University of Tartu

Institute of Philosophy and Semiotics

Department of Semiotics

Risto Rõõmus

The Reconfiguration of the Web: Open Standards, Platform Power, and Developer Configuration

Bachelor's Thesis

Supervisors: Katre Pärn, Auli Viidalepp

Tartu

2025

Abstract

This thesis examines the implicit politics underpinning the early Web and how web technologies originally designed for openness and decentralization can become reconfigured as instruments of platform control. Drawing on Langdon Winner's analysis of the politics of artifacts (1980, 1986) and Lucy Suchman's concept of configuration (2012), alongside platform studies scholarship on infrastructuralization and lock-in, the thesis analyzes two case studies. First, it traces the evolution of early web technologies, showing how cookies transformed from privacy-preserving solutions into foundations of surveillance capitalism. Second, it examines React as a corporate-owned open-source framework, analyzing its licensing controversies, its capacity to configure developer cognition, and Vercel's growing influence over its direction. The thesis shows how technologies designed as open can be shaped to serve commercial interests while maintaining the appearance of openness.

Keywords: web technologies, configuration, platformization, open-source, developer practices

Table of Contents

Introduction

Web development has over time evolved from a simple, standards-based and technologically decentralized practice to a complex and multilayered one with a growing dependency on technologies and infrastructures built and provided by large corporations. A subset of these technologies operates on a middle ground: they are open-source, free to use, yet ultimately built by companies with a vested interest in shaping the Web[1] towards their interests. How do these open-yet-closed dependencies shape these technological tools and the wider Web? What kind of role do different actors (platforms, foundations, interest groups and developers) with their individual power dynamics play in these changes? And what sort of impact does their growing dominance have on the practices, mental models and professional autonomy of those who build the web?

Inspired by Langdon Winner's analysis of technological politics (1980, 1986), the core aim of this thesis is to show how the World Wide Web, as a technological artifact, embeds and enforces particular arrangements of power. I will firstly examine how the Web was designed and whether its core requirements can be construed as forms of politics, and how its early innovations were primarily driven by the ideals of its creators. Using the browser cookie as a case study, I will show how these technologies, despite best intentions, found themselves exploited by commercial interests in ways their designers actively opposed.

I will then analyze the open-source but privately owned web development framework React as a modern-day example of these very same tendencies. By applying Lucy Suchman's concept of configuration, I will analyze aspects of React's licensing, its governance and how external private interests are increasingly taking control of its direction. Further attention will also be given to how React shapes not only what developers build, but how they think about building.

---

[1] Throughout this thesis I will refer to the Web in capital letters when it concerns the concept and the object itself. Extensions and outgrowths from the Web such as 'web development' and 'web technologies' will be in lowercase.

Research Questions:

1. How did Berners-Lee's requirements for the Web embed political arrangements into its architecture?

2. Using cookies as an example, how can technologies designed with specific boundaries be repurposed to enforce power dynamics beyond their intended use?

3. What are the implications of corporate control over open-source frameworks, and what boundaries does such control draw?

4. How do frameworks like React shape the practices, mental models, and professional autonomy of web developers?

5. Lastly, how does Vercel's growing influence over React's direction illustrate the mechanisms through which platform interests encroach upon open-source technologies and how are these mechanisms challenged?

1. Theoretical foundations

This chapter establishes the theoretical foundations for analyzing how web technologies, despite being framed as open and neutral, become instruments of control serving particular interests, and which mechanisms both enable and obscure this transformation. Three separate, but complementary perspectives will be used.

Langdon Winner's work on the politics of technology (1980, 1986) provides the core inspiration for this thesis: the claim that technologies are not neutral tools but can embody and enforce particular arrangements of power, which are always in contestation. His 1980 essay establishes this general framework, while his later analysis of decentralization offers specific tools for examining how power is distributed within technological systems. Lucy Suchman's concept of configuration (2012) offers a method for examining how technologies and their users mutually shape one another, with a focus towards boundaries, temporalities and individual imaginaries. Finally, platform studies scholarship (Plantin et al. 2016; de Reuver et al. 2018) provides concepts such as infrastructuralization and lock-in for understanding the specific mechanisms through which contemporary digital technologies achieve and maintain dominance. Together, these frameworks allow examination of not only that technologies are political, but how they become so and what sustains their power.

1.1 Langdon Winner on technological artifacts and their politics

Langdon Winner's essays “Do Artifacts Have Politics?” (1980) and “Decentralization Clarified” (Winner 1986)[2] provide the foundational framework for this thesis, offering both theoretical

[2] The essay originally appeared in 1983 as Winner, Langdon 1983. Decentralization: Its Meaning in Politics and Material Culture. — Durbin, Paul T. (ed.). Research in Philosophy and Technology. Greenwich: JAI Press, 43–52. This thesis paper will use Winner’s 1986 book as the source.

claims as well as analytical tools. While his writings predate the invention of the Web, they manage to offer a unique and surprisingly applicable lens by which to investigate the dynamics of power within it as well as between the technological outgrowths that followed.

Langdon Winner's 1980 essay questions the assumption that technologies, while heavily interwoven into modern politics, are in and of themselves neutral tools, whose social impacts depend only how they are embedded in their respective social and economic system, and how people use them within that system. This position, which Winner calls the "social determination of technology," treats technologies and their impacts as interchangeable with any other object of social analysis, "essentially no different from the social determination of, say, welfare policy or taxation" (Winner 1980: 122). While this view is analytically more useful than naïve technological determinism, which sees technology as a single, immutable driver of change, it still has a critical shortcoming: "taken literally, it suggests that technical *things* do not matter at all" (ibid., 122). Instead, he argues for a third position, that technical artifacts can embody specific forms of power and authority, or in other words, that they are *inherently* political (ibid., 123 – emphasis in original) and that we should "pay attention to the characteristics of technical objects and the meaning of those characteristics" (ibid., 123). He presents two distinct ways this may happen, using examples from industrial and infrastructural technologies of his time: urban planning, factory machinery, and energy systems.

The first is about design: technologies can be deliberately designed to enforce power dynamics or settle an issue in particular ways. His famous (and at the time somewhat controversial) example concerns a series of parkway overpasses designed by New York's powerful urban planner Robert Moses. These overpasses are scattered around the wider Long Island area in New York, crisscrossing the roads to a more affluent set of smaller islands. He alleges that they were designed and built to be too low for buses to pass through, limiting its access only to cars and thus only to people who could afford them at the time (ibid., 124). In this way a seemingly neutral technology, a bridge, became political as it enforced certain class and racial divides present at the time. Another example he highlights is from industrial manufacturing in the 19th century. A factory in Chicago installed new, untested pneumatic molding machinery at great cost, ostensibly to modernize the factory and give unskilled laborers work opportunities. In practice these machines produced inferior results at higher costs, but the actual aim of their implementation was realized a few years later: the destruction of the local union. Once that was achieved, the machines were abandoned (ibid., 125).

In both examples he argues that the technology had been "designed and built in such a way that it produces a set of consequences logically and temporally *prior* to any of its professed uses" (ibid., 125). Neither bridges nor molding machinery are intended to do harm, but, like a deck of cards in a game, they can be "stacked" in favor of certain social interests (ibid., 125). And as such technical systems settle social issues not through explicit debate but through the material arrangements of "steel and concrete, wires and transistors, nuts and bolts" (ibid., 128).

The second category is about what Winner calls "inherently political technologies": systems that require or are strongly compatible with particular forms of social organization (ibid., 128). He gives two versions of this claim, a strong and a weak one. The strong version posits that certain technologies *demand* the creation and maintenance of a particular set of social conditions to function or even exist in the first place, primarily out of practical necessity (ibid., 130). The weak version meanwhile suggests a strong compatibility rather than an absolute requirement for it (ibid., 130). The examples he provides pertain to energy: solar energy is more compatible with decentralized, democratic arrangements[3], while nuclear power is more compatible with hierarchical control structures, yet neither strictly necessitates these forms[4] (ibid., 133).

Moral claims against such structuring are usually dismissed: one cannot question the enforcement of hierarchies for a factory, railroad or a ship without being dismissed as a naïve idealist, someone who doesn't understand what is needed to maintain such systems as smoothly working entities. Patterns of authority which work effectively in corporations become "the desirable model against which to compare political and economic relationships in the rest of society" (ibid., 133). And once those patterns are in place and embedded in the technology itself, their durability may make them near impossible to remove.

---

[3] The article was published in 1980, which saw a growing interest in solar panels as an after effect for the price hike of crude oil during the 1979 energy crisis.

[4] He also points out how nuclear energy in particular involves risks not only to the environment, but also to civil liberties, given the strict security requirements involved for those employed at the reactors. This is doubly relevant today, given how much of the Russo-Ukrainian war centers on energy infrastructure and the need to protect it.

1.1.1 Decentralization

Winner also provides analytical tools for examining aspects of power distribution by looking at the concept of (de)centralization. In his chapter "Decentralization Clarified," (Winner 1986) he argues that claims about technologies being (or wanting them to be) "centralized" or "decentralized" first require specifying the aspect assumed to be problematic. In lieu of a definition of the word he proposes five key questions one could ask about it: How many centers are there? Where are they located? What is their relative power? What is their diversity and vitality (ibid., 86-88)? These questions would help define what decentralization would mean in a particular scenario, as well as we help us "notice who cares about the idea and why" (ibid., 88).

Questions of centralization apply not only to political institutions but equally to "the ownership, initial source, and conditions of production, distribution, consumption, or use of goods and services" in material culture (ibid., 91). He illustrates this with the petroleum industry: the existence of multiple competing oil producers, spread around the globe, does not make the industry decentralized when each producer remains "huge, extremely powerful, and centrally controlled within themselves" (ibid., 91).

He makes note of the fact that consumption continues to be "centered in the individual", but our ability to have control over production or any decisions beyond immediate gratification is lacking (ibid., 93). This is a worsening tendency as the design of many technologies "embodies centralization in material form" as the production of increasingly complex technologies requires further centralization (ibid., 94). This, he concludes, means that decentralization is now less of a vision of a better structuring of a society, but rather a faint hope that "one may still create institutions here and there that allow ordinary folks some small measure of autonomy" (ibid., 96).

1.2 Suchman on configurations in technology

The second author who I will use extensively for this thesis will be Lucy Suchman and her concept of configurations from the 2012 article of the same name (Suchman 2012: 48-60). Below I will try to describe the key aspects of it: configuration as a method and object; its

perpetually shifting boundaries and fundamental incompleteness; figuration and the reanimation of it; and lastly imaginaries, their designers and users.

Initially framing it as a device, configurations can be seen both as a method of analysis for individual artifacts and the socio-technical systems they constitute, as well as the very nature of them. As a method, configuration can help delineate boundaries between objects as it "brings things together – at once reiterating the separate existence of the elements assembled, and drawing the boundaries of new artefacts" (ibid., 50). These boundaries may appear as if they were present from the start when in actuality they are redrawn continuously as ongoing consequences of different socio-technical engagements (ibid., 50). These redrawings result in artifacts and their configurations to be characterized by incompleteness (ibid., 56).

Figuration is the action through which configurations take shape. To figure is "to assign shape, designate what is to be made noticeable and consequential, to be taken as identifying" (ibid., 49). Figuration holds the material and semiotic together in ways that become naturalized over time. Initial choices in design become future inevitabilities, ending up obscured and invisible in the process. This in turn calls for the act of "unpacking" to recover the constituent elements that have been obscured, which then need to be reanimated (ibid., 49).

Reanimation of the figure is to make visible the obscured, in order to "recover the practices through which it comes into being and sustains its effects" (ibid., 49). When technical practices appear as simply "how things are done," reanimation makes visible the specific design decisions, institutional interests, and ongoing work that maintains this appearance of inevitability.

A core aspect of configurations are materialized imaginaries: visions, ideas and conceptions that are made material through artifacts, which then shape ("narrate") future imaginaries. This in turn will lead to borders being redrawn further since a configuration is never fundamentally finished but always enacted and re-enacted. Things are "fixing them through reiteration but also always engaged in 'the perpetuity of coming to be'" (ibid., 50). The "fixity" of an artefact is an effect of reiterative enactments of a particular subject/object configuration, while fluidity articulates the inherent multiplicity always present beneath apparent stability (ibid., 56).

In discussing imaginaries Suchman makes mention of a dynamic between the designer of an imaginary and its user, which can be both an imagined user as well as a real one. First, she states that we need to see both more and less of the designer: more, because designers are situated within particular practices and exigencies and less because artefacts are already

characterized by open-endedness regarding how they might be incorporated into use (ibid., 56). Suchman then goes on to say that designers appear to carry some sense of primacy over the configuration while also situating themselves outside of it. And the user meanwhile assumes the position of "manifestly absent" in design and that we need to disregard the categories of the user and the designer altogether (ibid., 57). While I see clear merit in the latter, I find it hard to agree with her absolutist stance on the user's absence in design, in both the general and the configuration-specific meaning of it.

Over time, configurations can also evolve from artifacts and systems to become what we recognize as skills or expertise: "a new life-world in which new agents interact and move" (Knorr Cetina 1999: 219[5], cited in Suchman, 2012: 56). Technologies don't simply influence pre-existing users but actively produce particular kinds of subjects, or as put succinctly: objects make subjects. Suchman cites Woolgar (1991) that "by setting parameters for the user's actions, the evolving machine effectively attempts to configure the user" (Grint and Woolgar 1997: 71)[6]. This is not shaping an individual actor but "the incorporation of the user into the socio-material assemblage that comprises a functioning machine" (ibid., 55). Agencies of subjects and objects are figured together, which is why the specificities of configuration carry political stakes (ibid., 52).

She finishes her article by providing a certain degree of interpretative freedom for configuration as a tool by stating that *as a method,* configuration does not contain inherent boundaries, nor strict instructions for its use. Rather one would use it to "reanimate the figures that populate our socio-material imaginaries and practices, to examine the relations that they hold in place and the labours that sustain them, and to articulate the material semiotic reconfigurations required for their transformation" (ibid., 58): to reveal, reanimate and eventually reconfigure the semiotic and material contained within technological configurations.

---

[5] Knorr Cetina, Karin 1999. Epistemic Cultures: How the Sciences Make Knowledge. Cambridge, MA: Harvard University Press.

[6] The essay originally appeared as: Woolgar, Steve 1991. Configuring the user: The case of usability trials. In J. Law (ed.), A Sociology of Monsters: Essays on Power, Technology and Domination (pp. 57–102). London: Routledge.

1.3 Platformization and lock-in

The third theoretical perspective to be used in this thesis concerns platforms within the digital realm and how they become infrastructural dependencies. The term platform is used near-ubiquitously across a variety of contexts, which, combined with its "distributed nature and intertwinement with institutions, markets and technologies" (de Reuver et al. 2018: 124), has led to a "wide variety of conceptualisations" across disciplines (ibid.: 127). For this thesis, a software-based platform can be understood as "the extensible codebase of a software-based system that provides core functionality shared by the modules that interoperate with it and the interfaces through which they interoperate" (Tiwana et al. 2010[7], cited in Sandoz and Stiefel 2022: 1). Platforms can be viewed as a specific type of Suchman's configuration, one where boundaries are primarily drawn around infrastructures through a variety of technological means and almost always to commercial ends.

Plantin et al. (2016) identify two interconnected processes shaping digital environments: the "infrastructuralization" of platforms and the "platformization" of infrastructures. Infrastructuralization occurs when platform-based services become "so ubiquitous and deeply embedded" that they become essential, thus acquiring the characteristics of public utilities while remaining under corporate control (ibid., 3). Conversely, platformization moves in the opposite direction: previously open or public systems become reorganized under platform logic, with corporate entities extending their reach into domains that were once organized differently. Together, these processes create a fundamental conflict where "media environments increasingly essential to our daily lives (infrastructures) are dominated by corporate entities (platforms)" (ibid., 3). Once technologies become embedded in this way, replacement grows difficult because it would require reorganizing the practices built upon it.

Central to both processes is lock-in which Plantin et al. describe as "the accumulation of dependencies that make alternatives impractical" (ibid., 9); users or developers become so dependent on a particular technology that switching to alternatives becomes prohibitively difficult or costly. "Achieving lock-in is among platform builders' principal goals" (ibid., 9),

---

[7] Tiwana, Amrit; Konsynski, Benn; Bush, Ashley 2010. Platform Evolution: Coevolution of Platform Architecture, Governance, and Environmental Dynamics (Research Commentary). *Information Systems Research* 21(4), 675–687.

achieved not through coercion but by actively discouraging interoperability with competitors, forcing developers “either to commit to just one platform or to build and maintain multiple versions of the same product” (ibid., 9). Each dependency added raises the cost of exit and eventually the platform becomes difficult to leave regardless of whether better alternatives exist.

***

Winner’s work establishes that technologies can embody political arrangements rather than serving as purely neutral tools and that these politics may manifest in various forms. Suchman provides configuration, an analytical framework to analyze how technologies and users mutually constitute one another by establishing and redrawing boundaries, informed through their imaginaries. Lastly, platform studies provides us with the concept of platformization to describe the formation of a specific kind of configuration, and lock-in as the means to make this configuration durable and difficult to escape.

2. The evolution of the Web

The early Web was shaped by deliberate design choices to favor simplicity and openness. In this chapter I examine those foundations and trace how early technologies such as cookies saw commercial exploitation in ways their creators did not intend or even actively tried to prevent.

2.1 “Vague but exciting…”

The comment in the title was a scribbled note by Mike Sendall at CERN upon having reviewed an initial proposal he received in 1989 by Tim Berners-Lee, titled “Information Management: A Proposal” (Berners-Lee 1989[8], Kardell 2015b). Further pages reveal more comments by him, which praised the idea, but were hesitant to entirely support due to the number of unknowns it contained (Kardell 2015b). While his initial proposal was rejected, he kept working on the idea and submitted a second one in 1990 (Berners-Lee 1990). Little did he, or anyone really know how foundational this proposal would become.

The World Wide Web as it was initially known, was conceived as a solution to a problem of institutional memory. Tim Berners-Lee, working at CERN, had noticed how the high turnover of people led to a growing problem of properly stored information being either difficult to find or ending up lost entirely due to the frequent use of proprietary technologies which were unable to communicate between each other (ibid., 3). Instead, it was mostly informal, loosely-defined human-to-human structures of communication which helped maintain a steady transfer of knowledge within the organization (ibid., 3).

[8] The 1989 version of this document is provided as an annotated version by CERN’s archive (see references). Further elaborations on Berners-Lee’s proposal rely on the 1990 version.

The early Web that Berners-Lee envisioned would be the opposite of what he had been forced to use so far at CERN. Hypertext as a technology of linked documents did exist, as did the Internet as an infrastructure of networked computers spanning the globe, but a truly open solution combining both technologies had not been established. While the hypertext solutions of the late 1980s offered sophisticated features exceeding those of the early Web, they functioned as closed silos that were either operating within a closed network or even on a single isolated machine, required significant upfront costs and lacked any interoperability with other hypertext-based technologies (Kardell 2015a). Similarly, early iterations of interconnected computing also inherited their designs from the so-called "computer utility" model from the 1960s. They usually featured a single monolithic system acting as the dominant, centralized resource people would connect to, akin to public utilities such as electricity and the power plants that feed them (Plantin et al. 2016: 12), rather than a decentralized and ever-growing network of locations across the globe. One of Winner's core questions on decentralization is "How many centers are there?" (Winner 1986: 86). For the pre-web solutions the answer would essentially be one, as the user was trapped either within a hypertext silo or a singular mainframe utility.

The technological solution Berners-Lee proposed for document discovery and retrieval would be structured just as freely and unrestrained as the "web" of interconnected communications he described:

> In providing a system for manipulating this sort of information, the hope would be to allow a pool of information to develop which could grow and evolve with the organisation and the projects it describes. For this to be possible, **the method of storage must not place its own restraints on the information**. This is why a "web" of notes with links (like references) between them is far more useful than a fixed hierarchical system. (Berners-Lee 1990: 5 – emphasis in original)

Avoiding these restraints is a foundational element of his proposal and informs the political dimension of the early concept of the Web. While the concept Berners-Lee drew and the requirements he gathered were built upon internal challenges, he makes a prescient prediction that CERN is just miniature model of the rest of the world and as such the problems it faces are ones that the world will encounter in a few years' time (Berners-Lee 1990: 4).

This leads me to believe that he was aware of the concept being not just a solution to certain bureaucratic problems within an organization but a wider proposal for the free, unrestrained dissemination of information around the world. As such, the overall reasoning behind the proposal, as well as the technical design described as "CERN Requirements" (ibid., 11-12) ought to be viewed as the permissive boundaries he wants to be established to achieve these

aims for the wider Web as well. I will use these requirements as well as other parts of the document to highlight four core ideas which I deem to be particularly notable due to their political implications: his rejection of hierarchical structure, his insistence on decentralization, the requirement for heterogeneity and openness, and his explicit designation of copyright enforcement as a non-requirement.

Non-hierarchical structure. Berners-Lee explicitly states that a hierarchical structure akin to a folder tree does not represent the natural modes of structuring information (ibid., 6) and that despite there being a nominal management structure, information flows freely within CERN, enabling them to communicate, and share anything across groups (ibid., 3). He wants the Web's structure of linked information to mirror this side-stepping of hierarchical boundaries. Hierarchies within complex technical systems are what Winner describes as an essential part of "inherently political technologies" wherein hierarchical organization is presumed to be a prerequisite for the system to work efficiently, or at all (Winner 1980: 130). While the existence of such hierarchies isn't necessarily bad, they should not be seen as intractable properties of the technologies themselves; they could instead be a "pattern imposed independently by a governing body, ruling class, or some other social or cultural institution to further its own purposes" (ibid., 131). I'd argue that Berners-Lee's decision to mark a non-hierarchical structure as a precondition for the design of the Web does, in a way, mirror Winner's thinking on the matter: that avoiding a hierarchy will make it less likely for the Web to be controlled by such interests.

Decentralization. Centralization was rejected because "a new system must allow existing systems to be linked together without requiring any central control or coordination" (Berners-Lee 1990: 11). The Web must be allowed to grow freely and organically without any approval from some central authority on how to. When looking at Winner's questions on how to specify decentralization, namely their number, location, power, diversity and vitality (Winner 1986: 86-87), then Berners-Lee's proposal appears to maximize the benefits of each: once connected, there can be a vast amount of websites spread across servers located anywhere in the world, built within the highly permissive structure of HTML and consummate links, offering up diversity in form and vitality for growth based on all of the affordances of the Web.

Heterogeneity and Openness. Using the client/server model the Web should be accessible from as many systems as possible, from wherever possible by separating "the information storage software from the information display software" (Berners-Lee 1990: 15), necessitating

the development of what we now commonly recognize as Web browsers. While “private links” are important as well, the foundational structure ought to be publicly accessible by way of providing “read access to the general public” (ibid., 18). This drive towards equal access from any system directly contrasts with the earlier hypertext solutions described above, which operated only within their own closed environments. Berners-Lee wanted the Web to accommodate any system capable of implementing its open protocols, allowing participation without gatekeepers.

Finally, **copyright enforcement** as well as access-restricting data security were deemed a “non-requirement”, of secondary importance towards information exchange (ibid., 12). While authorization solutions could be designed (and they will be), he recognizes their complexity as a hindrance for the initial implementation.

Looking at Suchman, she describes figuration as the action that “holds the material and the semiotic together” (Suchman 2012: 49). Berners-Lee's proposal does just that: the material architecture (non-hierarchical, decentralized, heterogenous) binds together the semiotic values of openness and unconstrained information exchange. The Web's boundaries were drawn precisely around what was purposefully left unbound, or to put it another way, they were meant to be boundaries around any interests wanting to control it, protecting the freedom of the Web. All this resulted in a web that was meant to be open because the “technical and institutional arrangements of the system permit anyone to create visible, findable, and linkable content that is encoded using public standards” (Plantin et al. 2016: 13-14).

2.1.1 Gopher: Another Web

The Web did have a competing alternative at that time: Gopher. It was similar to the Web in having a protocol for connecting to public resources and a client-server architecture, i.e. you’d use a browser application to reach pages on a remote server, but it had a more centralized, hierarchical way of structuring its contents, unlike the Web’s entirely freeform structure of linking from anywhere to anything. But Gopher’s simpler setup did give it a head start over the Web, so much so that Berners-Lee himself used Gopher to present his vision of the Web (Gihring 2017). But Gopher quickly fell out of favor when in 1993 when the University of Minnesota, the original proprietor of Gopher, announced it would charge licensing fees for commercial use of Gopher servers. This was seen as a breach of trust for anyone using,

especially the early adopters as the line between educational and commercial was deemed to be artificial and thus putting anyone tinkering with at risk of crossing it (Lee 1999). Meanwhile, CERN explicitly released the Web technology into the public domain with no fees, relinquishing "all intellectual property rights to this code, both source and binary form, and permission is granted for anyone to use, duplicate, modify and redistribute it" (CERN 1993: 2).

The topic of licenses as semiotic instruments of considerable consequence is a reoccurring theme in the development of the Web and its technologies. It will be analyzed in further detail in chapter 3.4.

2.2 Core technologies

Berners-Lee conceptualized the early web technologies already in his 1989 proposal but their naming and formal standardization took place gradually in the early 1990s. The three foundational technologies that continue to power the web are: HTML (HyperText Markup Language) for formatting and structuring documents; HTTP (HyperText Transfer Protocol) for retrieval of these documents; and the URL (Uniform Resource Locator)[9] which is their unique address on the web. Even as the web has evolved considerably over time and HTML's scope in particular has been vastly expanded as well as abstracted upon, the core operating principle of these technologies remains essentially the same as it was during its initial implementation: using a client application, usually a browser, you open a URL, a HTTP connection is established to a remote server and a new HTML document is loaded in.

HTML documents at their barest essentials are just text documents with additional pieces of code (markup) within it to further define the content and structure of the document. Text-based elements such as links and input fields are present *within* the HTML code, but more complex items such as images or videos are loaded in from external resources. HTML code contains a great amount of semantic specificity to separate regular text from titles, headers from footers

[9] URL's are actually a subcategory of URI's, or Uniform Resource Identifiers. URL's are how we commonly navigate through the Web: they are present in the address bar of a browser and as links they can be recognized by the http(s):// prefix, indicating the use of the HTTP protocol which is exclusive to URL's. Examples of other URI categories would be mailto: and tel:, indicating the use of an email system or telephony system respectively. As noted by Brian Kardell, for Tim Berners-Lee the URI was the most important aspect of his vision (Kardell 2015b).

and to clearly identify navigational items within a webpage. The primary reason for having all this defined within a document's structure is accessibility: technologies such as screen readers which audibly narrate the content of a webpage for people with visual impairments, can be used to much greater effect when such structures are in place (W3C WAI 2024). As Berners-Lee himself states, the "access by everyone regardless of disability is an essential aspect" of the Web (ibid.).

2.3 HTML Extended

Another two fundamental technologies arrived during the next years which extended functionalities within HTML considerably: CSS and JavaScript.

Håkon Wium Lie, a coworker of Berners-Lee at CERN, proposed Cascading Style Sheets (CSS) in October 1994 (Lie 1994) and following major contributions by Bert Bos, CSS1 was formally released in 1996. At its core CSS is used to separate the visual presentation from the actual content and structure of a HTML document. A CSS file defines a fixed "style" for elements such as fonts, colors and even layouts to be used across a document (or a whole website) while keeping the HTML document reasonably clean. A typical example of CSS in use is the difference between a desktop and mobile version of a website: the content is the same, but the formatting is adjusted based on where the content is viewed from.

The second extension was JavaScript. Created in 1995 in just ten days by Brendan Eich at Netscape, JavaScript was the key driver in changing the Web from a collection of static documents into an increasingly interactive and dynamic environment (Wirfs-Brock et al. 2020: 8). JavaScript eventually paved the way for websites to become web-based applications. JavaScript, its affordances as well as its limitations, are the foundation of React, the technology that will be analyzed in detail in chapter 3 of this thesis.

The earliest practical uses of JavaScript were input form validations, executing simple actions like pressing play for a media file and very early ways of altering the contents of a website itself or triggering additional events within the page. Examples at the time were mostly aesthetic in nature, such as scrolling text or changing certain colors on the page, but eventually more contentious items such as alert boxes and pop-ups came about. Ethan Zuckerman, the creator

of the `window.open()` function which enabled the use of pop-ups, would later on apologize for creating the technology and label pop-ups as “the internet’s original sin” (Zuckerman 2023).

2.4 Cookies

Even as JavaScript enabled certain forms of dynamism to websites, the early Web was, in technical parlance, stateless: there was no machine method of record-keeping if someone had visited a webpage, not for the server nor the visitor. When someone browsed around between different pages of a website, they’d be a blank slate for each page they’d visit, including the ones they return to. This became a key blocker for some of the first web-based commercial applications, web shops, and in 1994 led to the creation of the most recognized, if misunderstood web technology: the browser cookie. As explained by its creator Lou Montulli:

> “The specific use case that we were discussing at the time was to have a shopping cart model,” Montulli says. “So you're browsing items, and you say: I like those shoes. You click on the button that says: I want to buy these things. And you expect it to go in the cart—and not just disappear.” (Johnson 2022)

Montulli’s shopping cart use case set an interesting precedent for the Web: cookies can be considered one of the earliest examples of a core web technology introduced primarily in response to a commercial need rather than a technical one, even if the implementation itself was purely functional. Without a state (memory) to track, having something akin to a virtual, yet persistent shopping cart was not technically feasible. The solution was the creation of a small file—a “magic” cookie[10]—which would be stored onto the user’s machine, containing no other data than a unique identifier, thus establishing a high degree of privacy (Johnson 2022). Rather than giving each user a unique, trackable and permanent ID to be used across the Web, cookies were designed strictly to operate on a per-site basis. Going from one web store to another means the creation of a new cookie with neither store having access nor even being aware that the user has visited the other. And when the user returns to either web store it will request identifying

[10] The term 'magic cookie' predates web cookies, originating in Unix computing as a designation for capability tickets’ opaque data tokens (Magic cookie, s.a.). The exact origin of the term is disputed, but Montulli is credited with adapting the concept for HTTP state management.

data from the cookie stored on the person's machine and use that identifying data to load up the store cart from that particular store, establishing persistence.

The privacy-first design of the cookie found itself challenged withing a few years by the arrival of so-called third-party cookies. The two web stores described earlier would both act as first parties as the user is in *knowing* direct engagement with them. But both of them may choose to "provide" more than one identifying cookie to the user, this time from a third party: a separate web site, acting as a service, which is loaded in the background of the web store, without the person actively interacting with it or even being aware of it. This cookie will then be used whenever the customer visits another web site which uses that very same third-party service, be it another store, a news portal, a search engine or a social network (Cyphers et al. 2019).

Cookies make for a good example of Winner's argument on technologies becoming political outside of their intended use (Winner 1980: 125). The design and use of cookies was functional and specific: remembering cart contents, maintain login statuses and other, often minute conveniences. But when one takes their base functionality of identification, uses it to register and track the user's each and every interaction (recalling the earlier description of JavaScript's affordance of interactivity) and proceeds to extract further data from it, often in combination with other traces such as IP addresses, browser identifiers and several other elements, the cookie becomes the cornerstone technology of data extraction. It evolved into a foundational piece of infrastructure for what Shoshana Zuboff has termed surveillance capitalism: a business model based on the collection and commodification of private data, often without the user's knowledge or consent which is "grossly disfiguring the earlier dream of digital technology as an empowering and emancipatory force" (Zuboff 2019: 10). The technological solution to the shopping cart problem has morphed into a thousand anticipatory projections of a shopping cart, haunting us across the Web and every other platform it helped birth.

***

The early Web was designed as open, and in fundamental respects it remains so: anyone can still create a webpage using the core technologies Berners-Lee established. His requirements thus functioned as political choices materialized in technical design: by specifying what the Web would not constrain, he established boundaries against the concentration of control that characterized earlier systems. But the history of cookies demonstrates how technologies designed with specific boundaries in mind can be repurposed when actors exploit their base

functionality for ends their designers actively opposed. The cookie's identification mechanism, bounded by Montulli to operate per-site, became the cornerstone of cross-site, cross-Web surveillance structure once third parties circumvented that boundary.

The following chapter examines the open-source but privately owned framework React as a case where these dynamics play out within the practice of modern web development itself and how open-source code, private governance, and platform dependencies produce and proliferate new configurations of power.

3. React and corporate open-source

Before analyzing React, a library for building dynamic websites, as a specific technological artifact, it is necessary to situate it within the wider structural transformation of the Web that occurred between the early days described above and the modern day. While the introduction of the cookie (discussed in Chapter 2) signaled the appearance of commercial interest in the Web, the following decade saw a complete shift from the Web as a collection of static documents to a mass medium for complex software applications. This transformation is often referred to as "Web 2.0" (O'Reilly 2005).

The most dominant and complex of these applications were platforms such as Facebook (Plantin et al. 2016). Building these massive, centralized platforms on top of the decentralized, document-oriented standards of the early Web proved technically difficult and created a crisis of complexity. It is in this context that React, a library that abstracts JavaScript into an entirely new syntax, makes its appearance. What began as a tool to help solve a specific subset of problems has by 2025 evolved into a complex and market-dominant software solution with various intertwined dependencies – technological, material, societal – which extend far beyond its initial proposition.

In this chapter I will describe the history of React; its status as a corporate-owned open-source project and the politics that status entails; how React can configure a certain kind of web developer; and lastly how a diverse mixture of interests can lead to commercial encroachment of React.

3.1 History of React and its rise to dominance

The history of React begins in 2011 at Facebook (now Meta). The company was facing challenges that mirror those that Berners-Lee described in his early proposal, namely size,

complexity and compatibility problems with existing solutions. Facebook as a web application had expanded massively by that time and so did its set of features and requisite codebase. This in turn meant that knowing where exactly to change a line of code was terrifying enough that, as described by Jordan Walke, the engineer who created React, you could "potentially lose a day of revenue because you've missed a semicolon or something" (Hunt 2017). This complexity was exacerbated by Facebook's acquisition of Instagram in 2012. The mobile-first application needed a proper presence in the Web but operated on a separate codebase, making it essentially incompatible with Facebook's internal tools. React, written by Jordan Walke during his spare time, happened to be the most suitable solution available within Facebook to build out Instagram for the Web (ibid.)

The core need was the ability to have near-instantaneous dynamic changes within a single web page: whenever you browsed a feed, commented on a post or sent a message, the result should be on immediate display and thus avoiding the friction of moving to a new site. To achieve this, React introduced the idea of "composable" user interfaces which include components as their constituent parts. The latter are React's most essential building blocks: self-contained pieces of code that manage their own state and can be combined into larger structures. A component might represent a single button, a comment box, or an entire feed; each handles its own data and renders itself accordingly. This was coupled with the concept of a Virtual DOM (Document Object Model), which is essentially a lightweight copy of the actual website kept in memory. Rather than requiring developers to carefully track and update individual changes in the website on display, React would re-render the entire view in the background, then use the Virtual DOM to calculate and apply only the minimal actual changes needed. As an early blog post from React's authors states, "React really shines when your data changes over time" (Hunt 2013). The last ingredient was the introduction of JSX as the "syntactic sugar"[11]: a special syntax for writing components which allowed writing HTML *within* JavaScript – a novel approach at the time, as it blurred the lines between semantic structure (HTML) and actionable behavior (JavaScript) within a website.

Following internal implementation within Facebook and Instagram, React was introduced and open-sourced to the public in 2013. While its initial reception was lukewarm, with JSX drawing

[11] React Team, s.a.. JSX In Depth. React Documentation (Legacy). Accessed: https://legacy.reactjs.org/docs/jsx-in-depth.html, 13.11.2025

particular criticism (Ball & Nisi 2024), the following years would see growing adoption. The open-source nature of React helped form a community of developers around it who would build complementary tools: ready-to-use collections of components, many types of architectural templates and scaffoldings to help kick-start the building of different application types, and a variety of other solutions to build, preview and eventually deploy applications with minimal effort. These contributions, acting as "adoption drivers" (Noor 2024: 6–7), were not centrally coordinated by Facebook. They emerge from developers solving their own practical problems and sharing the results publicly as a form of contribution to the community.

The growing amount of such contributions meant that React was becoming less of a library, a tool among many to solve certain problems, and more of a "mature" framework and workflow for building web applications. To put it another way, "You don't just 'use React' anymore. You choose a constellation of tools that orbit it" (Williams 2025). Tools such as external component libraries, testing frameworks and "meta-frameworks" such as Next.js (discussed in subchapter 3.5) become "extensions of React's DNA" (ibid.). Therefore, using React right now often means you don't just choose a tool, you choose an entire toolbox, along with a set of best practices. This will see further discussion in subchapter 3.3.

Stack Overflow, the largest online community for software developers, has since 2011 published a Developer Survey to assess the popularity of different technologies developers are using, or plan to use in the future. In its 2021 survey, React seized the top spot within the most commonly used web frameworks and technologies category, and has maintained a dominant position ever since[12]. When it comes to practical deployment, current estimates show that React is being used by 6% of *all* websites, commanding a ten-fold lead over functionally similar front-end frameworks such as Vue.js and Angular[13]. Lastly, an important extension from the core React domain to mobile applications was introduced in 2015: React Native. It allows developing applications for Android and iOS using a single programming language, rather than having to write your apps individually using a platform-specific "native" language.

---

[12] Stack Overflow Developer Survey 2021. Accessed: https://survey.stackoverflow.co/2021 and Stack Overflow Developer Survey 2025. Accessed: https://survey.stackoverflow.co/2025/, 8.12.2025

[13] W3Techs. Usage Statistics and Market Share of JavaScript Libraries for Websites. Web Technology Surveys. Accessed November 13, 2025. Accessed: https://w3techs.com/technologies/overview/javascript_library, 8.12.2025

3.2 React and the politics of open-source software

Free and open-source software, or FOSS for short, is an umbrella term used to describe any piece of software that can be modified, (re)distributed and used for any purpose, be they non-commercial or commercial. A common way to designate a piece of software as FOSS is, somewhat paradoxically, by giving it a specific license which explicitly states all of the mentioned freedoms. The licenses are built on copyright law but written in a way that essentially precludes copyright and related abuses, giving it the colloquial name “copyleft”. The anthropologist Gabriella Coleman describes how FOSS is used by developers to “reconfigure central tenets of the liberal tradition—and the meanings of both freedom and speech—to defend against efforts to constrain their productive autonomy” (Coleman 2009), echoing a certain libertarian hacker ethos that helps to protect the boundaries of free expression within software development from encroachment by the interest of private capital.

This ideological framing is muddled when met with the reality of who maintains many widely-used open-source projects, especially when it concerns web development. Aditya Pandey's 2025 study of open-source developer tools states that companies maintaining frameworks as “corporate-backed open source” do so because such projects “serve strategic purposes for their parent companies” rather than generating direct revenue (Pandey 2025: 17). React fits this pattern in some ways but not others. Its core codebase, the part one can still refer to as a library, is primarily maintained by Meta employees, its direction was and, in many ways, still is shaped by Meta's internal needs. And its existence as a standalone project was itself a byproduct of corporate reorganization following the Instagram acquisition. At the same time, React is not a product that Meta monetizes in anyway; it generates no licensing fees, and Meta offers no commercial hosting or deployment services built around it. This is because Meta's revenue comes from advertising rather than infrastructure.

The strategic logic underlying Facebook's open source program can be viewed as what Joel Spolsky described as “commoditizing your complement”, i.e. the practice whereby technology-driven companies in particular get the price of their complements low as possible to enhance control over their core product (Spolsky 2002, Gwern 2018). A frontend framework such as React is a complement for Meta, necessary for building the web applications to serve the

advertisements they depend on. By open-sourcing React, Meta ensures this layer of the stack remains free, widely adopted and not controlled by other companies who such as Google, whose competing AngularJS framework React eventually replaced.

Open-sourcing did have tangible technical benefits for Facebook. In 2015, James Pearce, then head of open source at Facebook, noted that:

> We find we write better, cleaner code. We are forced to create more modular, pluggable technologies that can work both within and without the Facebook infrastructure (Clark 2015).

One could see this as flexibility which anticipates challenges yet to come: by keeping the framework open and modular enough to have it work outside of the requirements of Meta's *current* applications it can more easily be put to use for Meta's *future* applications. The amount of openly available practical solutions built on React, especially by larger customers, provide ample inspiration for any upcoming technical challenges Meta itself may face.

While these strategic and technical benefits explain why Meta permitted and supported open-sourcing, the main driver still came from within. Pete Hunt describes "a groundswell of support for open sourcing this React thing" among the developers themselves (Hunt 2017). Despite the initially hostile reception in 2013, the main development team was very active in various channels, answering questions and building community despite the initial failure (ibid.). The drive to share and build upon each other's work was genuinely present among React's creators, even while employed at Facebook.

3.3 Developer configuration and lock-in

I will now look at React within the lens of Suchman's configuration and exemplify how React as a socio-technical system "figures" (Suchman 2012: 49) a certain kind of developer through its documentation and the mental models it suggests, as well as how this affects web developers and web development more broadly by establishing lock-in, or the dependence on a set configuration that forecloses alternatives (Plantin et al. 2016: 9).

James Pearce noted in a 2015 interview the impact on resourcing and hiring their open-source projects have had, noting that open source serves as "a great contribution to our overall engineering brand," with candidates actively joining on account of these projects (Clark 2015). Pete Hunt, a core React Developer, is even more blunt by stating that in 2013 people "hated Facebook" and that React was critical in reversing that image (Hunt 2017). But this influence flows both ways. If developers are drawn to Facebook through using React, React may have also shaped how they think and work. Using it helps configure a certain kind of developer, with a cascading set of impacts on web development as a whole.

The official documentation makes this effect explicit. The tutorial "Thinking in React" (React Documentation s.a.) opens with a bold claim: "React can change how you think about the designs you look at and the apps you build", implying a cognitive transformation of pre-existing mental models. The page prescribes a five-step process:

1) break designs into components,
2) build a static version (the virtual DOM mentioned in 3.1),
3) identify the minimal state of your UI,
4) determine where and which components define this state,
5) add inverse data flow.

Developers are instructed to "start by drawing boxes" around interface elements before writing any code and how these drawings will match your code (if well written) and therefore "naturally map to the component structure of your UI" (React Documentation s.a.). The word "naturally" is presented as an organic correspondence between data and display. Recalling Suchman's concept of figuration, an action that "holds the material and semiotic together in ways that

become naturalized over time" (Suchman 2012: 49), an analogy can be observed. The five-step process and the mental model it offers, once internalized by developers for being natural, becomes the dominant default on how web interfaces are designed, built and configured.

When observing the core technological stack of the Web discussed in the previous chapter, using a framework such as React based on the principles outlined in the process above it can begin to marginalize more freeform or framework-less approaches to web development, rendering them less visible and less viable in practice. The more dominant frameworks (not just React) become, the more the Web begins to look like them, as well as the way it is built by the developers using those frameworks.

When it comes to actual software projects, once the decision has been made to use a technology such as React (or "settled" via a corporate decree on software architecture) it is very difficult to switch as it necessitates extensive rewrites and therefore incurs costs. This is an example of Winner's first of two choices when it concerns the implementation of a technology: a simple "yes or no", do we go ahead and adopt this choice (Winner 1980: 127). This choice will fundamentally dominate all future choices in this regard as you are operating only within it, with its architectural makeups and patterns. Each new implementation within it excludes alternatives further. Plantin et al. notes how the primary goal of platform-builders is achieving lock-in, meaning "the accumulation of dependencies that make alternatives impractical" (Plantin et al. 2016: 9). In many ways the choice of a framework such as React locks you in the very same way, even without there being an actual platform in place to lock you in. Such scenarios can eventually arise and will be elaborated upon in chapter 3.5.

Lastly, with extensive operation *within* a framework such as React, it becomes more and more difficult to operate as efficiently *outside* of this framework as well. Joel Hassan Noor's 2024 study of JavaScript frameworks documents this aspect. Framework abstractions, he finds, "carry the risk of instilling framework-specific mental models and ways of working that do not easily transfer beyond that framework" (Noor 2024: 24). Developers, less experienced ones in particular, acquire "in-depth knowledge of a particular framework, which could impede their ability to transition efficiently to alternative frameworks or framework-less development" (ibid.). Professional autonomy becomes constrained from opposing directions: developers become dependent on frameworks for employment (as React skills become hiring criteria) while simultaneously losing facility with alternatives.

When "Thinking in React" becomes the epistemological operator lock-in is not only technological but also cognitive. Meta can capitalize on this not merely by hiring developers fluent in React, but by shaping the Web itself through a particular vision of how its technologies should be built and used.

3.4 Facebook v. WordPress: licensing as a contested boundary

Software licenses are documents that attempt to fix boundaries: what "free" and "open" mean, who may do what with code, and under what conditions. They configure relationships between technology, its creators, and its users through legal language. But as Suchman notes, such boundaries "may appear as if they were present from the start when in actuality they are redrawn continuously" (Suchman 2012: 50). A controversy related to Facebook, the initial BSD+Patents license by which it open-sourced React, and how different parties saw to interpret and apply it serves to exemplify the complexities involved.

React's open source licensing wasn't always the highly permissive MIT license it now uses: when it was initially open-sourced in 2013 it used a rather specific license called BSD+Patents. The latter part – Patents – essentially states that anyone using React would lose their patent license from Facebook if they sued Facebook for patent infringement on *any* matter, not just React-related patents (O'Grady 2017). What this means in practice is that when someone builds a web application using React, they essentially lose any judicial recourse in case Facebook infringes on one of their patents. Given the company's propensity to adopt ideas from competitors[14] as well as using copyrighted material to train their AI models (Milno, 2025) this can constitute a serious risk, even if Facebook themselves claim that the sole purpose of that license is to avoid frivolous lawsuits (Wolff 2017).

[14] Smolders, T. "10 Times Facebook Copied a Competitor." *Medium*, July 12, 2018. https://medium.com/ljosmyndun/10-times-facebook-copied-a-competitor-bda61d2d5c06.

In July 2017 the Apache Software Foundation, a non-profit governance body for FOSS licenses, added BSD+Patents to its "Category X" list of disallowed licenses[15]. Facebook's initial response was refusal: on August 18, 2017, the company published a post explicitly declining to change the licensing terms (Wolff 2017). This in turn prompted Wordpress, a hugely popular platform used to build roughly 25% of all websites at the time, to discuss a full excision anything React-related from their codebase – a monumental task (O'Grady 2017). The reasoning given was that the license is threating to many people and that there is a core long-term consistency which is worth more than a short-term delay in new features. A month later Facebook relented and gave up on the Patents license and relicensed React under MIT, a highly permissive license used by open source projects of all kinds (ibid.). The sequence suggests that while the Apache ban and developer petitions had failed to move Facebook, the threat of losing WordPress, and a sort of omnipresence in the Web being within its codebase provides, was cause enough to reconsider.

This particular episode is an example of how the Web that was, a set of fixed documents exposed to an audience, has become a complex array of technological configurations with assumed borders that are continually redrawn. Licenses are foundational elements of open-source software, yet practically invisible to most (web) developers. Changes to them, or the boundaries of use they define, may cause a rewrite of a vast amount websites they affect. An alternative conclusion for the previous story may have involved a forceful rewrite of Wordpress with millions of affected websites which run afoul of certain licensing rules unless they promptly update themselves. Or perhaps WordPress would have kept React, pushing users to abandon a platform that remains the standard for non-technical website creation. And, lastly, perhaps Facebook would have found some ways to litigate against a competitor, which would have led to various well-known platforms to either face the risk or use some other framework else to build their product. A series of future imaginaries which can be narrated, their borders enacted just because one particular document, assumed to be external, redrew the foundations of the wider Web.

Suchman references a study by Judith Gregory between 1993 to 1998 on the implementation of a health record software solution in the US (Gregory 2009). In it she discusses the competing

---

[15] This was decided within the comments section of an entirely separate issue: https://issues.apache.org/jira/browse/LEGAL-303?focusedCommentId=16088663.

imaginaries of the actors involved in building it and the varying subject/object configurations that motivated them:

> [The] effective manager made possible through the comprehensive decision support system, the innovative design company with its leading-edge prototype, the researcher with a perfect database prescribing error-free clinical practice, and the care provider with an always ready-to-hand patient record. (Suchman 2021: 51)

The license controversy can be viewed in a similar light: how different actors with competing views of what open-source means and what the implications are of licensing something as such. The list of actors could be as follows

1. The React team who built the framework, motivated primarily by their desire to build and share a technological framework that drives the development of the Web;
2. Facebook and their legal team who look to maintain some control over React as well as wanting to protect themselves from litigation;
3. The Apache Software Foundation who set the standards of what can be considered to be truly FOSS and thus free of legal restrictions;
4. The Wordpress team who feel inclined to follow the same principles as the foundation as a core value and as a way to protect the users of their platform;
5. The website owner, who has to accept whichever decision is made, probably unaware that they are even doing so.

And one can once more infer the manifestly absent user who, in an alternative timeline, may have seen the Web being shaped by a different set of technologies.

The licensing controversy reveals how corporate control draws boundaries through legal instruments that appear peripheral to technical function. Facebook's BSD+Patents license attempted to establish a boundary that the open-source community ultimately refused to accept as legitimate. The boundary was redrawn only when a sufficiently powerful actor (WordPress) threatened material consequences.

3.5 The influence of Vercel

Next.js is an open-source "meta-framework" (Noor 2024) built on top of React that adds features such as server-side rendering, routing, and deployment that React itself does not directly provide. Next.js is an entirely autonomous outgrowth of React and is an abstraction built upon an abstraction: React abstracts direct manipulation of JavaScript, and Next.js abstracts away direct use of React. Next.js was released as open source in 2016 and is built by Vercel, a private company which operates independently from Meta and therefore has no direct financial ties to it. While React/Next.js projects are not the only hosting Vercel provides, it is seen as a core revenue driver, "making it vulnerable if competing frameworks gain significant market share" (Sacra s.a.). Vercel's business incentives are directly linked to Next.js' popularity, how well it accommodates React and in turn how popular React remains.

One of the earliest steps of Vercel assuming a stake of ownership involves resourcing. The React development team had so far been in full employ of Meta. In 2021, Vercel and React began to merge together *in corpore* when Vercel hired React team lead Sebastian Markbåge from Meta, with several other members following thereafter (Vercel Blog 2021, Erikson 2025). Most still remain at Meta, but the concept of a React team member now extends to two patrons with different stakes and different objectives.

The dynamics between Meta, Vercel, Next.js and React illustrate how the direction of free and open-source technologies and dynamics of ownership become ambiguous when external commercial interests appear to have more vested interests in them than their primary caretakers.

3.5.1 React Server Components and the necessity of frameworks

What makes Next.js indispensable is the fact that it is a key (and currently only available) technical enabler for the most recent architectural shift within React: React Server Components (RSC) (Erikson 2025). While React originally expanded the degree of interactivity on a webpage, RSC lets the server hosting the page handle some of the functions instead, increasing the perceived performance of websites (React Blog 2023). It is described as an "unusual hybrid feature": it is built into React proper, but requires a separate framework implementation such

as Next.js as well as related infrastructure to function (Erikson 2025). This has material consequences in the form of hardware and infrastructure requirements, shifting React from an infrastructurally-agnostic library to one where making full use of its feature set forces strict choices. The framework layer (Next.js) mediates between the library (React) and the infrastructure (Vercel or alternatives), but RSC's architecture ensures that some infrastructure is necessary.

A key point of contestation is political in nature and involves the question of who actually determines the choice of which path React now proceeds upon. When considering Vercel to be a platform it can be argued that it has a clear incentive (and the means) to make the decisions which would "platformize", in the sense proposed by Plantin et al. (2016) React: to shift it towards a direction where Next.js becomes the de-facto standard for using React (with some similarities to how React became the dominant standard for using JavaScript) and from there the path of least resistance is choosing Vercel for hosting. The necessitation of Next.js can also be seen in how React's official documentation has changed over time. The original "Create React App" launcher by Meta was discontinued in favor of Next.js, which is now positioned at the top and in bold as being the best way to take "full advantage of React's architecture", whereas building a React application without a framework is framed as a challenge akin to "building your own framework" (React Documentation 2025).

Narratives countering this framing of Next.js/Vercel dominance over React and the RSC implementation in particular also exist. Mark Erikson, a key contributor to React-adjacent tools, offers a different perspective on the question of who settled the RSC vision:

> "RSCs were the React team's vision. They couldn't effectively be prototyped and tested inside of Meta, so for the first time there was a need for some other sponsor to invest in the development and provide a setting to iterate on the design… [T]he React team convinced Vercel to buy into the React team's vision, and let them drive rearchitecting Next to design the App Router approach that would match that vision." (Erikson 2025)

This framing implies a great deal of autonomy to the React Team, operating on the imaginaries that they as designers may have, independent of their employers. Yet as a technological project is fundamentally a "entanglement of imaginaries and artefacts" (Suchman 2012: 57) there is, whether by accident or by design, a "stacking of the deck" (Winner 1980: 125) in favor of Vercel's interests.

It has to be emphasized that making use of RSC and its infrastructure requirements is optional: a web application without RSC can still be built and deployed. But there is a growing sense of there now being "Two Reacts": one that adheres to the original UI focus of React and another that includes RSC and extends into the server space (Comeau 2023, Abramov 2024). In Suchman's terms they could be viewed as the competing imaginaries for React: one narrating a strict focus on the interactive dynamics of a single page and another envisioning a wider focus which involves server-side complexities, both also constituting different mental models for use of the core technology (Suchman 2012: 48). For the last few years, the direction of React has been purely towards the server-side imaginary. But the existence of this discourse implies that the matter is not yet settled and recent development have brought about new criticisms of this server-side focus. Namely, the introduction of server-side elements to React has also exposed it server-side vulnerabilities, as exemplified by React's recent critical vulnerability which received the maximum score of 10 on the CVSS scale for severity (React Blog 2025). While the vulnerability only affects React developments which actually use RSC, its existence is signal enough that this new direction carries new potential consequences, irrespective of how React is being used.

The growth of Vercel's influence can be observed through three interconnected mechanisms: hiring of key personnel, shaping architectural decisions within React that favor infrastructural dependency over self-hosting, and altering official documentation to position Next.js as default.. And yet, as Mark Erikson argues Vercel can also be construed as only a "sponsor" for achieving the goals of a React core team, operating with autonomy to further their own vision of the technology. Platformization only occurs as a side effect of these technological roadmaps. This, in turn, reveals how much of a closed black box a nominally open-source project can be.

3.5.2 OpenNext as an example of open-sourcing in practice

Mechanisms of encroachment within technology are often challenged. On its official website Next.js is described as a "collaborative open-source effort"[16] but as Mathias Biilmann, CEO of a competing hosting provider Netlify observes, it occupies "a weird middle ground between open source and closed source" (Branscombe 2024). Its source code is public and the

[16] Vercel Governance, Accessed: https://nextjs.org/governance, 07.01.2026

framework can be hosted outside of Vercel's servers but a set of its features are heavily optimized for Vercel's infrastructure and require extensive setup and configuration to work in any other environment. As Dax Raad, the maintainer of the documentation project OpenNext explains: "It's not that Next.js isn't open source. It's how to get every single Next.js feature listed in the Next.js documentation actually running in all kinds of environments—that information is just not open" (ibid.). OpenNext functions primarily as a documentation and tooling project that makes deployment to platforms such as AWS, Cloudflare, and Netlify possible by reverse-engineering [17] what Vercel does internally. Unlike other React-based frameworks which fully document self-hosting options, Next.js contains references to Vercel's proprietary infrastructure code without explaining how to replicate its behavior elsewhere (ibid.). While Raad doesn't see this as malicious but rather as something inherent to vertically integrated frameworks, he does see this as something one has to "buy into" metaphorically, but also literally (ibid.). Oftentimes one is already "bought into" some other platform or infrastructure; shifting your whole web application from AWS to Vercel just to have some specific functionalities available might not be feasible, or possibly even permitted as large corporations often enforce strict data security standards.

In Suchman's terms, OpenNext may represent an attempt at reanimation: making visible the "obscured" practices of a technology which are "presupposed and built into particular technological artefacts" (Suchman 2012: 49-50). Vercel, through Next.js, maintains a form of infrastructural monopoly over React which OpenNext as a form of unpaid, community-based labor aims to circumvent. That such a project is necessary at all reveals how "open-source" can function as a boundary that appears fixed and immutable, but is still open to collaborative and collective change, thus directly mirroring Next.js own description of being a "collaborative open-source effort to build tools for React and Web developers," but in actual, non-commercial practice.

[17] Reverse-engineering is a process of analyzing and dismantling a piece of software, hardware or even a biological system to understand how exactly it was designed and built, and how it functions.

4. Conclusion

As Winner states, we would be “blinded to much that is intellectually and practically crucial” if our evaluation of technology includes only categories having to do with tools and uses, and does not attend to “the meaning of the designs and arrangements of our artifacts” (Winner 1980: 125).

Within this thesis I wished to argue how the Web, as conceived by Tim Berners-Lee, embedded politics into its design to a considerable degree. His initial requirements of free, unrestricted and open access, non-centralization, heterogeneity and his own emphasis on how “the method of storage must not place its own restraints on the information” (Berners-Lee: 5) established a bold, yet highly democratizing vision for the Web and embedded strict political arrangements by establishing boundaries against control in any form. Yet even within this open architecture, technologies can be repurposed beyond their designers’ intentions. The history of cookies shows how a privacy-preserving solution, once its identification mechanism was exploited by third parties, became foundational to surveillance capitalism.

I then proceeded to argue how different kinds of arrangements within the Web’s technologies, be they as foundational as cookies or as evolved as a meta-framework like Next.js and platforms like Vercel, seek to redraw and reconfigure the boundaries of open-source technologies the Web was built upon. Corporate control over open-source frameworks can create ambiguities wherein the nature of this freedom can be challenged through licenses, the community as well as overall dominance within the Web. Frameworks such as React also shape those who use them, exemplified by the specific mental models that, once internalized, create lock-in that is both technological and cognitive, making use of alternatives an increasing challenge.

The case of Vercel illustrates how platform interests can encroach upon open-source projects such as React through various means, but also how the complexities of ownership and the multiplicity of actors can obscure this process. And finally, these mechanisms always face

resistance: OpenNext works to reverse-engineer Vercel's advantages, and the ongoing "Two Reacts" debate suggests that such matters are rarely truly settled.

The Web of the year 2026 finds itself at another moment of contestation. On the one hand we have the explosive rise of AI tools as a near-existential threat to the Web: being trained on data extracted from the Web they seek to effectively replace it with the walled garden of a chat interface; and being purpose-built tools for data reproduction and repetition, they are polluting the Web further with the so-called slop they generate. Further yet, browsers like OpenAI's Atlas promise to "unlock" the Web as it seeks to replace browsing, the core interaction with the Web, entirely with "ChatGPT by your side" (OpenAI 2025). These are all topics which warrant further investigation, given their profound impacts.

On the other hand, initiatives like the IndieWeb[18], new approaches to search like Kagi's Small Web[19] and the concept of a Post-Naive Internet (Matusek et al. 2025) all seek to reclaim the digital territory we have lost, aiming to re-establish the original ideas and a refreshed politics for a better Web of the future. I hope for this thesis to be a small contribution to that pursuit.

[18] The IndieWeb is a "a community of independent and personal websites", aiming for more personal control over ones' online identity through website ownership. See: https://indieweb.org/

[19] Kagi is a paid search engine that allows restricting search results strictly to smaller, personal websites and avoiding bigger platforms. See: https://blog.kagi.com/small-web

References

Abramov, Dan 2024. The Two Reacts. *Overreacted*. Accessed: https://overreacted.io/the-two-reacts/, 23.12.2025.

Ball, Kevin; Nisi, Nick 2024. React, JSX, and the Early Frontend Wars. *JS Party, episode 349, Changelog Media.* Accessed: https://changelog.com/jsparty/349, 12.11.2025

Berners-Lee, Tim. 1989. Information Management: A Proposal. *CERN*. Accessed: https://cds.cern.ch/record/1405411/files/ARCH-WWW-4-010.pdf, 11.01.2026

Berners-Lee, Tim. 1990. WorldWideWeb: Proposal for a HyperText Project. *CERN*. Accessed: https://repository.cern/records/6kxvc-v6203/preview/dd-89-001.pdf, 11.01.2026

Branscombe, Mary. 2024. OpenNext Gets Closer to Making Next.js Truly Portable. The New Stack. https://thenewstack.io/opennext-gets-closer-to-making-next-js-truly-portable/, Accessed: 28.12.2025

CERN 1993. Software release of WWW into public domain. *CERN*. Accessed: https://cds.cern.ch/record/1164399, 12.12.2025

Clark, Libby 2015. Facebook's James Pearce: Open Source Creates More Quality Code. — Linux.com. Accessed: https://www.linux.com/news/facebooks-james-pearce-open-source-creates-more-quality-code/, 29.12.2025

Coleman, Gabriella 2008. Code Is Speech: Legal Tinkering, Expertise, and Protest among Free and Open Source Software Developers," *Cultural Anthropology* 24(3) (2009): 420-454.

Comeau, Josh 2023. State of React 2023 Conclusion. Accessed: https://2023.stateofreact.com/en-US/conclusion/, 23.12.2025

Cyphers, Bennett and Gennie Gebhart 2019. Behind the One-Way Mirror: A Deep Dive Into the Technology of Corporate Surveillance. Electronic Frontier Foundation. Accessed: https://www.eff.org/wp/behind-the-one-way-mirror, 26.12.2025

de Reuver, Mark; Sørensen, Carsten; Basole, Rahul C. 2018. The digital platform: a research agenda. *Journal of Information Technology* 33(2): 124–135.

Gihring, Tim 2017. The rise and fall of the Gopher protocol. *MinnPost.* Accessed: https://www.minnpost.com/business/2016/08/rise-and-fall-gopher-protocol/, 05.12.2025

Erikson, Mark 2025. The State of React and the Community in 2025. Mark's Dev Blog. Accessed: https://blog.isquaredsoftware.com/2025/06/react-community-2025/, 21.12.2025

Gregory, Judith 2009. A complex model for international and inter-cultural collaboration in health information systems and higher education, in S. Poggenpohl and K. Sato (eds) *Design Integrations: Research and Collaboration*, Chicago: Intellect, University of Chicago Press

Gwern 2018. Laws of Tech: Commoditize Your Complement. — Gwern.net. Accessed: https://gwern.net/complement, 29.12.2025

Hunt, Pete 2013. Why Did We Build React? *React Blog*. Accessed: https://legacy.reactjs.org/blog/2013/06/05/why-react.html, 13.11.2025

Hunt, Pete 2017. Interview. In: The React Story. Stack Stories podcast, Episode 3. Accessed: https://stackshare.io/posts/the-react-story, 23.12.2025

Johnson, Steven 2022. The Magic Cookie: How Lou Montulli cured the Web's amnesia. Accessed: https://hiddenheroes.netguru.com/lou-montulli, 05.12.2025

Kardell, Brian 2015a. A Brief(ish) History of the Web Universe – Part I: The Pre-Web. *briankardell* (blog). Accessed: https://briankardell.wordpress.com/2015/11/22/a-briefish-history-of-the-web-universe-part-i-the-pre-web/, 04.12.2025

Kardell, Brian 2015b. A Brief(ish) History of the Web Universe – Part II: Time. *briankardell* (blog). Accessed: https://briankardell.wordpress.com/2015/12/07/a-briefish-history-of-the-web-universe-part-ii-time/, 04.12.2025

Lee, Christopher 1999. Where Have all the Gophers Gone? Why the Web beat Gopher in the Battle for Protocol Mind Share. Accessed: http://ils.unc.edu/callee/gopherpaper.htm, 06.12.2025

Lie, Håkon Wium 1994. Cascading HTML style sheets -- a proposal. Accessed: https://www.w3.org/People/howcome/p/cascade.html, 03.12.2025.

Magic Cookie (s.a.). — The Jargon File. Accessed: https://jargon-i18n.com/en/M/magic-cookie.html, 22.12.2025

Matusek, Severin; Houde, Nick; Moniz, Paloma 2025. Welcome to the Post-Naive Internet Era. — Mozilla Foundation. Accessed: https://www.mozillafoundation.org/en/nothing-personal/the-post-naive-internet-era/, 13.01.2026

Milmo, D. 2025. Zuckerberg Approved Meta's Use of 'Pirated' Books to Train AI Models, Authors Claim. The Guardian. Accessed: https://www.theguardian.com/tehnology/2025/jan/10/mark-zuckerberg-meta-books-ai-models-sarah-silverman, 14.12.2025

Noor, Joel Hassan 2024. The Effects of Architectural Design Decisions on Framework Adoption: A Comparative Evaluation of Meta-Frameworks in Modern Web Development. Master's thesis, Aalto University.

O'Grady, Stephen. 2017. Facebook's About Face. RedMonk (blog). Accessed: https://redmonk.com/sograd y/2017/09/26/facebooks-bsd-patents/, 06.01.2026

O'Reilly, Tim 2005. What Is Web 2.0: Design Patterns and Business Models for the Next Generation of Software. — O'Reilly Media. Accessed: https://www.oreilly.com/pub/a/web2/archive/what-is-web-20.html, 07.01.2026

OpenAI 2025. Introducing ChatGPT Atlas. — OpenAI. Accessed: https://openai.com/index/introducing-chatgpt-atlas/, 13.01.2026

Pandey, Aditya 2025. The Economics of Open Source Dev Tools: A Quantitative Analysis of GitHub Stars, Funding, and Revenue Models 2020-2025. Independent research report.

Plantin, J., Lagoze, C., Edwards, P. N., & Sandvig, C. (2016). Infrastructure studies meet platform studies in the age of Google and Facebook. *New Media & Society*, *20*(1), 293–310.

React Blog 2023. React Labs: What We've Been Working On — React Blog. Accessed: https://react.dev/blog/2023/03/22/react-labs-what-we-have-been-working-on-march-2023, 22.12.2025

React Blog 2025. Critical Security Vulnerability in React Server Components. — React Blog Accessed: https://react.dev/blog/2025/12/03/critical-security-vulnerability-in-react-server-components, 11.01.2026

React Documentation 2025. Start a New React Project 2025. — React Documentation. Accessed: https://react.dev/learn/start-a-new-react-project, 12.12.2025

React Documentation s.a. Thinking in React. — React Documentation Accessed: https://react.dev/learn/thinking-in-react, 07.01.2026

Sacra (s.a.). Vercel: Revenue, Valuation & Funding. Accessed: https://sacra.com/c/vercel/, 21.12.2025

Sandoz, Alain; Stiefel, Léa 2022. Untying the knot between software-based platforms and information infrastructures. In: *Proceedings of the 30th European Conference on Information Systems* (ECIS 2022).

Spolsky, Joel 2002. Strategy Letter V: The Economics of Open Source. — Joel on Software. Accessed: https://www.joelonsoftware.com/2002/06/12/strategy-letter-v/, 29.12.2025

Suchman, Lucy 2012. Configuration. In C. Lury and N. Wakeford (eds.), *Inventive Methods: The Happening of the Social* (pp. 48-60). London: Routledge.

Vercel Blog 2021. Supporting the Future of React. Vercel Blog. Accessed: https://vercel.com/blog/supporting-the-future-of-react, 18.12.2025

W3C WAI 2024. Introduction to Web Accessibility. Accessed: https://www.w3.org/WAI/fundamentals/accessibility-intro/, 14.12.2025

Wirfs-Brock, Allen; Eich, Brendan 2020. JavaScript: The First 20 Years. Accessed: https://dl.acm.org/doi/10.1145/3386327, 18.12.2025

Williams, Alexander 2025. Why React's 'Boring' Maturity Is Actually Its Main Strength. The New Stack. Accessed: https://thenewstack.io/why-reacts-boring-maturity-is-actually-its-main-strength/, 11.01.2026

Winner, Langdon 1980. Do artifacts have politics? *Daedalus, 109*(1), 121–136.

Winner, Langdon 1986. The Whale and the Reactor: A Search for Limits in an Age of High Technology. Chicago: The University of Chicago Press.

Wolff, Adam. 2017. Explaining React's License. Engineering at Meta. Accessed: https://engineering.fb.com/2017/08/18/open-source/explaining-react-s-license/, 11.12.2025

Zuboff, Shoshana 2019. Surveillance Capitalism and the Challenge of Collective Action. New Labor Forum 28(1): 10–29. DOI: 10.1177/1095796018819461.

Zuckerman, Ethan 2023. The Internet's Original Sin. The Atlantic. Accessed: https://www.theatlantic.com/technology/archive/2014/08/advertising-is-the-internets-original-sin/376041/, 06.12.2025

Resümee

Mu bakalauerusetöö ”Veebi ümberkonfigureerimine: avatud standardid, platvormivõim ja arendajate konfigureerimine” uurib, kuidas algselt avatuks ja detsentraliseerituks kujundatud veebitehnoloogiad võidakse eri mehhanismide kaudu erahuvide teenimisele suunata. Töö teoreetiline raamistik tugineb Langdon Winner’i esseedel tehnoloogiste artefaktide võimudünaamikaid ilmestavatele poliitilistele tahkudele (1980, 1986), Lucy Suchman’i konfiguratsiooni mõistele (2012) ning platvormiuuringute infrastrukturaliseerumise ja lukustumise kontseptsioonidele.

Winner’i järgi pole tehnoloogiad neutraalsed tööriistad, vaid võivad konkreetseid võimusuhteid nii kehastada kui ka jõustada. Suchman’i konfiguratsioon võimaldab analüüsida, kuidas tehnoloogiad ja kasutajad teineteist kujundavad piiride pideva ümberjoonistamise kaudu. Platvormiuuringud kirjeldavad mehhanisme, mille kaudu saavutavad domineeriva positsiooni.

Empiiriline analüüs jaotub kaheks osaks. Alustuseks vaadeldakse Tim Berners-Lee algset kavandit veebi jaoks ja selles sisalduvaid poliitilisi valikuid avatuse ja detsentraliseerituse suunas. Sellele järgnev veebiküpsiste algseid ambitsioone ja hilisemat kuritarvitamist analüüs demonstreerib kuidas privaatsust kaitsvaks kujundatud lahendus võib muutuda seda kuritarvitavaks tehnoloogiliseks arhitektuuriks, illustreerides Winneri väidet tehnoloogiate poliitilisest ümbermõtestamisest väljaspool kavandatud kasutust.

Teises osas analüüsitakse Facebook’i tarbeks loodud JavaScripti raamistikku React ja sellega seonduvate võimusuhete nüansse. Töö käsitleb React’i ajalugu, korporatiivomandis oleva vabavaralise tehnoloogia eripärasid ning litsentse kui võimudünaamikaid ilmestavad mehhanismid. Suchman’i mõisteid rakendades analüüsitakse, kuidas React’i dokumentatsioon konfigureerib arendajate mõttemustreid, luues nii tehnoloogilist kui ka kognitiivset lukustumist. Viimaks vaadeldakse Vercel’i kasvavat mõju React’i arengusuuna üle, illustreerimaks kuidas avatud lähtekoodiga tehnoloogia võib sattuda platvormihuvidele allutatuks ning viimaks kuidas neid tehnoloogiaid Suchman’i mõistes taaselustatakse.

Töö lõppsõna juhib tähelepanu AI tööriistade plahvatusliku kasvu mõju veebile ning samas ka positiivsetele, poliitiliselt informeeritud arengusuundadele veebi käsitlemises.

Lihtlitsents lõputöö reprodutseerimiseks ja üldsusele kättesaadavaks tegemiseks

Mina, Risto Rõõmus ,
(autori nimi)

1. annan Tartu Ülikoolile tasuta loa (lihtlitsentsi) minu loodud teose
The Reconfiguration of the Web: Open Standards, Platform Power, and Developer Configuration ,
(lõputöö pealkiri)

mille juhendaja(d) on Katre Pärn ja Auli Viidalepp ,
(juhendaja nimi)

reprodutseerimiseks eesmärgiga seda säilitada, sealhulgas lisada Tartu Ülikooli digitaalarhiivi kuni autoriõiguse kehtivuse lõppemiseni;

2. annan Tartu Ülikoolile loa teha punktis 1 nimetatud teos üldsusele kättesaadavaks Tartu Ülikooli veebikeskkonna, sealhulgas digitaalarhiivi kaudu Creative Commonsi litsentsiga CC BY NC ND 4.0, mis lubab autorile viidates teost reprodutseerida, levitada ja üldsusele suunata ning keelab luua tuletatud teost ja kasutada teost ärieesmärgil, kuni autoriõiguse kehtivuse lõppemiseni;

3. olen teadlik, et punktides 1 ja 2 nimetatud õigused jäävad alles ka autorile;

4. kinnitan, et lihtlitsentsi andmisega ei riku ma teiste isikute intellektuaalomandi ega isikuandmete kaitse õigusaktidest tulenevaid õigusi.

Risto Rõõmus
13.01.2026