author
Christine Brovkina

Supervisors
Prof. Dr. Andrea Sick
Prof. Dennis Paul

Univeristy

HFK BREMEN
Hochschule für Künste
University of the Arts
Bremen

Submitted
October 2024, Bremen, Germany